Identifying misinformation and their sources in social networks

Tang, Wenchang

2020

Tang, W. (2020). Identifying misinformation and their sources in social networks. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/138568>

<https://doi.org/10.32657/10356/138568>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:15:12 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

**IDENTIFYING MISINFORMATION AND THEIR
SOURCES IN SOCIAL NETWORKS**

TANG WENCHANG

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING
2020**

IDENTIFYING MISINFORMATION AND THEIR SOURCES IN SOCIAL NETWORKS

TANG WENCHANG

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

May 1, 2020

.....

Date

Tang Wenchang

.....

TANG WENCHANG

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

May 1, 2020

.....

Date

.....

Prof. Tay WEE PENG

Authorship Attribution Statement

This thesis contains material from 2 papers published in the following peer-reviewed journals and 1 paper accepted at conference in which I am listed as an author.

Chapter 2 is published as [F. Ji, W. Tang, W. P. Tay, and E. K. P. Chong, “Network topology inference using information cascades with limited statistical knowledge,” Information and Inference: A Journal of the IMA, 2019.](#) F. Ji and W. Tang contributed equally to this work.

The contributions of the co-authors are as follows:

- F. Ji proposed the initial idea and provided the theoretical results.
- I designed the algorithms and conducted the experiments.
- I and F. Ji prepared the manuscript.
- W. P. Tay revised the manuscript.

Chapter 3 is published as [W. Tang and W. P. Tay, “A particle filter for sequential infection source estimation,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., Mar. 2017.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, designed the algorithm, conducted the experiments, and prepared the manuscript.
- W. P. Tay revised the manuscript.

Chapter 4 and 5 are published as [W. Tang, F. Ji, and W. P. Tay, “Estimating infection sources in networks using partial timestamps,” IEEE Trans. Inf. Forensics Secur., vol. 13, no. 2, pp. 3035-3049, Dec. 2018.](#)

The contributions of the co-authors are as follows:

- I proposed the idea, designed the algorithms, conducted the experiments, and prepared the manuscript.
- F. Ji developed the theory of Gromov matrix and assisted in the algorithm design.
- W. P. Tay revised the manuscript.

May 1, 2020

.....

Date

Tang Wenchang

.....

TANG WENCHANG

Acknowledgements

I first wish to express my greatest gratitude to my supervisor, Professor Tay Wee Peng for the continuous support of my Ph.D research, for his enthusiasm, patience and immense knowledge. During my Ph.D. studies, he provided much freedom for my research. When I got stuck, he gave me encouragement and direction. He also taught me how to write a high-quality scientific paper by editing my manuscripts carefully and pointing out the areas to improve. I learnt what a true researcher should be from him. Moreover, he arranged group study sessions regularly to learn advanced mathematics as well as cutting edge technologies. All of these will no doubt be beneficial for my future development.

I would like to thank Dr. Ji Feng. I discussed with him on many of my research works and projects and was benefited from his insightful suggestions. I am also thankful to all my talented friends who have made my postgraduate life colorful and memorable, including Kang Qiyu, Yang Jielong, Lau Tze Siong, Sun Meng, Wang Chengcheng, Wang Yuan, Wang Junshan, Zeng Yijie, Lu Yiqi, and other lovely friends. Thanks for their accompany, for the inspiring discussions, and for all the good times we have had together in the last several years. A very special thanks goes to my girlfriend Li Yuting, for all her love and support.

Last but not least, I would like to thank my beloved parents, sister, brother and other family members. This thesis is dedicated to them for their unconditional love, support and understanding throughout my life.

"Try not to become a man of success, but rather try to become a man of value."

—Einstein, Albert

To my dear family

Abstract

Online social networks such as Twitter, Facebook and Sina Weibo are becoming increasingly important sources of information for its users. Misinformation such as rumors and fake news can spread quickly within social circles and may potentially incur massive losses to society. Therefore, it is often important to be able to accurately and promptly identify misinformation and their sources, so that proper control measures can be adopted. The research goal of this thesis is to develop methods estimating misinformation sources based on network topology and partial timestamps, and detecting misinformation based on appearance-level linguistic patterns.

In many practical applications, the network topology may not be known in advance and needs to be inferred. We first consider the problem of inferring network topology from information cascades and knowledge of some moments of the diffusion distribution across each edge, without needing to know the distribution itself. We first propose an iterative tree inference algorithm and then generalized the algorithm heuristically to general graphs. Simulations using synthetic and real networks, and experiments using real-world data suggest that our proposed algorithm performs better than some current state-of-the-art network reconstruction methods.

We then study the problem of identifying misinformation sources in a network based on the network topology, and a subset of infection timestamps. In the case of a single misinformation source in a tree network, we derive the maximum likelihood estimator of the source and the unknown diffusion parameters. We then introduce a new heuristic involving an optimization over a parameterized family of Gromov matrices to develop a single source estimation algorithm for general graphs. Compared with the breadth-first search tree heuristic commonly adopted in the literature, simulations demonstrate that our approach achieves better estimation accuracy than several other benchmark algorithms, even though these

require more information like the diffusion parameters. We next develop a multiple sources estimation algorithm for general graphs, which first partitions the graph into source candidate clusters, and then applies our single source estimation algorithm to each cluster. We show that if the graph is a tree, then each source candidate cluster contains at least one source. We also propose a sequential source estimation algorithm using a particle filter that is based on an approximate hidden Markov chain model, which can be interpreted as a “reverse” propagation process. This algorithm allows the source estimate to be updated quickly whenever a new timestamp is available. Simulations using synthetic and real networks, and experiments using real-world data suggest that our proposed algorithms are able to estimate the true misinformation source(s) to within a small number of hops with a small portion of the infection timestamps being observed.

Finally, we consider the problem of fake news detection based on statement and additional metadata. We develop a hybrid supervised learning framework combining a deep learning model and a graph model to predict the truthfulness label of a piece of news. We first train two deep neural networks, one to obtain a vector of likelihood scores for different truthfulness classes for any news, the other to show the relationship of truthfulness between any pair of news. A directed and weighted graph is then constructed to compute another vector of likelihood scores for different truthfulness classes. We finally combine two vectors from the first neural network and the graph model and obtain the label prediction by finding the truthfulness class with highest likelihood score. We apply our framework on the publicly LIAR dataset and obtained an improvement in prediction accuracy over the state-of-the-art.

Contents

Acknowledgements	ix
Abstract	xiii
List of Figures	xix
List of Tables	xxi
Symbols and Acronyms	xxiii
1 Introduction	1
1.1 Background and Motivations	1
1.2 Literature Review	2
1.2.1 Network Topology Inference	2
1.2.2 Misinformation Sources Estimation	4
1.2.3 Fake news Detection	6
1.3 Our Contributions	8
1.3.1 Network Topology Inference	8
1.3.2 Misinformation Sources Estimation	9
1.3.3 Fake news Detection	10
1.4 Thesis Outline	11
2 System Model and Preliminaries	13
2.1 System model and Problem Formulation	13
2.2 Graph Laplacian and Graph Smoothness	16
2.3 Machine Learning and NLP Techniques	17
2.3.1 Neural Network for Multi-label Classification	17
2.3.2 Word Embeddings	19
2.3.3 BERT Model	20
3 Network Topology Inference	23
3.1 Iterative Tree Inference Algorithm	23
3.2 The graph inference algorithm	26
3.3 Simulation results	30

3.3.1	Tree networks	31
3.3.2	General graphs	33
3.3.3	Distribution mismatch	36
3.3.4	Heterogeneous spreading distributions	36
3.3.5	Bimodal spreading distributions	38
3.3.6	Real dataset	39
3.4	Conclusion	40
4	Sequential Source Estimation	41
4.1	Sequential Source Estimation	41
4.1.1	Reverse Propagation Process	41
4.1.2	Particle Filtering	42
4.2	Experimental Evaluations	44
4.2.1	Synthetic Data	44
4.2.2	Weibo Data	49
4.3	Conclusion	50
5	Single Source Estimation	51
5.1	A Gromov Matrix Interpretation	51
5.2	Maximum Likelihood Estimator	54
5.3	Single Source Estimation for General Graphs	56
5.4	Discussion	60
5.5	Simulations for Single Source Estimation	61
5.6	Appendix	68
6	Multiple Sources Estimation	73
6.1	The Reduction Step for Trees	73
6.2	Multiple Sources Estimation for General Graphs	80
6.3	Experimental Evaluations for Multiple Sources Estimation	81
6.4	Conclusion	87
7	Fake News Detection	89
7.1	Proposed model	89
7.1.1	Feature Extraction	89
7.1.2	The Hybrid Learning Model	90
7.2	Performance Evaluation	94
7.2.1	Dataset	94
7.2.2	Experiment Settings and Results	95
7.3	Conclusion	100
8	Conclusion and Future Work	101
8.1	Conclusion	101
8.2	Future Work	102

List of Author's Awards, Patents, and Publications	105
Bibliography	107

List of Figures

2.1	Neural network architecture for multi-label classification.	18
2.2	CBOV model.	20
2.3	Skip Gram model.	20
2.4	Architecture of the MLM procedure when pre-training BERT. . . .	21
2.5	Apply BERT model to classification tasks.	22
3.1	This figure gives an example to explain why we use (3.2) in the transfer step.	26
3.2	This figure shows the performance of ITI for different parameters. .	31
3.3	Performance comparison between General-ITI and ITI.	32
3.4	Performance comparison between ITI, General-ITI and TR.	33
3.5	Performance comparison between the NetRate algorithm, Kernel-Cascade algorithm and the GI-algorithm	35
3.6	Performance comparison between NetRate, KernelCascade, GI and NetRate-GI	38
4.1	Sequential source estimation on ER(1000, 2).	46
4.2	Sequential source estimation on ER(1000, 4).	47
4.3	Comparison between SSE and GAU on ER(200, 2) for both Gaussian and exponential spreading.	48
4.4	Sequential source estimation on a Facebook subnetwork.	48
4.5	Evaluation of SSE on the Weibo data.	49
5.1	An example to illustrate the convex combination of Gromov matrices. .	59
5.2	Single source estimation on ER(500) and BA(500).	63
5.3	MSE of $\hat{\mu}$ and $\hat{\sigma}^2$ obtained by MLE and GSSI.	63
5.4	Single source estimation on ER(500, 4) and BA(500, 4).	64
5.5	Single source estimation on ER(500, 16) and BA(500, 16).	65
5.6	Single source estimation on two real networks.	66
5.7	Simulations to study how α_s varies with the fraction of observed timestamps and for different graphs.	66
5.8	Performance comparison for different algorithms.	67
5.9	Malware source estimation on Facebook network.	67
6.1	An example to illustrate the notions of observation cluster, source candidate cluster and the SCCE algorithm.	74

6.2	Two cases where we undereste the number of sources.	79
6.3	Multiple sources estimation on (a) BA(500) and (b) ER(500). . . .	83
6.4	Multiple sources estimation on (a) BA(500, 16) and (b) ER(500, 16). .	83
6.5	Multiple sources estimation on (a) Enron network and (b) Facebook network.	84
6.6	Performance of the average L for different graphs.	84
6.7	Malware sources estimation on Facebook network.	85
6.8	Performance of average Δ and average L for different $ S $ on B-A tree BA(500, 16) and B-A graph BA(500, 16).	85
7.1	An example to illustrate the label-mapping function.	91
7.2	Two neural networks used in our model.	91
7.3	Our hybrid fake news detection model.	94
7.4	Performance of our models on the LIAR dataset with corrupted labels.	100

List of Tables

3.1	Performance of the estimated average degree.	34
3.2	Computation time (in <i>seconds</i>) of GI, NetRate and KernelCascade for varying $ V_c / V $	36
3.3	Edge recovery rate comparison of three spreading distributions. . .	38
3.4	Comparison of edge recovery rate on the MemeTracker dataset. . .	40
4.1	Some graph properties of the networks used.	46
4.2	Standard deviation of the error distances on different networks. . .	50
5.1	Some graph properties of the networks used in our simulations. . . .	61
5.2	Parameter description and setting for malware propagation model. .	68
5.3	Percent of reduced average error distance GSSI compared with the next best algorithm.	68
6.1	Comparison between SCCE and BDSL on real data.	87
6.2	Percent of reduced average error distance of SCCE compared with BDSL.	87
7.1	An example to compute $P(l_j \mid l_i, r_{ij})$ where $l_i, l_j \in \mathcal{L}$ and $r_{ij} \in \mathcal{R}$ given the label-mapping function in Fig. 7.1.	93
7.2	The LIAR dataset statistics.	94
7.3	Two samples from the LIAR dataset.	95
7.4	Accuracy on LIAR dataset when only credit history is used in our NN1 model. Accuracies of other models in the literature are also given.	96
7.5	Hyperparameter settings for NN1 and NN2.	97
7.6	Performance comparison in prediction accuracy for fake news detection on LIAR dataset.	98

Symbols and Acronyms

Symbols

$\text{Unif}(A)$	the uniform distribution over the set A
$\mathcal{N}(\mu, \sigma^2)$	the Gaussian distribution with mean μ and variance σ^2
$\mathcal{TN}(\mu, \sigma^2)$	the truncated Gaussian distribution whose support is non-negative
$\text{Exp}(\lambda)$	the exponential distribution with rate λ
$\Gamma(\alpha, \beta)$	the Gamma distribution with shape α and rate β
$p(x \mid y)$	the conditional distribution of x given y
$\sim$	equality in distribution
$\mathbb{E}[\cdot]$	the expectation operator
$\mathbb{R}^n$	the n -dimensional Euclidean space
$[\mathbf{A}]_{i,j}$	the (i, j) -th entry of matrix $\mathbf{A}$
$\mathbf{A}'$	the transpose of $\mathbf{A}$
$\mathbf{A}^{-1}$	the inverse of $\mathbf{A}$
$\text{tr}(\mathbf{A})$	the trace of $\mathbf{A}$
$\det \mathbf{A}$	the determinant of $\mathbf{A}$
$O(\cdot)$	the order of magnitude or ergodic convergence rate (running average)
$\text{sign}(x)$	the sign of the real number x
$\ \cdot\ $	the 2-norm of a vector or matrix in Euclidean space

Acronyms

i.i.d.	independently and identically distributed
w.r.t.	with respect to
SI	Susceptible-Infected
SIR	Susceptible-Infected-Recovered

SIS	Susceptible-Infected-Susceptible
TRBS	Time-Reversal Backward Spreading
E-R	Erdős-Rényi
B-A	Barabási-Albert
ITI	Iterative Tree Inference
GI	Graph Inference
TR	Tree Reconstruction
EXP	Exponential distribution
POW	Power-law distribution
RAY	Rayleigh distribution
SSE	Sequential Source Estimation
HMM	Hidden Markov Model
MLE	Maximum Likelihood Estimator
BFS	Breadth-First Search
GSSI	Gromov Single Source Identification
MSR	Multiple Sources Reduction
SSSE	Source-Set Size Estimation
SCCE	Source Candidate Clustering and Estimation
NLP	Natural Language Processing
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
TFIDF	Term Frequency-Inverse Document Frequency
LIWC	Linguistic Inquiry and Word Count
MMFD	Multi-source Multi-class Fake news Detection

Chapter 1

Introduction

1.1 Background and Motivations

Online social networks such as Twitter, Facebook and Sina Weibo are becoming increasingly important sources of information for their users [1]. Rumors, fake news and other forms of misinformation can be shared and forwarded with high speed through friendship circles. Such misinformation are produced online for a variety of purposes including financial gains and political benefits, and may result in massive losses to both the individuals and the society [2–5]. For example, rumors can potentially mislead the public [6], triggering undesirable widespread panic [7]. Similarly, negative news on celebrities or organizations may damage their reputations or even impede their development. Fake news employed in election campaigns or politic-specific events can threaten public safety and break the authenticity balance of the news ecosystem [8]. In view of this, it is often important to be able to promptly and accurately identify the misinformation and their sources, so that proper control measures can be adopted. In this thesis, we consider the problem of identifying misinformation and their sources in social networks.

The propagation initiated by a misinformation source can be modeled using a diffusion or infection process in networks [9, 10]. The network topology and information of nodes observed on this network such as infection status and timestamps are often used to estimate the source. However, in many practical applications, the network topology may not be known in advance and needs to be inferred from other information collected from this network. Therefore, we first study the problem of

inferring network topology based on the knowledge of the time each node in the network receives a piece of information, when an information diffusion is initiated from a known source node. We then study the problem of estimating misinformation sources based on knowledge of the network topology, and a subset of infection timestamps. Identifying misinformation such as fake news is important yet challenging in the era of social media. We finally study the detection of fake news with different degrees of truthfulness based on textual statement and auxiliary metadata such as topic and speaker’s information.

1.2 Literature Review

This section discusses the related works, which are divided into three research topics including network topology inference, misinformation sources estimation and fake news detection.

1.2.1 Network Topology Inference

Complex networks, inspired by empirical research on real-world networks in nature and human societies, have been studied extensively in recent years [11, 12]. The theory of complex networks has found applications in diverse areas including theoretical physics, sociology and biology [13–16]. However, in many practical applications, the network topology may not be known in advance and needs to be inferred from the information observed on the network.

The network topology inference problem using information cascades has been investigated under various assumptions. For example, [17, 18] considered the inference of graphs with a continuous-time diffusion model. The diffusion along each edge is assumed to be exponential or satisfies a power law distribution. In [19], the authors assumed that the network can be undirected, and, along each edge, information spreads in two steps: a Bernoulli selection step and an exponential transmission step. The papers [20, 21] perform topology inference using likelihood maximization approaches. In all of these works, knowledge of the family of probability distributions that generates the information spreading along edges is

assumed, and many also assumed the spreading along different edges to be identically distributed. Theoretical guarantees and recovery conditions are provided in the references [19, 21, 22].

Similar research work also includes link prediction in social networks such as Twitter [23, 24]. Link prediction refers to inferring the future relationships between nodes in the complex network based on the observed network structure and node attributes. Therefore, it assumes the knowledge of existing network topology and does not use cascades as observations.

In practical applications, there are situations where it is not easy to know what the spreading distribution looks like *a priori*. For example, information diffusion in an online social network depends on various factors [16]. In some cases, *moments* – such as the average amount of time it takes for information to diffuse from a node to a neighbor node – can be estimated from historical data [25, 26], data collected from a related network, or based on domain experts’ opinions. In this thesis, we consider the case where the diffusion along different edges is independent, and the spreading distribution is unknown to an observer. However, certain moments of the distribution (instead of the full distribution) are known. Our assumptions are complementary to those made in [17–19, 21] in the following sense: while these previous works assume that the spreading distribution belongs to a known family of distributions with unknown parameters or moments, our work assumes that some parameters of the spreading distribution are known but not the family it belongs to. Either set of assumptions has its own merits and may be more suitable than the other under different situations. We also consider the case where our approach is combined with the NetRate algorithm proposed in [18]. We first make an assumption about the family of distributions and apply NetRate to estimate the moments. We then apply our proposed approach to infer the network topology. Our experiments in Section 3.3.4 indicate that the combined algorithm has the best performance. However, if the assumption about the family of distributions is wrong which leads to the problem of distribution mismatch, such strategy can fail. We apply different algorithms on the MemeTracker dataset in Section 3.3.6. We aim to recover connections between a collection of sites by analyzing their published posts. The groundtruth network topology is obtained from hyperlinks inside posts. However, such hyperlinks are not always available in practice. For example, some sites may post news or blogs without explicitly citing their sources. Thus, we only

observe the timestamps when they are posted. The moments of edge propagation delay can be estimated from historical data. Experiment results show that our approach performs better than those making use of distributions. Similarly, in virus propagation, we observe people getting sick without usually knowing who infected them. In a viral marketing setting, we observe people purchasing products or adopting particular behaviors without explicitly knowing who the influencer was. In such applications, our algorithms can be helpful since the estimation of moments is easier than distributions.

In [27], the authors proposed a kernel-based method that does not require the knowledge of the spreading distributions. They kernelize the transmission functions over network edges and then infer them from the data, which requires a large amount of data for accurate inference. Another drawback is that the performance is sensitive to the choice of hyperparameters and kernels, which need to be adjusted for different networks and amount of observed data. Experiments in Section 3.3 indicate that our proposed approach has a better performance.

Owing to the recent advancements in the field of graph signal processing (see [28] for an overview), several network topology inference methods using graph signals have been proposed (for example, [29–31]). These approaches are based on the idea that certain graph signals are closely related to the eigenvectors of graph-shift operators of the graph (e.g., graph adjacency matrix and graph Laplacian). However, in our setting, the timestamps are related to the distance from the nodes of a graph to a fixed source node. Such timestamps are usually not directly related to the above-mentioned shift-operators, but rather depend on the choice of the source node. Therefore, the graph signal processing approaches cannot be applied easily to our problem.

1.2.2 Misinformation Sources Estimation

Several works have tackled the problem of misinformation source estimation under different assumptions. Many of these works are based on the knowledge of network topology and the infection status of a portion of nodes in the network. In the literature, various spreading models have been proposed including the Susceptible-Infected (SI) model [32], Susceptible-Infected-Recovered (SIR) model [33], and Susceptible-Infected-Susceptible (SIS) model [34]. Specifically, in the SI model,

an infected node remains infected forever; in the SIR model, it can recover and cannot be further infected; and in the SIS model, a recovered node can become infected again. A rumor centrality estimator under the SI model was developed in [32, 35], while [36–39] developed estimators for identifying multiple infection sources under the SI model. The paper [40] considered infection source estimation when only a subset of infected nodes are observed. In [41], multiple observations of a SI spreading process are used for source estimation. In [33] and [42], the problem of infection source estimation is investigated under the SIR model, and [34] considers the SIS model. All the aforementioned works perform source estimation based on the observed status of the nodes and the network topology. Other related works assume additionally a priori knowledge like the infection spreading rate, including [43] which performed source inference via belief propagation, and [44] which developed a dynamic message passing algorithm. Recently, [45] considered an alternative independent cascade model for infection spreading, and proposed the short-fat tree (SFT) algorithm for source identification under the independent cascade model. The reference [46] developed a framework for optimal source estimation in arbitrary weighted networks with arbitrary distribution of sources by combining controllability theory and compressive sensing. The reference [47] considered the case where additional relative information about the infection timestamps of a fraction of node pairs is also available. Finally, the works [48, 49] studied the complementary problem of source obfuscation and proposed messaging protocols which can spread the infection as widely as possible while protecting the anonymity of the source.

Infection source estimation has also been investigated using infection timestamps. In this framework, we make use of the observations of the first infection timestamps of a subset of nodes in the network. In [50], an algorithm based on the maximum likelihood estimator (MLE) is proposed. Two ranking algorithms using a modified Breadth-First Search (BFS) tree heuristic for general graphs have been proposed in [51], which demonstrate improved estimation accuracy compared to the algorithm in [50]. In [52], a two-stage algorithm was proposed to locate a single source in large networks. The reference [53] proposed a sequential source estimation algorithm that allows online update of the source estimate as timestamps are observed sequentially. In [54], a Time-Reversal Backward Spreading (TRBS) algorithm was proposed to infer a single source in a weighted network. The papers [55, 56] discuss the selection of observer nodes under the deterministic slotted SI model, so as to achieve low

probability of error detection. The problem of multiple sources estimation with infection timestamps is less studied. In [57], a backward diffusion based method was proposed. However, the paper assumes that all infection sources initiate diffusion at the same start time, and the mean of the infection propagation delays along all edges and the number of sources are known. Such assumptions can be restrictive for practical applications.

1.2.3 Fake news Detection

Automatic fake news detection, which aims to assess the truthfulness of news, has become an increasingly important yet challenging topic in recent years. Individuals and society can be affected by the propagation of fake news. For example, people can be misguided and then make undesirable decisions, journalism can lose its integrity, and political turmoil can happen. Therefore, it is often significant to detect such fake news and prevent their negative impacts. On the other hand, it can be difficult to identify the truthfulness of a piece of news. One important reason is that a statement may be intentionally made by a particular entity for some social, financial or even political purposes. Hence, it is designed to contain lots of ambiguities and misleading information which can deceive the public. Although many theories and models have been developed to tackle such a fake news detection problem based on different knowledge, significant efforts are still needed to benefit the public and the news ecosystem.

Pioneering works such as [58, 59] have proposed feature-based approaches to solve the problem. Such methods achieved certain success but can be expensive and time-consuming. Knowledge-based approaches have also been proposed in [60–63]. However, these approaches are incapable of dealing with long sentences or unusual sentence structures which are typically found in online news [64].

Detecting fake news has recently attracted tremendous attention from the Natural Language Processing (NLP) community. Majority of the recent research uses machine learning models especially neural networks models such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Attention mechanism. Since we mainly deal with texts, some text preprocessing techniques in NLP such as pre-trained word embeddings, Term Frequency-Inverse Document Frequency (TFIDF) and Linguistic Inquiry and Word Count (LIWC) are frequently

used. The paper [65] presented a publicly benchmark dataset called LIAR for fake news detection. This dataset includes 12.8K manually labeled short statements from PolitiFact ¹. Different from most of the other datasets, there are six fine-grained labels for the truthfulness ratings in LIAR. To solve the problem, they proposed a hybrid CNN architecture and concatenated the max-pooled text features with the metadata features from the bi-directional LSTM. After that, some other models have been proposed and tested on the LIAR dataset. A Multi-source Multi-class Fake news Detection framework (MMFD) has been proposed in [66]. MMFD integrates automated feature extraction, multi-source fusion and fakeness discrimination into a coherent model. The model uses CNN to extract features from textual sources in a statement and use LSTM to analyze temporal dependencies in the entire text. The work [67] proposed a method incorporating speaker profiles into an attention-based LSTM model. Speaker profiles can be included as attention factors for the learning of news text or as additional input into the LSTM. They have shown the improvement of performance with attention mechanism. They also observed that a speaker’s credit history greatly contributes to a successful prediction. However, this information is not often available during predictions. In view of this, the work [68] exploited ways to incorporate a speaker’s credibility into fake news detection. A new resource Speaker2Credit which assigns speakers to a credibility vector is provided to make credibility information usable during prediction. They proposed a hybrid-LSTM model incorporating such credibility information and showed significant improvement in prediction accuracy. In [69], memory network which takes advantage of attention mechanisms has been applied to achieve better performance. The author used single layer memory network to learn a different representation of words by memorizing the set of words in the memory. The words in input sentences are then weighted by attention mechanism and related words can thus be extracted in memory. The model focus on the most relevant sub-parts of a given input and stores external information by the network itself.

LIAR is based on short claims. Similarly, a Fever dataset has been presented in [70]. This dataset contains 185,445 short claims and related evidences for fake news detection. The label is three-graded including *Supported*, *Refuted*, or *Not Enough Info*. Fever provides new possibilities to build a system which is able to predict

¹POLITIFACT.COM

the truthfulness of a claim together with the evidence. Related works include [71–74]. There are several datasets focusing on detecting fake news based on the whole article including FAKENEWSNET [64, 75, 76] and a dataset presented in [77]. For FAKENEWSNET, the label can be true or false and the state-of-the-art has achieved an accuracy greater than 0.9 [78]. In [77], the authors focus on analyzing linguistic characteristics of news. They adapted four types of lexical resources and tried to distinguish between truth news and fake news by examining their lexicons’ distributions. Similarly, the works [79, 80] also made use of linguistic characteristics. Some other datasets and related works can be found in survey papers [64, 81, 82].

1.3 Our Contributions

Overall, the thesis has the following threefold contributions.

1.3.1 Network Topology Inference

In this thesis, we consider the inference of a network topology based on the knowledge of the time each node in the network receives a piece of information, when an information diffusion is initiated from a known source node. Our basic goal is to develop low-complexity algorithms to infer graph topologies. Our main contributions are as follows:

- (i) We develop an iterative tree inference algorithm that makes use of information cascades and the average mean propagation time along each edge. Simulations suggest that our method outperforms the algorithm proposed in [19].
- (ii) We develop a heuristic general graph inference algorithm by extending our tree reconstruction method to make use of the higher order moments of the propagation time along each edge. Here, we assume that an estimate of the average node degree is available. Simulations suggest that our proposed approach has relatively low time complexity and good edge recovery rate compared to the current state-of-the-art methods in [18, 27].

1.3.2 Misinformation Sources Estimation

Most of the previous works perform source estimation based on a single snapshot observation of either the node status or infection timestamps, or on multiple snapshot observations of different infection cascades. In practical applications, observations of either a node status or an infection timestamp may arrive sequentially, and sometimes we do not know when the next observation will be available. In a sequential estimation setting, our goal is to develop algorithms that can estimate the source using all observations up to the current time, and to be able to quickly update our estimate when new observations are available. Therefore, we first propose a novel Sequential Source Estimation (SSE) algorithm that can estimate the infection source using a stream of infection timestamps arriving sequentially. To the best of our knowledge, this is the first sequential infection source estimation in the literature. Our proposed SSE algorithm is based on the construction of a “reverse” propagation process, which allows us to perform particle filtering on the sequential observations. Simulation results show that our algorithm can quickly and accurately update source estimation as timestamps arrive sequentially.

We then study the single source estimation problem with fewer assumptions compared with the works [50, 52, 53]. Similar to [50, 52, 53], we assume that the propagation delay along each edge can be modeled using a Gaussian distribution. However, unlike these references, the diffusion parameters, including the mean and variance of the infection propagation delay, are assumed to be unknown. When the graph is a tree, we derive the optimal MLE that simultaneously estimates the source and diffusion parameters. It is common in the literature [32, 50, 52, 83] to use a BFS tree heuristic to generalize an algorithm for identifying a single source in a tree to one for a general graph: for each node in the general graph, one constructs a BFS tree and assumes that any infection starting at this node diffuses along this BFS tree. In this thesis, we introduce a novel heuristic, which involves finding an optimal convex combination of two BFS trees and a target covariance matrix (cf. Section 5.3) for each node to better approximate the diffusion process, before generalizing our algorithm for trees to general graphs. We call this the Gromov Single Source Identification (GSSI) algorithm. Simulation results demonstrate that GSSI performs better than algorithms proposed in [50, 54] although GSSI requires less information.

We also study the case of multiple sources and propose a Source Candidate Clustering and Estimation (SCCE) algorithm, which makes fewer assumptions compared to the work [57] as we do not assume that the infection sources have the same diffusion parameters, and the number of sources is also unknown a priori. SCCE can be divided into two steps. In the first step, we reduce the multiple sources estimation problem to single source estimation problem by partitioning the graph into source candidate clusters. We show that if the graph is a tree, then each cluster contains at least one source. We then apply a procedure to estimate the number of sources in each cluster and the infection region of each source. In the second step, we apply the GSSI algorithm in each region. Simulation results demonstrate that our approach performs better than the algorithm proposed in [57], and is able to accurately infer multiple sources with a small portion of timestamps.

1.3.3 Fake news Detection

We study the problem of fake news detection based on statement and additional metadata. We intend to assess the truthfulness of a piece of news according to its own content as well as the information from historical or related news. We encode textual statement using a pre-trained and then fine-tuned BERT language representation model. The full feature vector is obtained by integrating metadata features with statement feature. We develop a hybrid supervised learning framework combining a deep learning model and a graph model to predict the truthfulness label of a piece of news. We first train two deep neural networks: one to obtain a vector of likelihood scores for different truthfulness classes for any news, and the other to show the relationship of truthfulness between any pair of news. A directed and weighted graph is then constructed to compute another vector of likelihood scores for different truthfulness classes. We finally combine two vectors from the first neural network and the graph model and obtain the label prediction by finding the truthfulness class with the highest likelihood score. We apply our framework on the publicly LIAR dataset and obtained an improvement in prediction accuracy over the state-of-the-art. We also make use of the techniques in graph signal processing to interpret our graph model and introduce an optimization step to further improve the performance.

1.4 Thesis Outline

In Chapter 2, we introduce the overall system model, problem formulation, and assumptions for the whole thesis. We also present some related concepts and techniques.

In Chapter 3, we consider the inference of a network topology based on information cascades. We first develop an iterative tree inference algorithm and then extend the algorithm to general graphs. We finally present simulation results and experiments on real data.

In Chapter 4, we study the sequential source estimation problem. We first construct a reverse infection propagation process and then present our SSE algorithm. Simulation results on both synthetic and real-world data are then presented.

In Chapter 5, we consider the problem of single source estimation. We provide an interpretation based on Gromov matrices. When the network is a tree, we first derive the MLE for estimating a single source. We then present our GSSI algorithm using Gromov matrices for identifying a source in a general graph. Simulations on both synthetic and real-world networks are also provided to compare the performance of GSSI with various other algorithms in the literature.

In Chapter 6, we investigate the problem of multiple sources estimation, present our SCCE algorithm, and provide simulation results to verify its performance.

In Chapter 7, we consider the problem of fake news detection. We develop a hybrid supervised learning framework. We also present experiment results on the real-world data and show the improvement compared to previous models.

Chapter 8 concludes the thesis and envisions some future work.

Notations: We use $\text{Unif}(A)$ to denote the uniform distribution over the set A . We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and variance σ^2 , and $\mathcal{TN}(\mu, \sigma^2)$ to denote the corresponding truncated Gaussian distribution in which the distribution support is constrained to be non-negative. We use $\text{Exp}(\lambda)$ to denote the exponential distribution with rate λ , and $\Gamma(\alpha, \beta)$ to be the Gamma distribution with shape parameter α and rate parameter β . For a set X , its cardinality is given by $|X|$. Given a matrix $\mathbf{A}$ (or A), we use $[\mathbf{A}]_{i,j}$ or $\mathbf{A}(i, j)$ to denote its (i, j) -th entry. We write $\mathbf{A}'$ for the transpose of $\mathbf{A}$, $\mathbf{A}^{-1}$ for its inverse, $\text{tr}(\mathbf{A})$ for

its trace, and $\det \mathbf{A}$ for its determinant. $\mathbb{E}[\cdot]$ is the expectation operator. $\text{sign}(x)$ is the sign of the real number x . The symbol $\sim$ means equality in distribution. We say that $f(N) = O(g(N))$ if $f(N)/g(N) \leq k$ for some fixed $k > 0$ as $N \rightarrow \infty$. For two nodes u and v in a graph G , we let $d(u, v)$ denote the length of a shortest path between nodes u and v , $[u, v]$ denote any shortest path from u to v , $(u, v]$ the path with the end node u excluded, and so on. For each node v , let $d_v(u) = d(v, u)$ denote the distance function (from any other node $u \in G$) to v , and ∂v denote the set of neighbors of node v .

Chapter 2

System Model and Preliminaries

In this chapter, we introduce the overall system model, problem formulation, and assumptions for the whole thesis. We also present some related concepts and techniques which can be used to solve our problems.

2.1 System model and Problem Formulation

We model the network as a connected and unweighted simple graph (i.e., undirected graph containing no graph loops or multiple edges) $G = (V, E)$, with V the set of nodes and E the set of edges. Suppose a collection of misinformation sources $S = \{s_1, \dots, s_{|S|}\} \subset V$ initiate a diffusion process on G at unknown start times. If a node $v_i \in G$ first receives the misinformation at time t_i , we say that v_i is *infected* and its *infection timestamp* is t_i . The misinformation propagates from an infected node to its neighbors stochastically along the edges connecting them. Similar to [50, 52, 53], we assume that the propagation delays $\{\tau_{uv}\}$ for all pairs of adjacent nodes u, v are independently and identically distributed (i.i.d.) continuous random variables that follow a distribution with probability density function $f_\tau(\cdot)$. We assume the diffusion follows the SI model: any infected node never recovers. Suppose that we observe a collection of infected nodes $\mathcal{V} = \{v_1, \dots, v_n\}$, which is a subset of all the infected nodes, and the vector of their associated infection timestamps $\mathbf{T} = [t_1, \dots, t_n]'$. We also assume $S \cap \mathcal{V} = \emptyset$ for simplicity. Our goal is to estimate S using $(\mathcal{V}, \mathbf{T})$.

In Chapter 5 and Chapter 6, we will respectively consider the case of single misinformation source, i.e., $|S|=1$ and multiple sources where $|S|>1$ but unknown. We assume that $\{\tau_{uv}\}$ follow a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$.¹ However, we also assume that the parameters (μ, σ^2) are unknown, and need to be estimated.

In Chapter 4, we study the problem of sequential single source estimation. Suppose that we observe the infection timestamps (v_n, t_n) , $n \geq 1$ sequentially. We do not assume that the infection timestamps $\{t_n\}$ are ordered in any way as a node that was infected earlier could be observed only later. We also assume $f_\tau(\cdot)$ is known. Let $\mathcal{O}_n = \{(v_k, t_k) : k \leq n\}$ be the set of observations up to the n -th timestamp, and let

$$s_1^n = s_1^n(\mathcal{O}_n) \quad (2.1)$$

be the misinformation source estimate based on $\mathcal{O}_n$. Our aim is to design s_1^n so that s_1^{n+1} can be updated from it easily, and s_1^n is in some sense close to the true source s_1 for n sufficiently large.

In Chapter 3, we consider the inference of a network using a collection of information cascades. A *cascade* consists of a single source u that initiates an information diffusion together with a set of infection timestamps $\{T_u(v) : v \in V\}$. We assume the start diffusion time of u is 0 so that $T_u(v)$ is the amount of time it takes information to propagate from node u to node v . For $\{\tau_{uv}\}$, we relax the i.i.d. assumption but assume that they are independent, and the average of some moments of $\{\tau_{uv}\}$ are known *a priori*. For each edge $(u, v) \in E$ and $k \geq 1$, let the k -th moment of τ_{uv} be $\mu_{u,v}^{(k)}$. We require to know $\mu^{(k)}$, the average of $\mu_{u,v}^{(k)}$ over the edges $(u, v) \in E$, for $k = 1$ if G is a tree, and for $k = 1, 2$ if G is a general graph. This assumption is valid in the following application scenarios:

- (a) Historical data is available for us to compute the empirical moments for information propagation between nodes in a network. Such data may be available only for specific nodes in the network whose neighbors are known, or for nodes in a different network that shares similar characteristics as the network of interest. In this case, we simply take $\mu^{(k)}$ to be the historical empirical moment. In simulations in Section 3.3.4 using the mean and variance

¹The Gaussian distribution is adopted for technical convenience although its realizations are not necessarily positive. In our simulations, we will generate the propagation delays using a truncated Gaussian distribution instead.

of the propagation time, we demonstrate that this approach can still produce reasonable results even if the actual mean and variance for each edge is unknown and heterogeneous over the edges. A heuristic basis for this observation is as follows: Consider the case where G is a tree, and suppose that the mean propagation time $\mu_{u,v}^{(1)}$ along any edge $(u, v) \in E$ is independently and identically distributed (i.i.d.) with mean μ . For any pair of nodes u and v , there is a unique simple path connecting them. Suppose it consists of d edges $e_1, \dots, e_d$. Let t_{e_i} denote the time it takes to pass the infection along e_i . Considering the cascade initiated from u , we have

$$\begin{aligned} \mathbb{E} [T_u(v) \mid (\mu_{e_i}^{(1)})_{i=1}^d] &= \sum_{1 \leq i \leq d} \mathbb{E} [t_{e_i} \mid \mu_{e_i}^{(1)}] \\ &= \sum_{1 \leq i \leq d} \mu_{e_i}^{(1)}. \end{aligned} \quad (2.2)$$

Applying the strong law of large numbers [84] to the sum (2.2), $\mathbb{E} [T_u(v) \mid (\mu_{e_i}^{(1)})_{i=1}^d] / d \rightarrow \mu$ almost surely as $d \rightarrow \infty$. If we adopt the stochastic approximation that the observed $T_u(v) \approx \mathbb{E} [T_u(v) \mid (\mu_{e_i}^{(1)})_{i=1}^d] \approx \mu d$ for nodes far apart, then using the common mean μ allows us to estimate the distance d between the nodes. We provide an example in Section 3.3.6, where we test our algorithm on a real-word dataset with the empirical mean and second order moment of the information propagation computed from historical data.

- (b) We estimate the propagation time moments by first running a separate statistical estimation procedure like the NetRate algorithm proposed in [18]. We then use the estimated propagation time moments for our proposed approach. In simulations in Section 3.3.4, we first run NetRate and then fuse our results with those produced by NetRate. We observe that this yields significant improvements over NetRate, even though the additional computation overhead incurred by running our method is negligible compared to NetRate.

We assume that each cascade persists long enough to infect all the nodes, and cascades are initiated from nodes in a subset $V_c \subset V$. The subset V_c is called the *source set*. The timestamp information $\{T_u(v) \in \mathbb{R} : u \in V_c, v \in V\}$ is recorded. Our goal is to infer the adjacency matrix A_G of G from the timestamp information.

In Chapter 7, we consider a special case of misinformation identification: automatic fake news detection. Instead making use of knowledge of the network topology or

information from the diffusion process, we aim to predict the truthfulness of a piece of news based only on its content and related metadata. We adapt techniques in machine learning and Natural Language Processing (NLP) which require a training set for a supervised learning framework. Specifically, suppose that we are given a training set of news or statements $\{S_1, S_2, \dots, S_N\}$ and associated truthfulness labels $\{l_1, l_2, \dots, l_N\}$. Each S_i can represent a piece of online news, a blog or a tweet which consists of the main statement and additional metadata such as topic and speaker's information. Each S_i has a unique truthfulness label $l_i \in \mathcal{L}$ where $\mathcal{L} = \{1, 2, \dots, L\}$ is the set of truthfulness classes with $L = |\mathcal{L}|$. For example, if a piece of news can be classified as false, half true or true. then $L = 3$. Our basic goal is to learn a model so that given a new S_k , we predict its label $\hat{l}_k \in \mathcal{L}$.

In the sequel, we present some useful concepts and techniques in graph theory and machine learning.

2.2 Graph Laplacian and Graph Smoothness

The problem of fake news detection does not involve a contact network any more. However, we propose to construct a weighted graph where each edge represents the relation between two different news and apply techniques such as graph signal and graph smoothness to help solve the problem. We need the following concepts.

Let $G = (V, E)$ be a weighted, undirected graph. The graph and edge weights can be encoded in the weighted adjacency matrix $\mathbf{A}$ of G : $w_{ij} = \mathbf{A}(i, j) > 0$ is the weight of the edge connecting nodes v_i and v_j . We also define the degree matrix $\mathbf{D}$ of G as: $\mathbf{D}(i, i) = \sum_j w_{ij}$ and $\mathbf{D}(i, j) = 0$ for $i \neq j$. The *graph Laplacian* is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The graph Laplacian and its eigenvalues contains geometric information of the graph, such as connectivity and sparsity [85].

A *graph signal* is a function defined on the nodes $f : V \rightarrow \mathbb{R}$. Signal values can be arranged in a vector form

$$\mathbf{f} = [f(v_1), \dots, f(v_N)]'.$$

The value of w_{ij} represents the relationship between signal components $f(v_i)$ and $f(v_j)$. A large weight indicates that we expect the signal values to be similar. The

smoothness of a graph signal $\mathbf{f}$ on G is then defined as:

$$\begin{aligned} S(G, f) &= \mathbf{f}'\mathbf{L}\mathbf{f} \\ &= \sum_{(v_i, v_j) \in E} w_{ij} (f(v_i) - f(v_j))^2. \end{aligned} \quad (2.3)$$

If the the quadratic form in (2.3) is smaller, for example, when strongly connected vertices (the edge weight between them is large) have similar signal values, then we say that the graph signal is smoother. It has been shown that smooth graph signal models can be used to solve many learning problems such as regularization and semi-supervised learning on graph [86, 87].

However, if we allow negative edge weights, we can generalize the above notion of smoothness of a graph signal as follows:

$$S(G, f) = \sum_{(v_i, v_j) \in E} |w_{ij}| (f(v_i) - \text{sign}(w_{ij})f(v_j))^2. \quad (2.4)$$

2.3 Machine Learning and NLP Techniques

In this section, we briefly present some machine learning and NLP models used for our fake news detection problem.

2.3.1 Neural Network for Multi-label Classification

Inspired by biological neuron systems, neural networks are algorithms that try to mimic the brain and the way it functions. Neural networks use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. See Fig. 2.1 for an example of the neural network architecture for multi-label classification. The input layer is a numerical feature vector x , then in the hidden layer we have:

$$h_W(x) = g(W'x)$$

where W is a weight matrix whose dimension is decided by the number of neurons between two successive layers and $g(\cdot)$ is a nonlinear activation function such as

FIG. 2.1: An example of the neural network architecture for multi-label classification.

Sigmoid, Tanh and ReLU. There may be more than one hidden layers and their choices of neuron numbers and activation functions can be different. When reaching to the output layer, a softmax function $\sigma(\cdot)$ can be used to normalize the output $y \in \mathbb{R}^K$ into a probability distribution consisting of K probabilities:

$$\sigma(y)_k = \frac{e^{y_k}}{\sum_{k=1}^K e^{y_k}}$$

We can then maximize such probabilities to obtain the label prediction.

To obtain such a neural network we need a collection of inputs and corresponding groundtruth outputs as the training set. We first compute a loss function which denotes the difference between predicted output y from the model and the ground truth output y^* . There are two main types of loss functions to apply, cross-entropy and mean squared error. For multi-label classification, after applying softmax function, cross entropy can be computed as:

$$C = \sum_{k=1}^K y_k^* \log(y_k)$$

Finally we minimize this loss and learn weights in neural network using backpropagation algorithms such as Stochastic Gradient Descent (SGD) method.

In this chapter we mainly use neural networks in Fig. 2.1. Other widely used architectures include Convolutional Neural Network (CNN), Recurrent Neural Network

(RNN), Long Short-Term memory Network (LSTM), Attention Mechanism, etc. We refer the reader to [88, 89] for more details.

2.3.2 Word Embeddings

Word embeddings, as one of the most commonly used representations of document vocabulary, map words that co-occur in a similar context to nearby points in the embedding space [90]. They are vector representations of a particular word that can capture word context in a document, syntactic and semantic information and relationship with other words. Therefore, they form the first layer of an architecture in a lot of NLP tasks. The most popular techniques for this purpose include Word2Vec [90], GloVe [91] and fastText [92]. All of them train models over large unlabeled corpus and generate pre-trained embeddings which have a single vector representation for each token.

In this chapter, we use Word2Vec. Word2Vec can be obtained using Skip Gram and Common Bag Of Words (CBOW) based on shallow neural network. See Fig. 2.2 for the CBOW model. It takes the context of a target word as the input and tries to predict the target word corresponding to the context. In the input layer, each context word is a one hot encoded vector of size V . We average over C context words to calculate hidden layer inputs. The hidden layer contains N neurons without activation function. The output is again a V length vector with softmax function applied. After training the neural network using backpropagation, the weight matrix $\mathbf{W}'_{N \times V}$ is taken as vector representations of words.

Skip Gram model follows similar topology as of CBOW but flips CBOW's architecture on its head. It takes the target word as the input and tries to predict the context. See Fig. 2.3 for the architecture. Both methods have their own advantages and disadvantages. CBOW is faster and has better representations for more frequent words. On the other hand, Skip Gram works better with small amount of training data and is found to represent rare words well. There are some pre-trained models of Word2Vec, for example, Google's pre-trained 300-dimensional Word2Vec embeddings [93] from Google News.

FIG. 2.2: CBOW model.

FIG. 2.3: Skip Gram model.

2.3.3 BERT Model

BERT, standing for Bidirectional Encoder Representations from Transformers, is a new language representation model from Google AI [94]. This simple but powerful model have presented state-of-the-art results in a broad set of NLP tasks, including natural language inference, question answering, named entity recognition, and so on.

There are also some other language representation models including ELMo [95] and GPT from OpenAI [96]. These works apply pre-trained language representations to down-stream tasks using strategies such as features-based (like ELMo) or fine-tuning (like GPT). However, both Elmo and GPT use unidirectional language models, which limits the choice of architectures used during pre-training and can fail when working on some token-level tasks such as sentence tagging and question answering. BERT model is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT representations can be fine-tuned with just one additional output layer to create models for a wide range of tasks without substantial task-specific architecture modifications.

FIG. 2.4: Architecture of the MLM procedure when pre-training BERT.

Specifically, the BERT architecture is a multi-layer bidirectional Transformer encoder. Transformer model proposed in [89] uses attention mechanism that learns contextual relations between words in a text. The input of BERT is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of hidden size, in which each vector corresponds to an input token with the same index. There is a base model with 12 layers and hidden size 768 and a large model with 24 layers and hidden size 1024. The input can be a single text sentence or a pair of sentences. For a given token, the input representation is obtained by summing the token, segment and position embedding. BERT is pre-trained using two novel unsupervised prediction tasks: Masked Language Model (MLM) and next sentence prediction. MLM allows the representation to fuse the left and the right context which allows to pre-train a deep bidirectional Transformer. See Fig. 2.4 for the architecture of the MLM procedure when pre-training BERT.

Applying BERT on a specific NLP task is relatively straightforward, we only need to incorporate the model with one additional output layer. Therefore, we actually learn a minimal number of parameters from scratch. For classification tasks such as sentiment analysis, we simply add a classification layer on top of the Transformer output for the first token (corresponds to the first input token [CLS], which is the special classification embedding). See Fig. 2.5 for a visualization. For token-level tasks such as sentence tagging and question answering, more details can be found in [94]. In the fine-tuning training, most parameters stay the same as in

(a) Sentence pair classification tasks. (b) Single sentence classification tasks.

FIG. 2.5: Apply BERT model to classification tasks.

BERT training. The paper also gives specific guidance (Section 3.5) on the hyper-parameters that require tuning.

Chapter 3

Network Topology Inference

In this chapter, we consider the inference of a network topology based on information cascades. We first develop our approach for the case where the network is a tree, and then extend to the general graph case.

3.1 Iterative Tree Inference Algorithm

Before presenting our approach, we refer the reader to paper [97] Section III and V for some important concepts and theories. We develop the underlying theory and procedure for the case where only distance information is available. Specifically, we introduce the concept of a separating vertex set for a graph, which is a set of nodes in which for any two given distinct nodes of the graph, there exists a node whose distance to them are different. We show that a necessary condition for reconstructing a tree perfectly using distance information between pairs of nodes is given by the size of an observed separating vertex set. We also introduce the concept of redundant vertices, which is used to develop our tree inference algorithm. We skip such technical results, the interested reader can refer to paper [97] for more details and intuitions. We will also include more intuitions in the following sections so that it does not affect the logic flow. Recall that for a cascade starting at node u , we use $T_u(v)$ to denote the first time that node v receives the information of the cascade. Recall also that $V_c = \{v_1, \dots, v_l\}$ is the set of nodes containing the sources of all the cascades. If there are several cascades with the same source, we average their infection timestamps. The timestamp information is therefore

$\{T_{v_i}(\cdot) : v_i \in V_c\}$. From our discussion in the previous sections, in order to use infection-time information associated with cascades from various sources effectively, we propose the following general scheme if G is a tree. More details are provided in the following discussion.

- (1) **Selection step:** Given $V_c = \{v_1, \dots, v_l\}$, find pairs of nodes (v_i, v_j) having a “high chance” of being connected by a single edge.
- (2) **Transfer step:** For a pair of nodes (v_i, v_j) connected by an edge, if $v_i, v_j \in V_c$, i.e., cascades at both v_i and v_j exist, we use the cascade at v_i to construct a new cascade at v_j , and average the infection timestamps with those from the existing cascade at v_j . This is discussed in more detail below. This is where we use the concept of redundant vertices.
- (3) **Reconstruction step:** Use the new timestamp information obtained in the previous steps to estimate $d_{v_i}(\cdot)$ for each $v_i \in V_c$, and reconstruct the graph G .

We first discuss the **reconstruction step**. To determine if u, v are connected by an edge, we want to compare $|d_{v_i}(u) - d_{v_i}(v)|$ with 1 for each $v_i \in V_c$. This motivates us to introduce the following weight for each pair u, v :

$$W(u, v) \triangleq \frac{\sum_{v_i \in V_c} |T_{v_i}(u) - T_{v_i}(v)| / \mu^{(1)} - 1|}{|V_c|}, \quad (3.1)$$

where we recall that $\mu^{(1)}$ is the average mean propagation time across each edge. Here, we take the average difference with 1.

If $W(u, v)$ is small, it suggests a higher chance that u, v are connected by a direct edge. We call W the *weight matrix*, and its entries the *weights*. If we want to infer the structure of a tree of size n , we can select $n - 1$ (total number of edges) pairs of distinct nodes (u, v) such that $W(u, v) = W(v, u)$ are the smallest $n - 1$ weights.

Recall that we have assumed that each cascade persists long enough to infect all the nodes for theoretical convenience. In the algorithm application, this assumption can be relaxed. Given u, v and $v_i \in V_c$, if $T_{v_i}(u)$ or $T_{v_i}(v)$ does not exist, we skip this cascade when computing $W(u, v)$. Therefore, in (3.1) for each pair of (u, v) , we replace V_c with $V_c(u, v) = \{v_i \in V_c : T_{v_i}(v) \neq \emptyset \text{ and } T_{v_i}(u) \neq \emptyset\}$. It is easy to see

that the variance of $W(u, v)$ is greater with smaller $|V_c(u, v)|$, making it harder to judge whether an edge exists between u and v . In applications, if $|V_c(u, v)| \ll |V_c|$ for some (u, v) , we recommend to ignore such $W(u, v)$.

Let the estimated graph be $G' = (V, E')$. We can directly use E' for the **selection step**. An alternative is to use the weight matrix W for the job. More precisely, we can set a numerical condition $\mathcal{C}$ based on W , and select a pair (u, v) as long as $W(u, v) = W(v, u)$ satisfies the pre-set condition $\mathcal{C}$. For example, for a fixed parameter k , we can choose kn pairs of (u, v) with the smallest $W(u, v)$ value. This reflects our belief that each selected (u, v) has a “high chance” of being connected by an edge.

If the cascades originate from the same source, then averaging the values of the propagation delays allows to filter out the noise affecting their values and get better estimates of the true distances. When cascades do not originate from the same source, such averaging is not directly possible. To bypass this problem, we leverage the multiplicity of the cascades by merging it in a so-called **transfer step**.

For a selected pair of nodes (v_i, v_j) with $v_i, v_j \in V_c$, and $u \in V$, we construct the times $T_{v_j}^{v_i}(\cdot)$ at v_j using T_{v_i} in the transfer step as follows: for each $u \in V$,

$$T_{v_j}^{v_i}(u) = \arg \min_{x \in \{T_{v_i}(u) + T_{v_i}(v_j), T_{v_i}(u) - T_{v_i}(v_j)\}} |x - T_{v_j}(u)|. \quad (3.2)$$

We now update $T_{v_j}(\cdot)$ by its average with $T_{v_j}^{v_i}(\cdot)$. Once $T_{v_j}(\cdot)$ are updated for every $v_j \in V_c$, we repeat the reconstruction step.

In (3.2) of the transfer step, we consider a cascade $\{T_{v_i}(\cdot)\}$ from v_i and attempt to reinterpret the time information as an observed cascade from v_j instead. If the observed cascade was initiated by v_j , the term $T_{v_i}(u) + T_{v_i}(v_j)$ in (3.2) is the amount of time information takes to propagate from v_j to u if $v_i \in [v_j, u]$, while $T_{v_i}(u) - T_{v_i}(v_j)$ applies if $v_j \in [v_i, u]$. To determine which of these two cases is more likely, we compare them to the observed $T_{v_j}(u)$. An example is shown in Fig. 3.1 for illustration.

In the transfer step, note that no additional observations other than the given $\{T_{v_i}(\cdot) : v_i \in V_c\}$ are used in our inference procedure. The transfer step is merely an averaging mechanism over the given timestamp information $\{T_{v_i}(\cdot) : v_i \in V_c\}$ that allows us to average out the noise in $T_{v_j}(\cdot)$ for some $v_j \in V_c$. The intuition

FIG. 3.1: This figure gives an example to explain why we use (3.2) in the transfer step. There are 5 nodes in the tree. We have a cascade initiated by v_2 , with the red numbers being $\{T_{v_2}(\cdot)\}$. Now we want to construct $T_{v_3}^{v_2}(\cdot)$. For nodes v_1 and v_2 , let $T_{v_3}^{v_2}(\cdot) = T_{v_2}(\cdot) + T_{v_2}(v_3)$. For nodes v_3 , v_4 and v_5 , let $T_{v_3}^{v_2}(\cdot) = T_{v_2}(\cdot) - T_{v_2}(v_3)$. Then for $T_{v_2}(\cdot)$ and $T_{v_3}^{v_2}(\cdot)$, the propagation times along all edges are all the same, but they are initiated by different sources. However, if we do not know the graph topology in advance, then for each node v_j , whether to choose $T_{v_2}(v_j) - T_{v_2}(v_3)$ or $T_{v_2}(v_j) + T_{v_2}(v_3)$ is unknown *a priori*. We propose to compare with a cascade initiated by v_3 .

is that if two nodes v_i and v_j are connected by an edge in a tree, then the cascades generated by them are mutually “transferable”. We modify $T_{v_i}(\cdot)$ to obtain $T_{v_j}^{v_i}(\cdot)$ using (3.2) and regard $T_{v_j}^{v_i}(\cdot)$ as another cascade “generated” by v_j . An average with $T_{v_j}(\cdot)$ is then taken, thus reducing the randomness inherent in the timestamps. Simulation results in Section 3.3.1 Fig. 3.2(b) show that the performance can be greatly improved if we perform additional transfer steps.

We summarize the above discussion as our Iterative Tree Inference (ITI) algorithm in Algorithm 3.1. The overall time complexity of ITI is $O(I|V_c||V|^2)$.

3.2 The graph inference algorithm

In this section, we consider general simple graphs. We first provided some theoretical insights into how the moments of the propagation time between two nodes in a general graph behave, which lend support to our heuristic extension of the ITI algorithm. We then propose a general graph inference algorithm.

We summarise the theoretical discussion in paper [97] Section VI-A and VI-B in the following dichotomy: when the graph is sparse (as measured by the edge to node ratio, for example) or the distributions of the propagation times along each edge have small variance, it is enough to use the mean of the distributions as in the

Algorithm 3.1 Iterative tree inference (ITI) algorithm

-
- 1: **Input** I : number of transfer steps, m_s : number of edges selected for the selection step, n : number of nodes, V_c : set of cascade sources, $\{T_{v_i}(\cdot) : v_i \in V_c\}$: timestamp information, $\mu^{(1)}$: average mean propagation time.
 - 2: Initialize $\eta_v = 0$ for each $v \in V_c$.
 - 3: **for** $s = 1, \dots, I$ **do**
 - 4: Compute weight matrix W in (3.1) and select m_s pairs $\{(v_i, v_j)\}$ having the m_s smallest $W(v_i, v_j)$ weights.
 - 5: **for** Each pair of (v_i, v_j) selected **do**
 - 6: Construct $T_{v_i}^{v_j}(\cdot)$ and $T_{v_j}^{v_i}(\cdot)$ using (3.2).
 - 7: Update $\eta_v = \eta_v + 1$ for $v = v_i, v_j$.
 - 8: Update $T_{v_i}(\cdot) = (\eta_{v_i} T_{v_i}(\cdot) + T_{v_i}^{v_j}(\cdot))/(\eta_{v_i} + 1)$ and $T_{v_j}(\cdot) = (\eta_{v_j} T_{v_j}(\cdot) + T_{v_j}^{v_i}(\cdot))/(\eta_{v_j} + 1)$.
 - 9: **end for**
 - 10: **end for**
 - 11: Select $n - 1$ edges $\{(u, v)\}$ corresponding to the $n - 1$ smallest $W(u, v)$ weights.
 - 12: **Output** $\{(u, v)\}$.
-

case of trees. On the other hand, if the graph is highly connected, the existence of an edge between two nodes u and v can make the mean propagation time between u and v small relative to $\mu^{(1)}$. Therefore, it is instructive to use the length of the propagation time between u and v to decide if they are connected by an edge or not. The same consideration applies to other moments (as compared against unbiased sample moments); and they can be used as additional criteria to decide the existence of edges.

In a general graph, it might not be known whether the connection between two nodes u and v is dense or sparse. One way to overcome such a difficulty is to compute the difference between the sample mean (respectively, sample moments) with both the theoretical mean (respectively, theoretical moments) as well as 0. Once these two values are obtained, it is enough to take the smaller one. Hence the weight matrix being used in ITI should be modified based on available moment information. Suppose that the average mean $\mu^{(1)}$ and average second order moment $\mu^{(2)}$ of the propagation time along each edge are known. We define the following:

$$W_1(u, v) = \frac{1}{|V_c|} \min \left\{ \sum_{v_i \in V_c} |T_{v_i}(u) - T_{v_i}(v)|, \sum_{v_i \in V_c} ||T_{v_i}(u) - T_{v_i}(v)| - \mu^{(1)}| \right\}, \quad (3.3)$$

$$W_2(u, v) = \frac{1}{|V_c|} \min \left\{ \sum_{v_i \in V_c} |T_{v_i}(u) - T_{v_i}(v)|^2, \sum_{v_i \in V_c} ||T_{v_i}(u) - T_{v_i}(v)|^2 - \mu^{(2)}| \right\} \quad (3.4)$$

and

$$W(u, v) = W_1(u, v) + W_2(u, v). \quad (3.5)$$

We then choose $n \cdot \deg_{\text{ave}}/2$ edges with the smallest $W(u, v)$ values to form the estimated graph, where $\deg_{\text{ave}}$ is an estimate of the average degree. In the case of general graphs, we assume that we have some prior knowledge of the underlying network, so that a reasonable estimation of the average degree can be performed (similar to [17]). There are quite a few important occasions that we can do so, and we list a few of them as follows:

- (i) We know how the graph is generated, or the distribution that governs the generation of the graph. For example, the graph generated according to the Erdős-Rényi (E-R) graph [98] has an expected degree for each node. If the network is modeled using the E-R graph, we can regard the expected degree as the average degree of the graph. Another example is the Forest-fire model [99]. If the forward and backward burning probabilities are available, we can obtain an estimate of the average degree.
- (ii) There are a small percent of nodes whose degrees are available. For example, in a social network, we can take a survey to learn the number of friends of some users and estimate the average degree $\deg_{\text{ave}}$ through sampling.
- (iii) There are situations where *extreme value theory* [100] can be applied and a direct estimation of the average degree from the cascades is possible. As a typical example, suppose that the propagation times along all edges are i.i.d. with exponential distribution having mean $\mu^{(1)}$. Consider a cascade with source u whose degree is $\deg(u)$, define

$$t_{\min}(u) = \min_{v \in V \setminus u} T_u(v).$$

It is easy to verify that $t_{\min}(u)$ follows the exponential distribution with mean $\mu^{(1)}/\deg(u)$. Therefore, we can estimate the degree of u as $\mu^{(1)}/t_{\min}(u)$. Since

Algorithm 3.2 Graph Inference (GI) algorithm

-
- 1: **Input** I : number of transfer steps, m_s : number of edges selected for the selection step, $\deg_{\text{ave}}$: estimated average degree of G , V_c : set of cascade sources, $\{T_{v_i}(\cdot) : v_i \in V_c\}$: timestamp information, the average mean $\mu^{(1)}$ and second order moment $\mu^{(2)}$ of the propagation time.
 - 2: Initialize $\eta_v = 0$ for each $v \in V_c$.
 - 3: **for** $s = 1, \dots, I$ **do**
 - 4: Compute weight matrix W in (3.5) and select m_s pairs $\{(v_i, v_j)\}$ having the m_s smallest $W(v_i, v_j)$ weights.
 - 5: **for** each pair of (v_i, v_j) selected **do**
 - 6: Construct $T_{v_i}^{v_j}(\cdot)$ and $T_{v_j}^{v_i}(\cdot)$ using (3.2).
 - 7: Update $\eta_v = \eta_v + 1$ for $v = v_i, v_j$.
 - 8: Update $T_{v_i}(\cdot) = (\eta_{v_i} T_{v_i}(\cdot) + T_{v_i}^{v_j}(\cdot)) / (\eta_{v_i} + 1)$ and $T_{v_j}(\cdot) = (\eta_{v_j} T_{v_j}(\cdot) + T_{v_j}^{v_i}(\cdot)) / (\eta_{v_j} + 1)$.
 - 9: **end for**
 - 10: **end for**
 - 11: Select $n \cdot \deg_{\text{ave}} / 2$ edges $\{(u, v)\}$ corresponding to the smallest $W(u, v)$ weights.
 - 12: **Output** $\{(u, v)\}$.
-

we observe a collection of cascades, then by averaging we obtain the estimate

$$\deg_{\text{ave}} = \frac{\sum_{u \in V_c} \mu^{(1)} / t_{\min}(u)}{|V_c|}. \quad (3.6)$$

Because of the outlier problem when estimating the parameter of an exponential distribution, we revise our estimation to make it robust according to [101, 102]. For other spreading models, we adopt the same heuristic to obtain $\deg_{\text{ave}}$. Simulation results in Section 3.3.2 demonstrates that (3.6) is a reasonable estimate of the average degree.

Summarizing the above discussions, our Graph Inference (GI) algorithm as a heuristic extension of ITI is shown in Algorithm 3.2. The overall time complexity of GI is $O(I|V_c||V|^2)$.

A possible generalization of Algorithm 3.2 if higher-order moments are available is to modify W_1, W_2 and hence W accordingly as follows. We can use $|T_{v_i}(u) - T_{v_i}(v)|$ to estimate the sample moments. For each $k \geq 1$, the k -th sample moment is denoted by $S_{u,v}^{(k)}(v_i)$ as the average of $|T_{v_i}(u) - T_{v_i}(v)|^k$ over $v_i \in V_c$. Suppose the $k_1 = 1, k_2, \dots, k_m$ -th moments are available. Let ϕ be a continuous m -variable

function, and for each $1 \leq i \leq m$, let

$$W_i(u, v) = \frac{1}{|V_c|} \min \left\{ \sum_{v_i \in V_c} S_{u,v}^{(k_i)}(v_i), \sum_{v_i \in V_c} |S_{u,v}^{(k_i)}(v_i) - \mu^{(k_i)}| \right\}. \quad (3.7)$$

We then define

$$W(u, v) = \phi(W_1(u, v), \dots, W_m(u, v)). \quad (3.8)$$

Under this generalization, the procedure depicted in Algorithm 3.2 uses $m = 2$, $k_1 = 1$, and $k_2 = 2$, while ϕ is the averaging function. The choice of ϕ should reflect one's belief about which moment should play a more important role in the network inference task.

3.3 Simulation results

In this section, we present simulation results to illustrate the performance of our proposed topology inference algorithms. We first apply our ITI algorithm on tree networks, and compare its performance with the tree reconstruction (TR) algorithm proposed in [19], which is most similar to ours in assumptions. We then perform simulations on general graphs, including some real-world networks, and compare the performance of the GI algorithm with the NetRate algorithm proposed in [18]. As the NetRate algorithm assumes knowledge of the diffusion distribution, we also study the performance impact of a mismatch between the assumed and actual distributions. To the best of our knowledge, there are no other works on topology inference making similar assumptions as ours. TR and NetRate are the closest methods that allow feasible comparison.

Suppose that A_G is the true adjacency matrix of G and A is an estimated adjacency matrix. To evaluate the performance of our method, we define the *edge recovery rate* as

$$R \triangleq 1 - \frac{\sum_{1 \leq i < j \leq n} |A(i, j) - A_G(i, j)|}{2|E|}. \quad (3.9)$$

For the same number of edges, each mistake in identifying an edge causes a mistake at another pair of nodes. To account for this, we have a factor of 2 in the denominator; and this makes R a real number in $[0, 1]$. The term $1 - R$ is called

FIG. 3.2: This figure shows the performance of ITI for different parameters. In (a), performance of ITI with $I = 0$ with varying $|V_c|$ and κ_v . In (b), performance of ITI with varying $|V_c|, \kappa_v, I$ and m_s . Curves with the same color have the same $|V_c|$ and κ_v values.

the *error rate*, in which both undetected edges and false positives are taken into account. All the simulation results shown in the following sections are averaged over 200 trials. For each trial, given the graph G , we randomly pick $|V_c|$ nodes. For each source $v_i \in V_c$, we initiate the diffusion process and obtain a cascade. We then run the algorithms given the $|V_c|$ cascades to obtain the edge recovery rate for this trial.

3.3.1 Tree networks

In this subsection, we show and discuss simulation results to study the performance of our proposed algorithm for tree networks. As we are considering tree networks, in the reconstruction step, we always have $|E| = |V| - 1$ edges.

In each simulation run, we randomly generate trees with $|V| = 500$ nodes. We then randomly choose $|V_c|$ distinct sources, and generate κ_v cascades per source using independent exponential spreading with mean $\mu^{(1)} = 1$ at each edge. In applications, κ_v can be different for distinct nodes v ; however, in our simulations below, we keep it constant for all v for simplicity. We apply the ITI algorithm, and evaluate the performance by using (3.9).

We first study the edge recovery rate by varying $|V_c|$ and κ_v , and skip the selection and transfer steps by setting $I = 0$ (see Fig. 3.2(a)). We notice that κ_v has a significant impact on the performance. If the number of cascades per source κ_v is

FIG. 3.3: Performance comparison between General-ITI and ITI.

small, the performance can be greatly improved if we perform additional transfer steps (see Fig. 3.2(b)). These results demonstrate the usefulness of the theory on redundant vertices. The choice $I = 2$ is usually enough, and the number of edges m_s in the selection step can be chosen between $|V|$ and $1.5|V|$. Moreover, comparing the curves for $|V_c|/|V| = 0.3, \kappa_v = 2$ with $|V_c|/|V| = 0.5, \kappa_v = 1$, we see that the performance is improved if we have more cascades originating from the same source. This suggests that the averaging process allows a better approximation of the distance on a tree.

In our experiments, we found that it is still possible to infer a tree topology when the mean propagation time is unknown using the following procedure: We first initiate $\mu^{(1)}$ to be a small value and run ITI to obtain a collection of edges, each with an estimated propagation time. We then estimate the mean propagation time by averaging the propagation times along the estimated edges. This procedure is then repeated. We call this heuristic method General-ITI. We compare the performance of General-ITI with ITI ($I = 2$ and $m_s = 1.5|V|$) in Fig. 3.3. We see that the estimated mean propagation time increases and exceeds the actual value as the number of iterations increases. The recovery performance improves over 2 or 3 iterations and is comparable with ITI, where the actual mean is known.

Finally, we compare ITI and General-ITI (3 iterations) with the TR algorithm proposed in [19]. Although the theoretical part of [19] assumes i.i.d. exponential distribution for propagation time along different edges, the TR algorithm itself requires $|V|$ and $\{T_{v_i}(\cdot) : v_i \in V_c\}$ as the only inputs. The comparison is shown in Fig. 3.4. In Fig. 3.4(b), we test the effect of having different diffusion distributions

FIG. 3.4: Performance comparison between ITI, General-ITI and TR. In (a), the propagation time follows a fixed exponential distribution. In (b), the propagation time along each edge follows a randomly selected Gamma distribution.

for different edges. The distributions are randomly selected unknown Gamma distributions with the same mean. As shown from the plots, our methods perform much better in all the cases because our methods do not require identical distributions for propagation times along different edges. The performance of General-ITI suggests that we can even recover a tree network without knowing the exact mean propagation time.

3.3.2 General graphs

For general graphs, we first perform experiments to show the estimation accuracy of $\deg_{\text{avg}}$ if we adopt (3.6) proposed in Section 3.2 occasion (iii). We consider two graphs: the Forest-fire network [99], and a real-world Email network [99] and four spreading models for the propagation time along each edge: exponential distribution $\text{Exp}(1)$ with fixed mean 1, exponential distribution with mean chosen uniformly and randomly in $[0.5, 1.5]$ (denoted as $\text{Exp}([0.5, 1.5])$), truncated Gaussian distribution $\mathcal{TN}(1, 0.5^2)$ with mean 1 and deviation 0.5, and Gamma distribution $\Gamma(1, 2)$ with shape parameter 1 and scale parameter 2. For each simulation, we choose $|V_c|/|V|$ uniformly at random from $[0.1, 1.0]$. We compute the sample mean μ_r and sample standard deviation σ_r of $\deg_{\text{ave}}/\deg_{\text{ave}}^*$ using 200 trials where $\deg_{\text{ave}}^*$ is the actual average degree of the graph. Simulation results are shown in Table 3.1. We see that (3.6) gives the best estimate in the exponential spreading model. For

other three models, we are also able to estimate the average degree within a reasonable range. In all our subsequent simulations, we use (3.6) to obtain $\deg_{\text{ave}}$.

TABLE 3.1: Performance of the estimated average degree.

μ_r, σ_r	Exp(1)	Exp([0.5, 1.5])	$\mathcal{TN}(1, 0.5^2)$	$\Gamma(1, 2)$
Forest-fire network	0.98, 0.2	1.09, 0.22	1.24, 0.28	0.98, 0.18
Email network	1.14, 0.17	1.24, 0.23	0.91, 0.37	1.20, 0.21

We compare our GI algorithm with the NetRate algorithm proposed in [18]¹ and KernelCascade algorithm proposed in [27]. NetRate works under a completely different set of assumptions (in particular, [18] assumes propagation along edges follows one of the following families of distributions: exponential, power-law, or Rayleigh). As it performs better than NetInf in [17] and ConNie in [20], we do not compare against the latter two methods. For KernelCascade, we choose a set of hyperparameters and kernels similar to the settings in [27].

To accommodate comparison with NetRate, we use the standard exponential distribution with mean 1 for propagation time along each edge. We compare the performance of NetRate, KernelCascade and GI on E-R graphs, the Forest-fire network [99], and a real-world Email network [99]. The parameters of all networks are described in the plots.

From Fig. 3.5, we see that our method performs best in all the tested cases if the number of cascades does not exceed 60% of the number of nodes. For E-R graphs with large average degree and the Email network, GI performs best for the entire spectrum of $|V_c|/|V|$ from 10% to 100%. GI has a noticeably better performance than NetRate and KernelCascade when $|V_c|/|V|$ is very small. For example, in the case of the Email network and $|V_c|/|V| = 10\%$, NetRate and KernelCascade have less than 5% edge recovery while GI has more than 27% edge recovery. We note that since the Email network is dense, all inference methods based on the assumption that diffusion across each edge follows a distribution will have limited performance. On the other hand, the advantage of NetRate starts to show up when there is a large number of cascades. In particular, if the ratio $|V_c|/|V|$ is closer to 1 (i.e., on average each node sends a cascade), then NetRate has a better

¹The source code for NetRate was retrieved from SNAP (Stanford Network Analysis Project; <http://snap.stanford.edu/data/memetracker9.html>). We thank the authors of [18] for sharing it online.

FIG. 3.5: Performance comparison between the NetRate algorithm (blue), KernelCascade algorithm (green) and the GI-algorithm (red), on (a) E-R graphs with 300 nodes and average degree 4, (b) E-R graphs with 300 nodes and average degree 8, (c) Forest-fire network with 500 nodes and average degree about 5 and (d) Email network with 500 nodes and average degree 12. The information propagation along the edges follow the exponential distribution with parameter $\lambda = 1$. The dashed curves show the performance of NetRate if there is a distribution mismatch for comparison purpose.

performance for certain graph types (Fig. 3.5(a) and (c)). KernelCascade performs worst but its performance improves significantly when the number of cascades is large. That is because KernelCascade learns the edge transmission functions from the data, which is difficult to accomplish with limited cascades. In addition, the choice of kernels and hyperparameters, which may vary for different networks or number of cascades, can also affect the performance.

Another advantage of our method is that it is much more computationally efficient than NetRate and KernelCascade. Using the same computational resource (Processor: Intel(R) Xeon(R) CPU E3-1226 v3 3.30GHz, RAM: 8.00GHz) under

the same simulation settings, the average time used to run an instance of GI, NetRate and KernelCascade is shown in Table 3.2. Our method is more suitable in time-critical applications when computational resources are limited.

TABLE 3.2: Computation time (in *seconds*) of GI, NetRate and KernelCascade for varying $|V_c|/|V|$

$ V_c / V $	0.1	0.2	0.3	0.4	0.5	0.6	0.6	0.8	0.9	1.0
Email (GI)	11.9	16.7	20.6	25.3	30.0	35.3	38.1	41.8	42.6	44.8
Email (NetRate)	866.2	1086.4	1380.5	1730.9	2102.4	2470.6	2877.2	3359.8	3880.1	5263.6
Email (KernelCascade)	1206.6	1550.1	1938.2	2504.2	3072.4	4034.1	5031.4	6584.6	8297.7	10852.3
Forest-fire (GI)	10.1	13.4	15.8	18.9	22.1	25.1	27.4	30.2	33.8	36.5
Forest-fire (NetRate)	681.5	1013.3	1354.3	1690.8	2083.2	2406.7	2806.8	3257.1	3757.4	5128.1
Forest-fire (KernelCascade)	739.7	1009.7	1319.8	1731.4	2288.6	3025.1	3788.6	5385.4	6936.3	9534.5

3.3.3 Distribution mismatch

For the next set of simulations, we test the effect of distribution mismatch on NetRate. Note that since both KernelCascade and GI does not assume any spreading distribution, they do not have such a problem.

The setup of the simulations is as follows: on each type of network, we generate the cascades using the exponential distribution with rate 1. We run the NetRate algorithm with the incorrect distributions: either the power-law distribution (POW) or the Rayleigh distribution (RAY), as these are the other two families of distributions discussed in [18]. The results are shown in the same Fig. 3.5 by dashed curves.

From the plots, we see that the performance of NetRate drops significantly if there is a distribution mismatch. For example, if the power-law distribution is used, the edge recovery of NetRate is close to 0% regardless of $|V_c|/|V|$ and the network type.

The experiments suggest that prior knowledge of the diffusion distribution is important to guarantee the performance of NetRate. In contrast, for both GI and KernelCascade, no such prior knowledge is required.

3.3.4 Heterogeneous spreading distributions

We next study the performance of GI when the spreading distributions are heterogeneous. Along each edge, the propagation time follows an exponential distribution with mean chosen uniformly and randomly in $[0.5, 1.5]$.

Since both GI and NetRate outperform each other in different $|V_c|/|V|$ regimes, we develop a procedure to fuse their results together. We first run NetRate to estimate the average mean and second order moment of the edge propagation time. Then, we use these in GI to compute the weights $W(u, v)$ in (8). To interpret these weights as “likelihood scores”, let $W^* = \min_{u,v} W(u, v)$, and

$$\ell(u, v) = \phi\left(\frac{W(u, v) - W^*}{\hat{\sigma}}\right),$$

where $\phi(\cdot)$ is the pdf of the standard normal distribution, and $\hat{\sigma}^2 = \frac{1}{n_e} \sum_{u,v} |W(u, v) - W^*|^2$ with the sum being taken over the $n_e = \deg_{\text{ave}} \cdot n/2$ smallest $W(u, v)$. We treat $\ell(u, v)$ as the likelihood score for (u, v) . The intuition is that those (u, v) with $W(u, v) = W^*$ are most likely edges, and we take this as the baseline. The “likelihood” of any other (u, v) being an edge is then computed w.r.t. this baseline. NetRate can also produce such likelihood scores (or in equivalent forms) for distinct nodes being connected by an edge in the graph G . We first normalize the scores (with unit total sum) for both methods; and take their average as the final likelihood scores. The edges with higher scores are selected. We call this approach NetRate-GI.

From the solid curves in Fig. 3.6, we observe that GI has the best performance when $|V_c|/|V|$ is small, while NetRate-GI has the best performance when $|V_c|/|V|$ is large. We also observe that in very dense networks like the Email network in Fig. 3.6(b), GI performs the best over a large range of $|V_c|/|V|$ values. This is because NetRate was not able to accurately estimate the edge propagation time moments as well as the likelihood scores, which led to errors in NetRate-GI. However, the good performance of GI comes at the price of knowing the average mean and second order moment of the edge propagation times *a priori*. We also test the effect of distribution mismatch on NetRate-GI. The results are shown in the same Fig. 3.6 by dashed curves. Fig. 3.6(a) shows that even though there is distribution mismatch, the combined approach NetRate-GI (RAY) can still outperform the rest. NetRate-GI (POW) performs worst; we believe that is because NetRate (POW) produces very poor likelihood scores according to Fig. 3.5.

FIG. 3.6: Performance comparison between NetRate, KernelCascade, GI and NetRate-GI on (a) Forest-fire network and (b) Email network with heterogeneous spreading distributions. The dashed curves show the performance of NetRate-GI if there is a distribution mismatch for comparison purpose.

3.3.5 Bimodal spreading distributions

We also study the performance of GI when the spreading distributions are bimodal. A mixture of two normal distributions with equal standard deviations is bimodal if their means differ by at least twice the common standard deviation [103]. In our simulations, we generate the propagation time along each edge from the truncated Gaussian distribution $\mathcal{TN}(\mu, \sigma^2)$ and the bimodal distributions $0.5\mathcal{TN}(\mu_1, \sigma^2) + 0.5\mathcal{TN}(\mu_2, \sigma^2)$, for different values of μ_1 and μ_2 , where $|\mu_1 - \mu_2| \geq 2\sigma$. Let $|V_c|/|V| = 0.5$, the results are shown in Table 3.3. We see that in the case of bimodal spreading distributions, the performance of GI is worse when $|\mu_1 - \mu_2|$ is larger or when the network is denser.

TABLE 3.3: Edge recovery rate comparison of three spreading distributions.

Network/Distribution	$\mathcal{TN}(5, 1)$	$0.5(\mathcal{TN}(3, 1) + \mathcal{TN}(7, 1))$	$0.5(\mathcal{TN}(1, 1) + \mathcal{TN}(9, 1))$
E-R tree	0.98	0.96	0.74
E-R graph ($\deg_{\text{ave}} = 4$)	0.97	0.68	0.42
E-R graph ($\deg_{\text{ave}} = 8$)	0.59	0.48	0.28
Forest-fire network	0.48	0.42	0.11
Email network	0.32	0.17	0.08

3.3.6 Real dataset

Finally, we use the MemeTracker dataset [18] from SNAP to compare GI, NetRate, KernelCascade and NetRate-GI. MemeTracker builds maps of the daily news cycle by analyzing around 9 million news stories and blog posts per day from 1 million online sources. We use hyperlinks between articles and posts to represent the flow of information from one site to other sites. A site publishes a new post and puts hyperlinks to related posts published by some other sites at earlier times. At a later time, this site's post can also be cited by newer sites. This procedure is then repeated and we are able to obtain a collection of timestamped hyperlinks between different sites (in blog posts) that refer to the same or closely related pieces of information. This collection of timestamps is recorded as a cascade.

We divide the dataset into two parts. From the first part, we extract a sub-network with top 500 sites and 2457 edges, which contains blog posts in two months. This is used as the historical data to estimate the moments of edge propagation times required by our algorithm. If one site re-posted a blog published by another site, we connect an edge between these two sites and obtain the propagation time according to the timestamp information. From this sub-network, we estimate the mean and second order moment of the edge propagation time as 0.46 second and 1.29 second squared respectively. We then use the second part of the dataset to extract 2400 cascades from 1,272,031 posts in a month to test different algorithms (notice that for each cascade, only a subset of vertices are timestamped).

For NetRate and NetRate-GI, we assume an exponential diffusion model. The comparison results are shown in Table 3.4. By comparison, NetRate-GI has the best performance when the number of cascades is at least 1600, with GI in second place. GI however has the best performance when the number of cascades is small. Note that the hyperlinks described above are used to obtain the groundtruth network topology. All 4 algorithms infer network topology based only on information cascades.

For other applications, the historical data we need to estimate the edge propagation time moments is similar to the first part of the dataset described above. The historical dataset's size can be small and may not even come from the same source. For example, while trying to infer the topology of a Facebook sub-network, we can use data from another known Facebook sub-network or another social network like

Twitter to estimate the moments. We see from results in Sections 3.3.4 and 3.3.5 that GI is relatively robust to errors in the moment estimation.

TABLE 3.4: Comparison of edge recovery rate on the MemeTracker dataset.

Number of cascades	GI	NetRate (EXP)	KernelCascade	NetRate-GI (EXP)
800	0.32	0.24	0.14	0.27
1600	0.53	0.46	0.35	0.57
2400	0.69	0.63	0.67	0.76

3.4 Conclusion

In this chapter, we have developed and methods for graph topology inference using information cascades and knowledge of some moments of the diffusion distribution across each edge, without needing to know the distribution itself. In the case of tree networks, we propose an iterative tree inference algorithm. Simulations demonstrate that our method outperforms the tree reconstruction algorithm in [19]. We then extended our tree inference method heuristically to general graphs. Our simulation results suggest that our graph inference algorithm performs reasonably well, if the total number of cascades is not too small compared to the size of the network, and often outperforms the NetRate algorithm in [18] and KernelCascade algorithm in [27]. Moreover, our method is suitable for time-critical applications owing to its low complexity.

Chapter 4

Sequential Source Estimation

In this chapter we study the problem of identifying single misinformation source in a network based on the network topology and a stream of infection timestamps.

4.1 Sequential Source Estimation

In this section, we first construct a stochastic process that we use to represent the reverse infection propagation process. Then, using a series of approximations, we derive a particle filter for sequentially estimating the misinformation source given a stream of infection timestamps.

4.1.1 Reverse Propagation Process

We define a V -valued discrete-time Markov chain $(X_n)_{n \geq 0}$ where $X_0 \sim \text{Unif}(V)$, and for $n \geq 1$, the conditional probability of $X_n = x_n$ given $X_{n-1} = x_{n-1}$ is given by

$$p(x_n \mid x_{n-1}) = \begin{cases} \alpha & \text{if } x_n = x_{n-1}, \\ \frac{1-\alpha}{|\partial x_{n-1}|} & \text{if } x_n \in \partial x_{n-1}, \\ 0 & \text{otherwise,} \end{cases} \quad (4.1)$$

where $|\partial x_{n-1}|$ is the number of neighbors of node x_{n-1} , and $\alpha \in (0, 1)$ is denoted as self-transition probability. We interpret this as a “reverse” propagation process

as follows: Consider the infection path $\rho(s_1, v_1)$ from the actual source s_1 to the first observed infected node v_1 . The node X_1 is a node in $\rho(s_1, v_1)$. We say that X_1 is an ancestor to v_1 . Similarly, given v_2 , we wish to find a common ancestor in $\rho(s_1, v_1) \cap \rho(s_1, v_2)$ to both v_1 and v_2 . If a neighbor of X_1 is also its ancestor in $\rho(s_1, v_1) \cap \rho(s_1, v_2)$, we let X_2 be chosen to be this neighbor, otherwise we let $X_2 = X_1$. This procedure is then repeated. We approximate this stochastic process with the Markov chain described in (4.1), where α is an appropriately chosen constant. In Section 4.2, we indicate how to choose this constant through simulations.

4.1.2 Particle Filtering

We can now cast our sequential source estimation problem as an approximate hidden Markov model (HMM). Let $(X_n)_{n \geq 0}$ be given by the Markov chain defined in (4.1), and the hidden states be $(Z_n)_{n \geq 1} = (X_n, \mathcal{O}_{n-1})_{n \geq 1}$, where $\mathcal{O}_0 = \emptyset$. Let the observations be $(y_n)_{n \geq 1} = (v_n, t_n)_{n \geq 1}$. The hidden states $(Z_n)_{n \geq 1}$ do not form a Markov chain, but we will adopt the particle filtering framework of a HMM as an approximation. Experiments in Section 4.2 indicate that this approximation does not unreasonably impair our estimation results.

To perform particle filtering, we make further approximations as follows. For the state transition probability, we take

$$\begin{aligned} p(z_n \mid z_{n-1}) &= p(x_n, \mathcal{O}_{n-1} \mid x_{n-1}, \mathcal{O}_{n-2}) \\ &\approx p(x_n \mid x_{n-1}), \end{aligned} \quad (4.2)$$

where we ignore the contributions of $\mathcal{O}_{n-1}$ and $\mathcal{O}_{n-2}$. We assume that infections are transmitted from the source to each infected node through the shortest path, so that the infection propagation graph is a tree. Suppose that x_n is infected at time $\hat{t}_{x_n}$. Then,

$$p(y_n \mid z_n) \approx f(t_n - \hat{t}_{x_n}; d(v_n, x_n)), \quad (4.3)$$

where $f(\cdot; d)$ is the d -fold convolution of f_τ . We estimate $\hat{t}_{x_n}$ from $\mathcal{O}_{n-1}$ as follows:

$$\hat{t}_{x_n} = \arg \max_t \prod_{i=1}^{n-1} f(t_i - t; d(v_i, x_n)), \quad (4.4)$$

where we have assumed that the infection time of each v_i , $i \leq n-1$, is independent of the others. This is an approximation to allow easy computation of $\hat{t}_{x_n}$.

To illustrate (4.4), we now derive the estimator $\hat{t}_{x_n}$ for the case where the propagation delay over each edge follows a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$. Then $f(\cdot; d)$ is the density of $\mathcal{N}(d\mu, d\sigma^2)$. It can be shown that in this case,

$$\hat{t}_{x_n} = \frac{\sum_{i=1}^{n-1} (t_i - d_i \mu) / d_i}{\sum_{i=1}^{n-1} d_i^{-1}}, \quad (4.5)$$

where $d_i = d(v_i, x_n)$. By taking the expectation of (4.5), it is clear that $\hat{t}_{x_n}$ in (4.5) is an unbiased estimate of t_{x_n} . In fact, this is true for any propagation delay distribution. For any propagation delay distribution for which no analytical form for $\hat{t}_{x_n}$ exists, either a numerical method can be used to solve (4.4) or we simply use (4.5) with μ being the mean of the distribution.

We can now describe the steps involved in our particle filter. Let $x_{1:n}$ represent $(x_i)_{i=1}^n$. For each $n \geq 1$, we wish to obtain a set of weighted particles $\{x_n^m, w_n^m\}_{m=1}^M$ so that we can approximate

$$p(x_n \mid y_{1:n}) \approx \sum_{m=1}^M w_n^m \delta(x_n - x_n^m), \quad (4.6)$$

where $\delta(\cdot)$ is the Dirac delta function. For each $n \geq 1$, we perform the following steps:

1. Sample particles from a proposal density

$$x_n^m \sim q(x_n \mid x_{1:n-1}^m, y_{1:n}). \quad (4.7)$$

2. Update $\{x_{n-1}^m, w_{n-1}^m\}$ to $\{x_n^m, w_n^m\}$ where

$$w_n^m \propto \beta_n w_{n-1}^m, \quad (4.8)$$

and

$$\beta_n = \frac{p(x_n^m \mid x_{n-1}^m) f(t_n - \hat{t}_{x_n}; d(v_n, x_n))}{q(x_n \mid x_{1:n-1}^m, y_{1:n})}. \quad (4.9)$$

Various particle filter algorithms differ in their choices of proposal density $q(\cdot)$ in (4.7). The popular bootstrap filter in [104] uses transition function (4.2) as the proposal density, and thus β_n in (4.9) is reduced to the likelihood function $f(t_n - \hat{t}_{x_n}; d(v_n, x_n))$. In our SSE algorithm, we also adopt this strategy. To avoid degeneracy, we also implement a resampling step in which $\{x_n^m, w_n^m\}$ is resampled to obtain M equally-weighted particles $\{\bar{x}_n^m, 1/M\}$, which are then used in place of x_n^m in the sampling step (4.7). We adopt the widely used systematic resampling method [105], which outperforms other resampling schemes in most scenarios and is easy to implement.

Finally, the misinformation source is found by finding the maximum a posteriori estimate using (4.6), i.e., $s_1^n = x_n^{m_n}$, where

$$m_n = \arg \max_m w_n^m. \quad (4.10)$$

The detailed SSE algorithm is shown in Algorithm 4.1. For each new observed y_n where $n \leq |V|$, we first perform a breadth-first search tree rooted at v_n and store the pairwise distances in a hash table. This process incurs a time complexity of $O(|V|^2)$. We then compute $\hat{t}_{x_n^m}$ for $m = 1, \dots, M_n$ according to (4.5), where $M_n \leq |V|$ is the number of distinct nodes used as particles. The complexity of this step is $O(nM_n)$. We have $nM_n \leq |V|^2$, therefore the overall time complexity for each new observed y_n is $O(|V|^2)$. Experiments in Section 4.2 also show that M_n generally decreases as n increases because the resampling step allows us to discard nodes with low weight. At each new observation, SSE thus incurs a lower time complexity than GAU, which requires $O(|V|^3)$ time complexity if it is to be recomputed for each new observation.

4.2 Experimental Evaluations

4.2.1 Synthetic Data

We first perform experiments on E-R graphs $ER(n, np)$ proposed in [98], where n is the number of nodes, and np is the expected degree of each node. We choose $n = 1000$, and $np = 2$ or 4 to simulate a graph that is relatively sparse or dense respectively. Some properties like the average diameter and pairwise distance are

Algorithm 4.1 Sequential Source Estimation (SSE)

Input: Adjacent matrix of the graph $\mathcal{G}$, a stream of timestamps $(v_n, t_n)_{n=1}^N$, and $f_\tau(\cdot)$

Output: Infection source estimates $(s_1^n)_{n=1}^N$

```

for  $n = 1, \dots, N$  do
  if  $n = 1$  then
    Sample  $x_1^m$  from  $\text{Unif}(V)$ , for  $m = 1, \dots, M$ .
    Let  $\bar{x}_1^m = x_1^m$ ,  $w_1^m = 1/M$ .
  else
    Sample  $x_n^m \sim p(x_n^m | \bar{x}_{n-1}^m)$  in (4.1) for  $m = 1, \dots, M$ .
    Find distinct nodes  $\{x_n^m\}_{m=1}^{M_n}$  and count their respective numbers  $\{c_n^m\}_{m=1}^{M_n}$ 
    where  $\sum_{m=1}^{M_n} c_n^m = M$ .
    for  $m = 1, \dots, M_n$  do
      Compute  $\hat{t}_{x_n^m}$  according to (4.5).
      Compute weight  $w_n^m = c_n^m p(y_n | z_n^m)$  according to (4.3).
    end for
    Normalize  $\{w_n^m\}$  so that  $\sum_{m=1}^{M_n} w_n^m = 1$ .
    Use systematic resampling to obtain  $M$  equally weighted particles
     $\{\bar{x}_n^m, 1/M\}_{m=1}^M$ .
     $s_1^n = x_n^{m_n}$  where  $m_n = \arg \max_m w_n^m$ .
  end if
end for

```

listed in Table 4.1. In each simulation, we assume that the propagation delay over every edge follows either a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ or an exponential distribution $\text{Exp}(\lambda)$.

We first generate 1000 timestamps initiated by a random node, and then randomly choose a portion of the timestamps to perform sequential source estimation. In Algorithm 4.1, we use $M = 2000$ particles. Simulation results for $\text{ER}(1000, 2)$ and $\text{ER}(1000, 4)$ are shown in Fig. 4.1 and Fig. 4.2, respectively. Each curve is averaged over 300 experiments. For the performance metric, we define the error distance as the distance between the estimated source and the real source. Following [51], we define $\gamma\%$ -accuracy as the proportion of simulations in which the real sources are ranked in the top γ percent of all the nodes. From parts (a) and (c) in both Fig. 4.1 and Fig. 4.2, we see that α should be chosen from $(0.9, 1.0)$ to get a better performance. The performance gets worse when $\alpha = 1$ because of the implementation of the resampling step. At early stage, the estimation of source is inaccurate with fewer observations. Then the actual source may have lower weight and be discarded after resampling steps. If $\alpha = 1$, then from (4.1) and (4.7) we

Graph	$ E / V $	Diameter	Pairwise distance
ER(1000, 2)	1	21.6	8.98
ER(1000, 4)	2	11	5.13
Facebook	25.9	9	2.95
Weibo	1.02	15	3.48

TABLE 4.1: Some graph properties of the networks used.

FIG. 4.1: Sequential source estimation on ER(1000, 2). (a) and (b) are based on Gaussian spreading with $\mu/\sigma = 2$, while (c) and (d) are based on exponential spreading with $\lambda = 1$. In (b) and (d) we average the estimation deviation over 200 experiments. In (a) and (c) we obtain the overall average by averaging the estimation deviation over 200 experiments and 40% timestamps.

see that the source node will never be included since then. Therefore, by setting $\alpha < 1$, our algorithm is able to correct early errors and becomes more robust. From parts (b) and (d) in both Fig. 4.1 and Fig. 4.2, we see that with 10% to 20% of the total timestamps, SSE finds an estimate that is on average within 2 hops of the real source if $\alpha = 0.95$.

FIG. 4.2: Sequential source estimation on ER(1000, 4). (a) and (b) are based on Gaussian spreading with $\mu/\sigma = 2$, while (c) and (d) are based on exponential spreading with $\lambda = 1$. In (b) and (d) we average the estimation deviation over 200 experiments. In (a) and (c) we obtain the overall average by averaging the estimation deviation over 200 experiments and 40% timestamps.

We compare SSE with GAU proposed in [50] in Fig. 4.3. Since GAU requires computing a breadth-first search tree rooted at every infected node, it has high complexity for large graphs. Therefore, we perform experiments on smaller graphs ER(200, 2). From Fig. 4.3, we see that SSE has smaller average estimation deviation and higher detection rate for both Gaussian and exponential spreading when there are fewer than 50% timestamps.

We also evaluate the performance of our algorithm with Facebook network provided by SNAP.¹ We extract a subgraph with 1034 nodes and 26749 edges for infection spreading using a Gaussian distribution for the edges' propagation delay with $\mu/\sigma = 4$. We then randomly choose timestamps to perform sequential source

¹<https://snap.stanford.edu/data/>

FIG. 4.3: Comparison between SSE and GAU on ER(200,2) for both Gaussian and exponential spreading, averaged over 300 experiments. Comparison is based on average error distance in (a) and 0.5%-accuracy in (b). In SSE, we set $\alpha = 0.95$.

FIG. 4.4: Sequential source estimation on a Facebook subnetwork.

estimation (we let $\alpha = 0.95$ and average over 300 experiments). Results in Fig. 4.4 show that the average estimation deviation decreases with more timestamps, and approaches to around 1.4 hop with 10% of the total timestamps available. However, the performance is not as good compared to the E-R graphs, because the Facebook network is much denser.

FIG. 4.5: Evaluation of SSE on the Weibo data.

4.2.2 Weibo Data

We evaluate the performance of SSE with a sample of Sina Weibo data provided by the WISE 2012 challenge.² We first extract a subgraph with 4642 nodes and 4755 edges (if user a retweets user b 's tweet then there is an edge between nodes a and b) using 44017 tweets. Some properties of the graph generated are summarized in Table 4.1. Secondly, we choose a message that was retweeted 2877 times and we use the first 2000 timestamps for SSE. The total time spanned by these tweets is about 5 hours. Finally, we learn the mean time of a tweet spreading through an edge from historical tweets, which yields an estimate of about 65 minutes. We set $\alpha = 0.95$ in SSE and adopt an exponential spreading model with different rates λ in SSE. We perform 300 experiments, and for each experiment, we randomly choose the ordering of the timestamps. Results averaged over all the experiments are shown in Fig. 4.5. We see that the performance is best when $1/\lambda = 65$ minutes, which matches the historical estimate, and is worse when $1/\lambda$ is too small or too big.

Finally, we calculate the standard deviation of the error distances obtained from 300 experiments on different networks. We use the exponential spreading, choose the fraction of timestamps to 0.1 and 0.2 and apply our SSE algorithm. Results in Table 4.2 suggest that the standard deviation decreases as the fraction of timestamps

²<http://www.wise2012.cs.ucy.ac.cy/challenge.html>

Fraction of timestamps	ER(200, 2)	Facebook	Weibo
0.1	1.20	0.87	0.42
0.2	0.96	0.75	0.39

TABLE 4.2: Standard deviation of the error distances on different networks.

increases. Therefore, we are more confident in the estimations of our algorithm if there are more observations.

4.3 Conclusion

In this chapter, we have developed a sequential infection source estimation algorithm for identifying an infection source using a stream of infection timestamps. Our method is based on constructing a reverse propagation process, which allows us to build a particle filter for sequential estimation. Each sequential update has low complexity, which allows the source estimate to be updated quickly whenever a new timestamp is available. Experiments on both synthetic and real-world data suggest that our approach can find the infection source to within a small number of hops from the true source.

Chapter 5

Single Source Estimation

In this chapter, we consider the single source case, i.e., $S = s_1$. We first derive a Maximum Likelihood Estimator (MLE) that simultaneously estimates the source and diffusion parameters when the network is a tree. We then describe our source estimation approach using Gromov matrices for general graphs. Finally, simulations on both synthetic and real-world networks are provided to compare the performance of our proposed algorithm with several other timestamp based source estimation algorithms in the literature.

5.1 A Gromov Matrix Interpretation

For two nodes u, v in a tree $T \subset G$, let $d_T(u, v)$ be the length of the unique path between u and v (assuming each edge has length 1). For a node s of T , the *Gromov product* (Definition 2.6 of [106]) of u and v in T with respect to (w.r.t.) s is given by

$$(u, v)_s = \frac{1}{2}(d_T(u, s) + d_T(v, s) - d_T(u, v)). \quad (5.1)$$

The following lemma summarizes some simple properties of the Gromov product.

Lemma 5.1. Let s, u and v be nodes of a tree T .

- (i) If $u = v$, then $(u, v)_s = d_T(u, s)$.

- (ii) $(\cdot, \cdot)_s$ is symmetric and non-negative.
- (iii) $(u, v)_s = 0$ if and only if s is on the unique path $[u, v]$.
- (iv) $(u, v)_s \leq d_T(u, s)$.

Proof. Both claim (i) and claim (ii) follow immediately from definition (and the triangle inequality). For claim (iii), $(u, v)_s = 0$ if and only if $d_T(u, s) + d_T(v, s) = d_T(u, v)$. On a tree, this happens only when $s \in [u, v]$. Claim (iv) clearly holds if $(u, v)_s = 0$. If $(u, v)_s > 0$, from claim (iii), there exists a node $w \in [u, v]$ such that $[u, s] \cap [u, v] = [u, w]$ and $[v, s] \cap [u, v] = [v, w]$. Then,

$$\begin{aligned} (u, v)_s &= \frac{1}{2}(d_T(u, w) + d_T(w, s) + d_T(v, w) + d_T(w, s) - d_T(u, w) - d_T(w, v)) \\ &= d_T(w, s) \\ &\leq d_T(u, s), \end{aligned}$$

and the proof is complete. $\square$

Let $U = \{u_1, \dots, u_n\} \subset V$ be a subset of nodes of G , $s \notin U$ be a node of G , and T be a tree that spans $U \cup \{s\}$ (i.e., $U \cup \{s\}$ is a subset of the node set of T). We define an $n \times n$ matrix $\mathbf{\Lambda}$ whose (i, j) -th entry is the Gromov product $(u_i, u_j)_s$ in T . We call $\mathbf{\Lambda}$ the *Gromov matrix* with base (T, s, U) , where T is known as its base tree and s its base node.

Recall that two trees T_1 and T_2 are said to be *isometric* to each other if there is a bijection $\phi : T_1 \rightarrow T_2$ such that $d_{T_1}(u, v) = d_{T_2}(\phi(u), \phi(v))$ for any $u, v \in T_1$. The map ϕ is called an *isometry*. We say that the two basis (T_1, s_1, U_1) and (T_2, s_2, U_2) are *isometrically equivalent* to each other if there is an isometry $\phi : T_1 \rightarrow T_2$ such that $\phi(s_1) = s_2$ and $\phi(U_1) = U_2$.

Proposition 5.1. Suppose that $\mathbf{\Lambda}$ is a Gromov matrix, then its base (T, s, U) is uniquely determined, up to isometric equivalence.

Proof. We assume that (T_1, s_1, U_1) and (T_2, s_2, U_2) are two basis of $\mathbf{\Lambda}$. We prove that they are isometrically equivalent to each other by induction on $|U_1| = |U_2|$. The statement is clearly true if $|U_1| = 1$. Indeed, the condition that T_i spans $U_i \cup \{s_i\}$, for $i = 1, 2$ imply that both T_1 and T_2 are simple paths, with end points s_i , and the

single node in U_i , respectively. Therefore, (T_1, s_1, U_1) and (T_2, s_2, U_2) are equivalent as T_1 and T_2 have the same length given by $\mathbf{\Lambda}$ (which is a 1×1 matrix).

Suppose the statement holds for $|U_1| = |U_2| = n$. Consider the case where $|U_1| = |U_2| = n + 1$. Let $\mathbf{\Lambda}'$ be the $n \times n$ upper-left block of $\mathbf{\Lambda}$. Since $\mathbf{\Lambda}$ is a Gromov matrix, so is $\mathbf{\Lambda}'$. For each $i = 1, 2$, there is a subset $U'_i \subset U_i$ with $|U'_i| = n$, and $T'_i \subset T_i$ such that (T'_i, s_i, U'_i) is a base of $\mathbf{\Lambda}'$. By the induction hypothesis, $(T'_i, s_i, U'_i), i = 1, 2$ are equivalent to each other via an isometry $\phi' : T'_1 \rightarrow T'_2$. Without loss of generality, for $i = 1, 2$, let $U_i = \{u_{i,k} : k = 1, \dots, n + 1\}$ such that $\phi'(u_{1,k}) = u_{2,k}$, for $k = 1, \dots, n$.

Let $1 \leq j \leq n$ be an index such that the $(j, n + 1)$ -th entry $[\mathbf{\Lambda}]_{j,n+1}$ of $\mathbf{\Lambda}$ satisfy $[\mathbf{\Lambda}]_{j,n+1} \geq [\mathbf{\Lambda}]_{k,n+1}$ for all $1 \leq k \leq n$. From Lemma 5.1(iv), for each $i = 1, 2$, let w_i be the unique node on the path from $u_{i,j}$ to s_i such that $d_{T'_i}(w_i, s_i) = [\mathbf{\Lambda}]_{j,n+1}$. As ϕ' is an isometry and $\phi'(s_1) = s_2, \phi'(u_{1,j}) = u_{2,j}$, we have $\phi'(w_1) = w_2$.

We claim that the path $[u_{i,n+1}, s_i]$ in T_i satisfies $[u_{i,n+1}, s_1] \cap T'_i = [w_i, s_i]$. To see this, by the choice of w_i , we have

$$[w_i, s_i] = [u_{i,n+1}, s_i] \cap [u_{i,j}, s_i] \subset [u_{i,n+1}, s_i] \cap T'_i.$$

On the other hand, the maximality on $[\mathbf{\Lambda}]_{j,n+1}$ implies that $[u_{i,n+1}, s_i] \cap T'_i$ cannot be a path longer than $[w_i, s_i]$. Hence, we have $[u_{i,n+1}, s_i] \cap T'_i = [w_i, s_i]$.

Therefore, $(T_i \setminus T'_i) \cup \{w_i\} = [w_i, u_{i,n+1}]$ for both $i = 1, 2$. Moreover, both $[w_1, u_{1,n+1}]$ and $[w_2, u_{2,n+1}]$ are of the same length $[\mathbf{\Lambda}]_{n+1,n+1} - [\mathbf{\Lambda}]_{j,n+1}$. Therefore, ϕ' can be extended to an isometry $\phi : T_1 \rightarrow T_2$ such that $\phi(u_{1,n+1}) = u_{2,n+1}$. The induction is now complete and the proposition is proved. $\square$

For a base (T, s, U) , as the ordering of the nodes in the set U is not fixed, its Gromov matrix $\mathbf{\Lambda}$ is determined uniquely up to conjugation by permutation matrices of size $|U|$. With this observation and Proposition 5.1, we conclude that $\mathbf{\Lambda}$ gives the same amount of information as (T, s, U) . Now we can describe a reformulation of the problem and the basic idea of our approach.

Since the propagation delays along each edge are continuous random variables, with probability one, the misinformation diffusion on a graph from a source s forms a propagation path that is a spanning tree of all the nodes infected by s . Therefore,

in the source inference task, in addition to the source s , we are implicitly required to find a T in the set of spanning subtrees in G of the observed infected set $\mathcal{V}$ rooted at s . If G is a tree, the source s uniquely determines T . However, in a general graph, there may be more than one subtree rooted at s that spans the observed infected nodes. Therefore, finding (s, T) together is no longer equivalent to finding s alone.

In a dense graph, it is usually intractable to identify all possible s and spanning subtrees rooted at s . On the other hand, the results in this section imply that to identify s and the misinformation propagation tree T is equivalent to estimating the Gromov matrix associated with the misinformation diffusion process. In this thesis, we formulate the source estimation problem as an optimization of a cost function of a parametrized Gromov matrix and the diffusion parameters (μ, σ^2) . By optimizing over a parametrized family of Gromov matrices, we overcome the intractability of identifying all possible Gromov matrices or misinformation propagation paths. However, the base tree of the optimal Gromov matrix found may not correspond to an actual subtree in G . Our proposed approach is therefore a heuristic in the case of general graphs. A widely used heuristic in the source inference literature [32, 50, 52, 83] for general graphs is the BFS heuristic, which assumes that the misinformation propagation path is a BFS tree, which means that the misinformation is spread from a source to each observed infected node along a minimum-length path. However, this approximation has many limitations. For example, in general there exists more than one BFS tree rooted at a node, and the BFS heuristic typically chooses only one BFS tree [32, 50, 52, 83]. Furthermore, the actual diffusion tree is not likely to be a BFS tree, especially when the graph is dense or σ^2/μ is large. In our Gromov matrix approach, some BFS trees are included in the family of Gromov matrices that we optimize over. Therefore, our method is expected to achieve better performance than the BFS heuristic.

5.2 Maximum Likelihood Estimator

We first consider the special case where G is a tree. Recall that $\mathcal{V} = \{v_1, \dots, v_n\}$ is a set of observed infected nodes with corresponding infection timestamps $\mathbf{T} = [t_1, \dots, t_n]'$. Suppose that s is the source node. Let t_0 be the time the source node starts its spreading. Then, it is easy to see that the infection time of each node

$v_k \in \mathcal{V}$ is $t_k \sim \mathcal{N}(t_0 + d_G(s, v_k)\mu, d_G(s, v_k)\sigma^2)$, since all propagation delays $\{\tau_{uv}\}$ are i.i.d. $\mathcal{N}(\mu, \sigma^2)$ variables. The covariance of t_i and t_j is $[\mathbf{\Lambda}_s]_{i,j} \cdot \sigma^2$, with $\mathbf{\Lambda}_s$ being the Gromov matrix with base $(T, s, \mathcal{V})$, where T is the subtree rooted at s that spans $\mathcal{V}$. The likelihood of observing the infection times $\mathbf{T}$ is then given by

$$p(\mathbf{T} \mid s, t_0, \mu, \sigma^2) = \frac{\exp\left(-\frac{1}{2\sigma^2}(\mathbf{T} - \mathbf{D}_s\boldsymbol{\beta}_0)' \mathbf{\Lambda}_s^{-1}(\mathbf{T} - \mathbf{D}_s\boldsymbol{\beta}_0)\right)}{\sqrt{(2\pi\sigma^2)^n \det \mathbf{\Lambda}_s}}, \quad (5.2)$$

where $\boldsymbol{\beta}_0 = [t_0, \mu]'$, and $\mathbf{D}_s$ is a matrix with its k -th row being $[1, d_G(s, v_k)]$. We have the following result.

Proposition 5.2. Suppose that G is a tree. Then, the MLE $(\hat{s}, \hat{t}_0, \hat{\mu}, \hat{\sigma}^2) = \arg \max p(\mathbf{T} \mid s, t_0, \mu, \sigma^2)$ for (s, t_0, μ, σ^2) is given by

$$\begin{aligned} \hat{s} &= \arg \min_{s \in V \setminus \mathcal{V}} \det(R_s \mathbf{\Lambda}_s / n), \\ [\hat{t}_0, \hat{\mu}]' &= \boldsymbol{\beta}_{\hat{s}}, \\ \hat{\sigma}^2 &= R_{\hat{s}} / n, \end{aligned} \quad (5.3)$$

where

$$\begin{aligned} R_s &= (\mathbf{T} - \mathbf{D}_s\boldsymbol{\beta}_s)' \mathbf{\Lambda}_s^{-1}(\mathbf{T} - \mathbf{D}_s\boldsymbol{\beta}_s), \\ \boldsymbol{\beta}_s &= (\mathbf{D}_s' \mathbf{\Lambda}_s^{-1} \mathbf{D}_s)^{-1} \mathbf{D}_s' \mathbf{\Lambda}_s^{-1} \mathbf{T}. \end{aligned}$$

Proof. For any estimator $\hat{s}$ of s_1 , by maximizing (5.2) w.r.t. $\boldsymbol{\beta}_0 = [t_0, \mu]'$, the MLE of $\boldsymbol{\beta}_0$ is given by

$$\begin{aligned} \boldsymbol{\beta}_{\hat{s}} &= \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^{2 \times 1}} (\mathbf{T} - \mathbf{D}_{\hat{s}}\boldsymbol{\beta})' \mathbf{\Lambda}_{\hat{s}}^{-1}(\mathbf{T} - \mathbf{D}_{\hat{s}}\boldsymbol{\beta}) \\ &= (\mathbf{D}_{\hat{s}}' \mathbf{\Lambda}_{\hat{s}}^{-1} \mathbf{D}_{\hat{s}})^{-1} \mathbf{D}_{\hat{s}}' \mathbf{\Lambda}_{\hat{s}}^{-1} \mathbf{T}. \end{aligned}$$

Similarly, the MLE of σ^2 is given by

$$\begin{aligned} \hat{\sigma}^2 &= \arg \max_{\sigma^2 \in \mathbb{R}} \frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp\left(-\frac{R_{\hat{s}}}{2\sigma^2}\right) \\ &= R_{\hat{s}} / n. \end{aligned}$$

Finally, the MLE of s_1 is

$$\begin{aligned}\hat{s} &= \arg \max_{s \in V \setminus \mathcal{V}} p(\mathbf{T} \mid s, \boldsymbol{\beta}_s, \sigma^2 = R_s/n) \\ &= \arg \min_{s \in V \setminus \mathcal{V}} \det(R_s \boldsymbol{\Lambda}_s/n),\end{aligned}$$

and the proof is complete. $\square$

The reference [50] derives the MLE of the source s_1 by assuming that (μ, σ^2) are known, while Proposition 5.2 provides the MLE for both the source and diffusion parameters. Recall that we have made the assumption $S \cap \mathcal{V} = \emptyset$, this simplifies our exposition and avoid non-invertible matrices in our cost functions in (5.2), (5.3) and (5.9). Actually this assumption is not essential since one can generalize our methods by adding an additional step in which each node in $\mathcal{V}$ is considered a potential source candidate, computing an appropriate variant of the respective cost functions in (5.2), (5.3) and (5.9) for these potential source nodes, and then choosing the node from amongst $\mathcal{V}$ and that estimated by our method with the lowest cost to be the final estimate.

5.3 Single Source Estimation for General Graphs

In this section, we consider source estimation for general graphs. To find the MLE for the source, we need to optimize the likelihood in (5.2) over all possible diffusion trees rooted at each candidate source node. As discussed in Section 5.1, this is intractable due to multiple paths between each pair of nodes. Therefore, we propose to perform the optimization over a parameterized family of Gromov matrices as follows. We call the following procedure GSSI, which is summarized in Algorithm 5.1.

- (i) For each candidate source node s , we first find two BFS trees. Let the two corresponding Gromov matrices be $\boldsymbol{\Lambda}_s^1$ and $\boldsymbol{\Lambda}_s^2$. For each $\theta \in [0, 1]$, let

$$\begin{aligned}\mathbf{M}_s(\theta) &= \theta \boldsymbol{\Lambda}_s^1 + (1 - \theta) \boldsymbol{\Lambda}_s^2, \\ \boldsymbol{\beta}_s(\theta) &= (\mathbf{D}'_s \mathbf{M}_s^{-1}(\theta) \mathbf{D}_s)^{-1} \mathbf{D}'_s \mathbf{M}_s^{-1}(\theta) \mathbf{T},\end{aligned}$$

and

$$R_s(\theta) = (\mathbf{T} - \mathbf{D}_s \boldsymbol{\beta}_s(\theta))' \mathbf{M}_s^{-1}(\theta) (\mathbf{T} - \mathbf{D}_s \boldsymbol{\beta}_s(\theta)).$$

We find

$$\theta_s = \arg \min_{\theta \in [0,1]} \det(R_s(\theta) \mathbf{M}_s(\theta)/n), \quad (5.4)$$

and let $\mathbf{M}_s \triangleq \mathbf{M}_s(\theta_s)$.

(ii) We introduce a target covariance matrix $\mathbf{H}_s$, which is chosen to be

$$\mathbf{H}_s \triangleq m_s \mathbf{I}_n \text{ or } \mathbf{H}_s \triangleq \text{diag}(\mathbf{M}_s), \quad (5.5)$$

where $m_s = \text{tr}(\mathbf{M}_s)/n$, $\mathbf{I}_n$ is the identity matrix with dimension n , and $\text{diag}(\mathbf{M}_s)$ is the diagonal matrix with $[\text{diag}(\mathbf{M}_s)]_{i,i} = [\mathbf{M}_s]_{i,i}$. This yields $\text{tr}(\mathbf{H}_s) = \text{tr}(\mathbf{M}_s)$. For each $\alpha \in [0, 1]$, let

$$\mathbf{A}_s(\alpha) = \alpha \mathbf{H}_s + (1 - \alpha) \mathbf{M}_s, \quad (5.6)$$

$$\tilde{\boldsymbol{\beta}}_s(\alpha) = (\mathbf{D}_s' \mathbf{A}_s^{-1}(\alpha) \mathbf{D}_s)^{-1} \mathbf{D}_s' \mathbf{A}_s^{-1}(\alpha) \mathbf{T}, \quad (5.7)$$

and

$$\tilde{R}_s(\alpha) = (\mathbf{T} - \mathbf{D}_s \tilde{\boldsymbol{\beta}}_s(\alpha))' \mathbf{A}_s^{-1}(\alpha) (\mathbf{T} - \mathbf{D}_s \tilde{\boldsymbol{\beta}}_s(\alpha)).$$

We find

$$\alpha_s = \arg \min_{\alpha \in [0,1]} \det(\tilde{R}_s(\alpha) \mathbf{A}_s(\alpha)/n), \quad (5.8)$$

and let $\mathbf{A}_s \triangleq \mathbf{A}_s(\alpha_s)$.

(iii) We define our estimator $(\hat{s}, \hat{t}_0, \hat{\mu}, \hat{\sigma}^2)$ for $(s_1, t_0, \mu, \sigma^2)$ as

$$\begin{aligned} \hat{s} &= \arg \min_{s \in V \setminus \mathcal{V}} \det(R_s(\alpha_s) \mathbf{A}_s/n), \\ [\hat{t}_0, \hat{\mu}]' &= \tilde{\boldsymbol{\beta}}_{\hat{s}}(\alpha_{\hat{s}}), \\ \hat{\sigma}^2 &= \tilde{R}_{\hat{s}}(\alpha_{\hat{s}})/n. \end{aligned} \quad (5.9)$$

Steps (i) and (ii) above essentially construct a Gromov matrix to approximate the misinformation diffusion path starting at a node s , based on the observed infection timestamps $\mathbf{T}$. In step (ii), we let the new Gromov matrix $\mathbf{A}_s$ be a convex combination of $\mathbf{M}_s$ and a predefined invertible target matrix $\mathbf{H}_s$ as shown in (5.6). This construction is in the same spirit as the Stein-type shrinkage covariance matrix estimator proposed in [107, 108], which is used to overcome the ill-conditioning problem of the sample covariance matrix. If $\mathbf{M}_s$ in (5.6) is replaced with the sample covariance matrix, then $\mathbf{A}_s(\alpha)$ is the Stein-type covariance estimator, which can be written as a convex combination of the sample covariance matrix and a predefined invertible target matrix $\mathbf{H}_s$. Two commonly used $\mathbf{H}_s$ are shown in (5.5). The parameter α is known as the shrinkage intensity, whose optimization can be found in works such as [107, 108]. In step (ii), we borrow the same idea; the difference is that we obtain the parameter α_s by solving the optimization problem in (5.8). In this heuristic, the Gromov matrix $\mathbf{A}_s$ obtained in step (ii) may not correspond to an actual subtree of G . However, in certain important special cases, for example, where either $\mathbf{H}_s$ or $\mathbf{M}_s$ is diagonal, then $\mathbf{A}_s$ corresponds to a subtree of G .

Forming convex combinations allows us to systematically approximate the Gromov matrices of spanning trees of G . In step (i), we propose to find the two BFS trees in opposite search directions: We first index the nodes from 1 to $|V|$. Then we construct one BFS tree by prioritizing nodes with smaller indices, and another BFS tree by prioritizing nodes with bigger indices. Choosing BFS trees labeled in opposite directions may give us very distinct spanning trees; and hence we obtain a large family of Gromov matrices by taking convex combinations. We illustrate the geometric intuition by a simple example as in Fig. 5.1.

We now give an interpretation of step (ii). When the graph is dense and the variance of the misinformation propagation delay is large, the spreading paths from the source to the observed infected nodes can be approximately regarded as uncorrelated (i.e., the spanning tree is a star tree). Therefore α in (5.6) indicates how far the spanning tree is from a star tree. Intuitively, if the graph is denser, then α_{s_1} is greater, which is validated by the simulation results in Fig. 5.7 of Section 5.5. To demonstrate that step (ii) improves our estimation performance, we present simulations in Section 5.5 where we compare the performance of GSSI with another procedure that we call naive-GSSI, which omits step (ii) and replaces $\mathbf{A}_s$ in step (iii) with $\mathbf{M}_s$.

FIG. 5.1: In this example, red nodes are the sources, blue nodes are the observed nodes. The Gromov matrices of the two trees on the left and middle are $\mathbf{M}_1$ and $\mathbf{M}_2$, respectively. The convex combination $\mathbf{M}_3 = (\mathbf{M}_1 + \mathbf{M}_2)/2$ is the Gromov matrix of the tree on the right.

In step (iii), the proposed estimator is simply the MLE assuming that $\mathbf{A}_s$ is the Gromov matrix of the underlying diffusion tree.

More discussions on the properties of Gromov matrix can be found in [109]. From the Corollary 1 of [109], we see that the convex combination of two Gromov matrices may not be a Gromov matrix, unless one of them is diagonal. Therefore $\mathbf{M}_s$ is in general not a Gromov matrix. However, [109] provided a procedure called G-convex to modify some entries of $\mathbf{M}_s$ to obtain a Gromov matrix. If the modified $\mathbf{M}_s$ is used, then $\mathbf{A}_s$ is a Gromov matrix of a weighted tree (though may not be a subtree of G) since $\mathbf{H}_s$ is diagonal. Through a case study, the paper [109] provided further insights into how well convex combinations of Gromov matrices can approximate an information propagation path in a network. We refer the interested reader to paper [109] for more details. In our thesis, we do not apply G-convex so that $\mathbf{A}_s$ may not be a Gromov matrix. However, experiments in Section 5.5 indicate that this does not unreasonably impair our estimation results.

For each node in $V \setminus \mathcal{V}$ we need to find two of their BFS trees, which can be computed with time complexity $O(|V|^2)$.¹ The optimization problems in (5.4) and (5.8) do not have analytical solutions. We utilize standard iterative techniques like gradient descent to solve them, which require a time complexity of $O(|\mathcal{V}|^3)$, assuming that the maximum number of iterations used is fixed. The overall time complexity

¹We say that $f(N) = O(g(N))$ if $f(N)/g(N) \leq k$ for some fixed $k > 0$ as $N \rightarrow \infty$.

Algorithm 5.1 Gromov Single Source Identification (GSSI)

Input: Adjacency matrix of the graph G , $\mathcal{V} = \{v_1, \dots, v_n\}$ and $\mathbf{T} = [t_1, \dots, t_n]'$.

Output: $(\hat{s}, \hat{t}_0, \hat{\mu}, \hat{\sigma}^2)$

- 1: Let $f_m = \infty$.
 - 2: **for** every $s \in V \setminus \mathcal{V}$ **do**
 - 3: Find two BFS trees rooted at s in opposite search directions to obtain $\mathbf{\Lambda}_s^1$ and $\mathbf{\Lambda}_s^2$.
 - 4: Find θ_s according to (5.4) to obtain $\mathbf{M}_s$.
 - 5: Find α_s according to (5.8) to obtain $\mathbf{A}_s$.
 - 6: Compute $f_s = \det(R_s(\alpha_s)\mathbf{A}_s/n)$.
 - 7: **if** $f_s \leq f_m$ **then**
 - 8: Set $f_m = f_s$, and $\hat{s} = s$.
 - 9: **end if**
 - 10: **end for**
 - 11: Set $(\hat{t}_0, \hat{\mu}, \hat{\sigma}^2)$ according to (5.9).
-

of GSSI is then $O(|V|(|V|^2 + |\mathcal{V}|^3)) = O(|V|^3 + |V||\mathcal{V}|^3)$. To compare with other algorithms, the time complexity of TRBS proposed in [54] is $O(|\mathcal{V}||V|^2)$; and the time complexity of the method proposed in [50] is $O(|V|^3 + |V||\mathcal{V}|^3)$ (note that $|V|$ matrix inverses are required in [50], each incurring a time complexity of $O(|\mathcal{V}|^3)$), assuming that the diffusion parameters are known a priori.

5.4 Discussion

In the GSSI algorithm, if G is a tree then we do not need to optimize θ because $\mathbf{\Lambda}_s^1 = \mathbf{\Lambda}_s^2$, but in this case, GSSI is not equivalent to MLE due to the introduction of the parameter α_s and target matrix $\mathbf{H}_s$. However, simulations in Section 5.5 indicate that the introduction of α_s and $\mathbf{H}_s$ does not unreasonably impair our estimation results in trees, while the introduction of the additional step in (5.8) leads to better performance for general graphs.

Simulations in Section 5.5 suggest that when G is a tree, then with high probability the MLE in Proposition 5.2 and GSSI give the same source estimates, and $\alpha_{\hat{s}}$ obtained by GSSI is close to 0, which implies that the performance of GSSI is close to the MLE when the graph is a tree. We provide some theoretical results in Section 5.6 that indicate why this is true.

Graph	$ E / V $	Diameter	Average pairwise distance
ER(500)	1.00	20	9.23
BA(500)	1.00	15	6.40
ER(500, 4)	1.98	11	4.69
BA(500, 4)	1.99	7	3.82
ER(500, 16)	8.12	4	2.54
BA(500, 16)	7.88	4	2.46
Enron	9.86	9	3.32
Facebook	25.9	9	2.95
Twitter	5.21	10	3.41

TABLE 5.1: Some graph properties of the networks used in our simulations.

5.5 Simulations for Single Source Estimation

In this section, we present simulation results on both synthetic and real networks to compare the performance of GSSI with the MLE in Proposition 5.2, the TRBS algorithm proposed in [54], and the method proposed in [50], which we call the GAU algorithm. We note that both the TRBS and GAU algorithms require prior knowledge of the mean and/or variance of the propagation delay along each graph edge, while our approach estimates these parameters from the observed timestamps.

We first perform simulations on two kinds of random trees. Starting from one node, we add a new node in every step and attach it to one of the existing nodes with probabilities proportional to their degrees to obtain a scale-free tree, or attach the new node to one of the existing nodes randomly to obtain a non-scale-free tree. We call these the Barabási-Albert (B-A) tree and Erdős-Rényi (E-R) tree, respectively. We denote B-A trees as $\text{BA}(N)$ and E-R trees as $\text{ER}(N)$, where N is the number of nodes. Some properties like the diameter and average pairwise distance are listed in Table 5.1. Our simulations indicate that choosing $\mathbf{H}_s = m_s \mathbf{I}_n$ or $\text{diag}(\mathbf{M}_s)$ in (5.5) does not lead to much difference in the performance of GSSI. Therefore, here we only present the results for $\mathbf{H}_s = m_s \mathbf{I}_n$.

For each simulation, we randomly pick a node from the network to be the source. We simulate the propagation delays along each edge using a truncated Gaussian distribution $\mathcal{TN}(2, 1)$, and randomly choose a subset of nodes as observed nodes. We perform 300 simulation runs. The error distance is defined as the distance between the estimated source and the real source. All four algorithms rank all the

nodes in the graph according to their likelihood of being the source. Following [51], we define $\gamma\%$ -accuracy as the proportion of simulations in which the real sources are ranked in the top γ percent of all the nodes.

Simulation results are shown in Fig. 5.2. We only consider the case where the fraction of timestamps is less than 50%. Simulations indicate that the performance of GSSI is very close to the MLE when G is a tree, which means that the introduction of parameter α_s and target matrix $\mathbf{H}_s$ in (5.6) does not unreasonably impair our estimation results when G is a tree. We observe that GSSI performs no worse than GAU in almost all cases, and better than TRBS for both kinds of random trees although GSSI requires less information. The reason why GAU performs worse than GSSI in most cases is because the simulated propagation delays are generated using a truncated Gaussian distribution instead of the Gaussian distribution assumed by GAU. Although GSSI also assumes a Gaussian distribution, it estimates the distribution parameters from the observed data, which help to mitigate this mismatch. TRBS has the worst performance because it does not utilize any variance information. For GSSI, we compute the mean squared error (MSE) of the estimates $\hat{\mu}$ and $\hat{\sigma}^2$ w.r.t. $\mu = 2$ and $\sigma^2 = 1$ respectively in Fig. 5.3, which show that the MSE of GSSI is very close to that of the MLE.

We next perform simulations on general B-A graphs $\text{BA}(N, Np)$ proposed in [110] and E-R graphs $\text{ER}(N, Np)$, where N is the number of nodes, and Np is the expected degree of each node. B-A graphs are scale-free while E-R graphs are not. We choose $N = 500$, and $Np = 4$ and 16 to simulate graphs with different densities. We also test on two real networks including the Enron email network² and a Facebook network³ provided by SNAP. For the Enron email network, we extract a subgraph with 670 nodes and 3303 edges, and for the Facebook network, we extract a subgraph with 1034 nodes and 26749 edges. See Table 5.1 for some graph properties. To demonstrate that the introduction of the parameters θ_s , α_s and $\mathbf{H}_s$ can improve performance, we also compare GSSI with two additional algorithms which we call BFS-MLE and naive-GSSI. For BFS-MLE we simply use the BFS-tree heuristic: For each node $s \in V \setminus \mathcal{V}$, we find a BFS tree rooted at s and then perform MLE to compute $g_s \triangleq \det(R_s \mathbf{\Lambda}_s / n)$ from Proposition 5.2. Finally, we minimize g_s over $s \in V \setminus \mathcal{V}$ to obtain the source estimate. For the naive-GSSI, we follow the same procedure as GSSI, but only optimize θ_s in (5.4), and let $\alpha_s = 0$.

²<https://snap.stanford.edu/data/email-Enron.html>

³<https://snap.stanford.edu/data/egonets-Facebook.html>

FIG. 5.2: Single source estimation on ER(500) and BA(500).

FIG. 5.3: MSE of $\hat{\mu}$ and $\hat{\sigma}^2$ obtained by MLE and GSSI.

Simulation results are shown in Figs. 5.4 to 5.6. The plots show that in almost all the cases, GSSI performs best and achieves significant improvement over BFS-MLE, while naive-GSSI also performs better than BFS-MLE. Therefore, we see that the introduction of θ_s , α_s and $\mathbf{H}_s$ in GSSI can improve estimation performance.

FIG. 5.4: Single source estimation on ER(500, 4) and BA(500, 4). For the diffusion process, set $(\mu, \sigma^2) = (2, 1)$.

We also study how α_s in (5.8) varies with increasing fraction of observed timestamps. From Fig. 5.7, we see that for two kinds of random trees, the average α_s is very close to 0, which agrees with the intuition provided by Lemma 5.6.1 in Section 5.6. For B-A and E-R graphs, the average α_s increases as the average degree of the graph gets larger, which is intuitively satisfying because the actual misinformation propagation path can deviate significantly from BFS trees as the graphs become denser.

Both GAU and SSE (proposed in Chapter 4) require to know the diffusion parameters (μ, σ^2) . We also test the performance of GAU and SSE when diffusion parameters are estimated by GSSI according to (5.9). We call two algorithms GAU2 and SSE2. We perform simulations on several networks from Table 5.1, and use the same simulation settings as the above. We set the fraction of timestamps to 20% and compute the average error distances for all algorithms. Comparison results are shown in Fig. 5.8. We see that GSSI still performs best. GAU2 and

FIG. 5.5: Single source estimation on ER(500, 16) and BA(500, 16), with $(\mu, \sigma^2) = (3, 1)$.

SSE2 perform better than GAU and SSE, which indicates that diffusion parameters estimated by GSSI are more reasonable if we assume the diffusion tree is a BFS tree.

Finally, we apply our GSSI algorithm to find the source of malware propagation in online social networks. Malware attacks have become ubiquitous in online social networks such as Facebook and LinkedIn [111, 112]. A successful attack using malware in a network can result in tens of millions of accounts being compromised and users' computers being infected. For example, Trojan, as the most popular type of malware, is able to steal confidential information, install ransomware and infect other computers in the network. Therefore, it is important to identify the sources of Trojans in order to analyse and remove them as early as possible. There have been many works studying the propagation of Trojans [113–116]. In this chapter, we adopt the propagation model in [116], which is a spatial-temporal SIR model that takes into account the network topology and temporal dynamics of user activities. Furthermore, it considers characteristics of modern Trojans, security practices, and

FIG. 5.6: Single source estimation on two real networks, with $(\mu, \sigma^2) = (3, 1)$. FIG. 5.7: Simulations to study how α_s varies with the fraction of observed timestamps and for different graphs.

FIG. 5.8: Performance comparison for different algorithms.

FIG. 5.9: Malware source estimation on Facebook network.

user behaviors. Similar to the simulations in [116], we use the Facebook network subgraph described above. Parameters used for the propagation model are shown in Table 5.2 (see [116] for more details). We assume that the malware is a zero-day Trojan, which propagates fast. Therefore, it has a high p_i , and small q_i and δ_i . The parameter $\beta_{\max}$ is set according to a survey conducted by Microsoft [117]. Since GAU and TRBS require knowledge of the mean propagation delays, we average the misinformation propagation times over the edges to obtain an estimate. Simulation results are shown in Fig. 5.9. The plots show that although the propagation model violates our Gaussian assumption, GSSI still performs best in most cases and is able to estimate the true malware source within 1.92 hops with at least 20% infection timestamps.

To summarize, we see that the estimation accuracy for all algorithms is influenced

Parameter	Description
$p_i = 0.8$	Probability of user i following a malicious post, unknowingly downloading the malware and executing it.
$\delta_i = 0.2$	Probability of user i accepting clean-up solutions from his non-infectious friends.
$q_i = 0.2$	Probability of user i recovering independently without assistance from his friends.
$\beta_{\max} = 0.75$	Percentage of the social network population that has anti-virus products installed on their computers.
$\tau_i \sim \text{Exp}(0.2)$	User message checking time.

TABLE 5.2: Parameter description and setting for malware propagation model of [116].

Graph	Error reduction(%)
ER(500)	21.2
BA(500)	4.5
ER(500, 4)	10.4
BA(500, 4)	13.0
ER(500, 16)	38.5
BA(500, 16)	67.9
Enron	21.6
Facebook	24.8
Facebook (malware)	6.1

TABLE 5.3: Percent of reduced average error distance GSSI compared with the next best algorithm.

by the network topology, diffusion model and fraction of timestamps. Generally it is costly to observe too many nodes, so we may need to estimate the source to within a small number of hops of the real source with a small fraction of observed timestamps. In all our experiments, GSSI is able to estimate the source on average to within 1.8 hops with only 30% of timestamps. A summary comparison of GSSI w.r.t. the next best algorithm used in our experiments when 30% timestamps are observed is provided in Table 5.3.

5.6 Appendix

This appendix is corresponding to Section 5.4. We provide some theoretical results to indicate why the performance of GSSI in Algorithm 5.1 is close to the MLE when the graph is a tree.

Suppose that the estimates by GSSI $(\hat{s}, [\hat{t}_0, \hat{\mu}]', \hat{\sigma}^2) = (s_1, \beta_0, \sigma^2)$, the true parameter values. Then, $\alpha_{\hat{s}} = \alpha_{s_1}$ in (5.8) is the solution of the following optimization problem:

$$\begin{aligned} \alpha_{s_1} &= \arg \max_{\alpha \in [0,1]} p(\mathbf{T} \mid s_1, t_0, \mu, \sigma, \alpha) \\ &= \arg \max_{\alpha \in [0,1]} \frac{\exp\left(-\frac{1}{2\sigma^2} \mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U}\right)}{\sqrt{(2\pi\sigma^2)^n \det \mathbf{A}_{s_1}(\alpha)}} \\ &= \arg \min_{\alpha \in [0,1]} f(\alpha) \triangleq \log \det \mathbf{A}_{s_1}(\alpha) + \frac{1}{\sigma^2} \mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U}, \end{aligned}$$

where $\mathbf{U} = \mathbf{T} - \mathbf{D}_{s_1} \beta_0$ and $\mathbf{A}_{s_1}(\alpha) = \alpha \mathbf{H}_{s_1} + (1 - \alpha) \mathbf{\Lambda}_{s_1}$ according to (5.6). Let $f_1(\alpha) = \log \det \mathbf{A}_{s_1}(\alpha)$ and $f_2(\alpha) = \mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U}$ so that $f(\alpha) = f_1(\alpha) + f_2(\alpha)/\sigma^2$. Our goal is to show that on average, $f(0)$ is close to a local optimum when G is a tree, since $\alpha_{\hat{s}} = 0$ makes GSSI equivalent to MLE.

Lemma 5.6.1. Suppose that G is a tree, and the GSSI estimates $(\hat{s}, [\hat{t}_0, \hat{\mu}]', \hat{\sigma}^2) = (s_1, \beta_0, \sigma^2)$. Then, we have

- (i) $f_1(\alpha)$ is concave and non-decreasing for $\alpha \in [0, 1]$.
- (ii) $f_2(\alpha)$ is convex w.r.t. α and $\left. \frac{d}{d\alpha} f_2(\alpha) \right|_{\alpha=0} \geq \mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U} (1 - \lambda_{\max}(\mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1}))$, where $\lambda_{\max}(\cdot)$ is the largest eigenvalue of its matrix argument.
- (iii) $\mathbb{E} \left[\left. \frac{d}{d\alpha} f(\alpha) \right|_{\alpha=0} \right] = 0$.

Proof.

- (i) The first and second derivatives of $f_1(\alpha)$ w.r.t. α are given by

$$\begin{aligned} \frac{d}{d\alpha} f_1(\alpha) &= \frac{1}{\det \mathbf{A}_{s_1}(\alpha)} \frac{d}{d\alpha} \det \mathbf{A}_{s_1}(\alpha) \\ &= \text{tr}(\mathbf{A}_{s_1}^{-1}(\alpha) \frac{d}{d\alpha} \mathbf{A}_{s_1}(\alpha)) \\ &= \text{tr}(\mathbf{A}_{s_1}^{-1}(\alpha) (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1})), \end{aligned} \tag{5.10}$$

and

$$\begin{aligned}
\frac{d^2}{d\alpha^2} f_1(\alpha) &= \text{tr}(-\mathbf{A}_{s_1}^{-1}(\alpha) \frac{d}{d\alpha} \mathbf{A}_{s_1}(\alpha) \mathbf{A}_{s_1}^{-1}(\alpha) \cdot (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1})) \\
&= \text{tr}(-\mathbf{A}_{s_1}^{-1}(\alpha) (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1}) \mathbf{A}_{s_1}^{-1}(\alpha) \cdot (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1})) \\
&\leq 0.
\end{aligned}$$

Hence $f_1(\alpha)$ is concave. We also have

$$\begin{aligned}
\left. \frac{d}{d\alpha} f_1(\alpha) \right|_{\alpha=1} &= \text{tr}(\mathbf{H}_{s_1}^{-1}(\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1})) \\
&= n - \text{tr}(\mathbf{H}_{s_1}^{-1} \mathbf{\Lambda}_{s_1}) \\
&= n - \sum_{i=1}^n [\mathbf{\Lambda}_{s_1}]_{i,i} / [\mathbf{H}_{s_1}]_{i,i} \\
&= 0,
\end{aligned}$$

therefore $\frac{d}{d\alpha} f_1(\alpha) \geq 0$ for all $\alpha \in [0, 1]$.

(ii) We have

$$\begin{aligned}
\frac{d}{d\alpha} f_2(\alpha) &= -\mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) \frac{d}{d\alpha} \mathbf{A}_{s_1}(\alpha) \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U} \\
&= -\mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1}) \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U}, \tag{5.11}
\end{aligned}$$

and

$$\begin{aligned}
\frac{d^2}{d\alpha^2} f_2(\alpha) &= 2\mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) \frac{d}{d\alpha} \mathbf{A}_{s_1}(\alpha) \mathbf{A}_{s_1}^{-1}(\alpha) \cdot (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1}) \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U} \\
&= 2[\mathbf{U}' \mathbf{A}_{s_1}^{-1}(\alpha) (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1})] \mathbf{A}_{s_1}^{-1}(\alpha) \cdot [(\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1}) \mathbf{A}_{s_1}^{-1}(\alpha) \mathbf{U}] \\
&\geq 0,
\end{aligned}$$

where the last inequality holds because $\mathbf{A}_{s_1}(\alpha)$ is positive semidefinite. Therefore, $f_2(\alpha)$ is convex.

To show the lower bound, we have

$$\left. \frac{d}{d\alpha} f_2(\alpha) \right|_{\alpha=0} = \mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U} - \mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1} \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}.$$

Let $\mathbf{W} = \mathbf{H}_{s_1}^{\frac{1}{2}} \mathbf{\Lambda}_{s_1}^{-\frac{1}{2}}$ and $\mathbf{u} = \mathbf{\Lambda}_{s_1}^{-\frac{1}{2}} \mathbf{U}$. Let $\sigma_{\max}(\mathbf{W})$ be the spectral norm of $\mathbf{W}$, then we obtain from Theorem 5.2.7 of [118],

$$\begin{aligned} \sigma_{\max}^2(\mathbf{W}) &= \sup_{\mathbf{x} \neq \mathbf{0}} \frac{\mathbf{x}' \mathbf{W}' \mathbf{W} \mathbf{x}}{\mathbf{x}' \mathbf{x}} \\ &\geq \frac{\mathbf{u}' \mathbf{W}' \mathbf{W} \mathbf{u}}{\mathbf{u}' \mathbf{u}} \\ &= \frac{\mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1} \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}}{\mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}}, \end{aligned}$$

and $\mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1} \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U} \leq \sigma_{\max}^2(\mathbf{H}_{s_1}^{\frac{1}{2}} \mathbf{\Lambda}_{s_1}^{-\frac{1}{2}}) \mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}$. Since $\sigma_{\max}^2(\mathbf{W}) = \lambda_{\max}(\mathbf{W} \mathbf{W}')$, we have

$$\left. \frac{d}{d\alpha} f_2(\alpha) \right|_{\alpha=0} \geq \mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U} (1 - \lambda_{\max}(\mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1})),$$

since $\mathbf{H}_{s_1}^{\frac{1}{2}} \mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1}^{\frac{1}{2}}$ and $\mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1}$ have the same eigenvalues.

- (iii) From (5.2), we have $\mathbf{T} \sim \mathcal{N}(\mathbf{D}_{s_1} \beta_0, \sigma^2 \mathbf{\Lambda}_{s_1})$ so $\frac{1}{\sigma} \mathbf{\Lambda}_{s_1}^{-\frac{1}{2}} \mathbf{U} \sim \mathcal{N}(0, \mathbf{I}_n)$, and $\frac{1}{\sigma^2} \mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}$ follows a chi-squared distribution with n degrees of freedom. From (5.10) and (5.11), we obtain

$$\begin{aligned} \mathbb{E} \left[\left. \frac{d}{d\alpha} f(\alpha) \right|_{\alpha=0} \right] &= \text{tr}(\mathbf{\Lambda}_{s_1}^{-1} (\mathbf{H}_{s_1} - \mathbf{\Lambda}_{s_1})) + \frac{1}{\sigma^2} \mathbb{E} [\mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}] - \frac{1}{\sigma^2} \mathbb{E} [\mathbf{U}' \mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1} \mathbf{\Lambda}_{s_1}^{-1} \mathbf{U}] \\ &= \text{tr}(\mathbf{\Lambda}_{s_1}^{-1} \mathbf{H}_{s_1}) - n + n - \text{tr}(\mathbf{\Lambda}_{s_1}^{-\frac{1}{2}} \mathbf{H}_{s_1} \mathbf{\Lambda}_{s_1}^{-\frac{1}{2}}) \\ &= 0, \end{aligned}$$

and the proof is complete. □

Lemma 5.6.1(iii) says that on average the first derivative of $f(\alpha)$ is zero at $\alpha = 0$, which implies that α_{s_1} cannot be much larger than zero with high probability.

Chapter 6

Multiple Sources Estimation

In this chapter, we discuss the case of multiple misinformation sources, i.e., $|S| > 1$ but unknown. We first reduce the multiple sources estimation problem to the single source estimation problem, which is then solved by applying GSSI algorithm introduced in Chapter 5. We develop a reduction step for trees, and then extend it heuristically to general graphs.

6.1 The Reduction Step for Trees

In this section, we consider the case where G is a tree, and introduce the concepts of observation cluster and source candidate cluster, which allow us to reduce the multiple sources estimation problem to that of estimating a single source within each cluster. Since all $\{\tau_{uv}\}$ are positive continuous random variables, the observed timestamps are distinct with probability one. The following definitions are illustrated in Fig. 6.1.

Definition 6.1. A node $v_k \in \mathcal{V}$ is called *observable* w.r.t. $u \in V$ if the following condition holds: for each pair of distinct nodes $v_i, v_j \in \mathcal{V} \cap [u, v_k]$ such that $v_i \in [u, v_j)$, then $t_i < t_j$. The collection of observable nodes w.r.t. u is called its observation cluster, and denoted by $\mathcal{V}_u$.

Definition 6.2. Let $u \in V$. The collection of all nodes $v \in V$ such that $\mathcal{V}_v = \mathcal{V}_u$ is denoted by A_u , which is called the source candidate cluster w.r.t. $\mathcal{V}_u$.

FIG. 6.1: An example to illustrate the notions of observation cluster, source candidate cluster and the SCCE algorithm.

- (a) Red nodes are the sources, blue nodes are the observed nodes. Three misinformation sources $S = \{2, 9, 13\}$ initiate a diffusion process on the tree shown. Suppose we observe 6 infected nodes $\mathcal{V} = \{1, 4, 7, 10, 12, 14\}$ and their respective timestamps. For node 9, since t_4 is greater than t_1 , t_2 and t_5 , the observation cluster for node 9 is $\mathcal{V}_9 = \{7, 10\}$. Similarly, $\mathcal{V}_8 = \mathcal{V}_{10} = \{7, 10\}$, and for any node $u \notin \{8, 9, 10\}$, we have $\mathcal{V}_u \neq \{7, 10\}$. Therefore, the source candidate cluster w.r.t. $\mathcal{V}_9$ is $A_9 = \{8, 9, 10\}$.
- (b) In the MSR step of SCCE, we first set $\mathcal{U} = \mathcal{V}$, and obtain $\xi_1 = \arg \min_{v_k \in \mathcal{U}} t_k = 1$ so that $\mathcal{V}_{\xi_1} = \mathcal{V}_1 = \{1, 4, 7, 10\}$ and $A_{\xi_1} = A_1 = \{1, 2, 3\}$. We next set $\mathcal{U} = \mathcal{U} \setminus \mathcal{V}_{\xi_1} = \{12, 14\}$, and obtain $\xi_2 = \arg \min_{v_k \in \mathcal{U}} t_k = 14$ with $\mathcal{V}_{\xi_2} = \mathcal{V}_{14} = \{7, 10, 12, 14\}$ and $A_{\xi_2} = A_{14} = \{13, 14\}$. We then have $l = 2$ observation clusters, and the subtrees spanning $\mathcal{V}_{\xi_1}$ and $\mathcal{V}_{\xi_2}$ are shown in (b). Note that A_{ξ_1} and A_{ξ_2} each contains at least one source, and $l < |S| = 3$ since $\nu_9 = 10 \in \mathcal{V}_{13}$. This is consistent with Proposition 6.2.
- (c) In the SSSE step of SCCE, for each subtree in (b), we first find a maximum length path P , and define $s_P(w)$ to be the sum of the distances between w and leaves that are closest to w compared with other nodes in $P \setminus w$. For the left subtree in (b), we find $P = [1, 7]$ with the maximal length 6. We have $s_P(w) = 1$ when $w \in \{2, 3, 8\}$, and $s_P(w) = 0$ when $w \in \{9, 10\}$. Since $s_P(9)$ is a local minimum, we delete the edge connecting nodes 9 and 10 to obtain two subtrees as shown in the left and middle of (c). For the right tree in (b), we have $P = [7, 14]$, $s_P(w) = 1$ when $w \in \{8, 13\}$, and $s_P(w) = 0$ when $w \in \{9, 10, 12\}$. We delete the edge connecting nodes 9 and 10 and obtain two subtrees as shown in the middle and right of (c).

The observation cluster $\mathcal{V}_u$ of u is the set of observed infected nodes in $\mathcal{V}$ that can possibly be infected by u . However, $\mathcal{V}_u$ may also be the observation cluster of other nodes $v \neq u$. We collect all these potential source nodes into the source candidate cluster A_u . From Definitions 6.1 and 6.2, we have the following immediate observations. For a $v_i \in \mathcal{V}$, since $v_i \in \mathcal{V}_{v_i}$, the set $\mathcal{V}_{v_i}$ is non-empty. Similarly, $v_i \in A_{v_i}$. Let $\mathcal{I}_{s_m}$ be the collection of nodes in $\mathcal{V}$ infected by $s_m \in S$. Specifically, if there is only one source s , then $\mathcal{V}_s = \mathcal{I}_s = \mathcal{V}$. For any $u \in V$, let

$$\nu_u \triangleq \arg \min_{v_k \in \mathcal{V}_u} t_k. \quad (6.1)$$

We have the following results regarding $\mathcal{V}_u$, A_u , and $\mathcal{I}_{s_m}$.

Lemma 6.1. Consider any node $u \in V$.

- (i) If $v_k \in \mathcal{V}_u$, then for any $v_i \in [u, v_k) \cap \mathcal{V}$, we have $v_i \in \mathcal{V}_u$ and $t_i < t_k$.
- (ii) $A_u \cap \mathcal{V} = \{\nu_u\}$. Specifically, if $u = \nu_{s_m}$ for some $s_m \in S$, then $\mathcal{V}_u = \mathcal{V}_{s_m}$ (or $s_m \in A_u$).
- (iii) A_u is connected.
- (iv) For each $s_m \in S$, $\mathcal{I}_{s_m} \subset \mathcal{V}_{s_m}$.
- (v) If $v_k \in \mathcal{V}_u$, then $\mathcal{V}_{v_k} \subset \mathcal{V}_u$. Specifically, if $v_k \in \mathcal{I}_{s_m}$, then $\mathcal{V}_{v_k} \subset \mathcal{V}_{s_m}$.
- (vi) If $\nu_u \in \mathcal{V}_v$, then $\mathcal{V}_u \subset \mathcal{V}_v$.

Proof.

- (i) This follows directly from Definition 6.1.
- (ii) We first prove that for any $v_k \in \mathcal{V} \setminus \nu_u$ then $\mathcal{V}_u \neq \mathcal{V}_{v_k}$. If $v_k \notin \mathcal{V}_u$, then it follows directly that $\mathcal{V}_u \neq \mathcal{V}_{v_k}$. If $v_k \in \mathcal{V}_u \setminus \nu_u$, then $\nu_u \notin \mathcal{V}_{v_k}$ because $t_k > t'$, where t' is the timestamp of node ν_u . Since $\nu_u \in \mathcal{V}_u$, we have $\mathcal{V}_u \neq \mathcal{V}_{v_k}$. We then prove that $\mathcal{V}_u = \mathcal{V}_{\nu_u}$. If $u \in \mathcal{V}$, then $u = \nu_u$ and it follows directly that $\mathcal{V}_u = \mathcal{V}_{\nu_u}$. We next consider the case where $u \notin \mathcal{V}$. Suppose $v_k \in \mathcal{V}_u \setminus \nu_u$. According to claim (i), we find $v_i \in \mathcal{V}_u$ such that $v_i \in [u, v_k]$ and $[u, v_i] \cap \mathcal{V} = v_i$. By the minimality condition (6.1), $[u, \nu_u] \cap \mathcal{V} = \{\nu_u\}$. Then we have $[\nu_u, v_i] \cap \mathcal{V} = \{\nu_u, v_i\}$ and $t' < t_i$, thus $v_i \in \mathcal{V}_{\nu_u}$; and moreover

- $v_k \in \mathcal{V}_{\nu_u}$ from Definition 6.1. Suppose on the contrary $v_k \notin \mathcal{V}_u$. Then from Definition 6.1, there exists a pair of distinct nodes $v_i, v_j \in \mathcal{V} \cap [u, v_k]$ such that $v_i \in [u, v_j]$. As a consequence, $t_i > t_j$. Since $[u, \nu_u] \cap \mathcal{V} = \{\nu_u\}$, we have $v_i \in [\nu_u, v_j]$; and hence $v_k \notin \mathcal{V}_{\nu_u}$. Therefore $\mathcal{V}_u = \mathcal{V}_{\nu_u}$.
- (iii) According to claim (ii), it is equivalent to prove A_{ν_u} is connected. We now prove that for any $v_k \in \mathcal{V}$, A_{v_k} is connected. From claim (ii) we have $A_{v_k} \cap \mathcal{V} = \{v_k\}$. Therefore given $u \in A_{v_k} \setminus v_k$, we have $u \notin \mathcal{V}$. We show that $(v_k, u) \cap \mathcal{V} = \emptyset$. Suppose on the contrary that there exists $v_i \in \mathcal{V}$ and $v_i \in (v_k, u)$. According to claim (i), since $v_k \in \mathcal{V}_u$ then $v_i \in \mathcal{V}_u$ and $t_i < t_k$. Then we must have $v_i \notin \mathcal{V}_{v_k}$ which contradicts $\mathcal{V}_u = \mathcal{V}_{v_k}$. Therefore for any $v \in (v_k, u)$ we have $[u, v] \cap \mathcal{V} = \emptyset$, then it is easy to see that $\mathcal{V}_u = \mathcal{V}_v = \mathcal{V}_{v_k}$ thus $v \in A_{v_k}$. Therefore the claim of connectedness holds.
- (iv) If $v_k \in \mathcal{I}_{s_m}$, then we have $[s_m, v_k] \cap \mathcal{V} \subset \mathcal{I}_{s_m}$. As the propagation delay over each edge is positive, it is easy to verify that $v_k \in \mathcal{V}_{s_m}$ from Definition 6.1.
- (v) We prove that if $v_i \in \mathcal{V}_u$ and $v_j \in \mathcal{V}_{v_i}$, then $v_j \in \mathcal{V}_u$. From Definition 6.1, for each pair $v_a, v_b \in [u, v_i] \cap \mathcal{V}$ such that $v_a \in [u, v_b]$, we have $t_a \leq t_b \leq t_i$. If $v_j \in \mathcal{V}_{v_i}$ then for each pair $v_c, v_d \in [v_i, v_j] \cap \mathcal{V}$ such that $v_c \in [v_i, v_d]$, we have $t_i \leq t_c \leq t_d$. So we have $[u, v_i] \cap [v_i, v_j] \cap \mathcal{V} = v_i$, and then for each pair $v_p, v_q \in [u, v_j] \cap \mathcal{V}$ such that $v_p \in [u, v_q]$, we have $t_p \leq t_q$. This implies that $v_j \in \mathcal{V}_u$. Specifically, if $v_k \in \mathcal{I}_{s_m}$, then $v_k \in \mathcal{V}_{s_m}$ thus $\mathcal{V}_{v_k} \subset \mathcal{V}_{s_m}$.
- (vi) According to claim (ii) and claim (v), if $w = \nu_u \in \mathcal{V}_v$, then $\mathcal{V}_u = \mathcal{V}_w \subset \mathcal{V}_v$. Specifically, if $\nu_{s_i} \in \mathcal{V}_{s_j}$, then $\mathcal{V}_{s_i} \subset \mathcal{V}_{s_j}$.

□

From Lemma 6.1(iv), we see that the set of observed nodes $\mathcal{I}_{s_m}$ infected by s_m , is a subset of $\mathcal{V}_{s_m}$. Therefore $\mathcal{V}_{s_m}$ can be used as an approximation of $\mathcal{I}_{s_m}$. We provide a procedure to approximate $\{\mathcal{V}_{s_m} : s_m \in S\}$ in the following.

For a given $v_i \in \mathcal{V}$, our aim is to develop a method to efficiently generate A_{v_i} . Starting from the root v_i , we perform BFS, with the search path stopping immediately before hitting any node in $\mathcal{V} \setminus \{v_i\}$. Let $B(v_i, \mathcal{V})$ be the collection of discovered nodes in this tree.

Algorithm 6.1 Multiple Sources Reduction (MSR)

Input: Adjacency matrix of the tree G , observations $\mathcal{V} = \{v_1, \dots, v_n\}$ and $\mathbf{T} = [t_1, \dots, t_n]'$.

Output: $\{\mathcal{V}_{\xi_i}\}_{i=1}^l$ and $\{A_{\xi_i}\}_{i=1}^l$.

- 1: Let $\mathcal{U} = \mathcal{V}$, and $i = 1$.
- 2: **while** $\mathcal{U} \neq \emptyset$ **do**
- 3: $\xi_i = \arg \min_{v_k \in \mathcal{U}} t_k$.
- 4: Find $\mathcal{V}_{\xi_i}$ according to Definition 6.1, then find $A_{\xi_i} = B(\xi_i, \mathcal{V})$ (cf. Proposition 6.2(i)).
- 5: Set $l = i$, $i = i + 1$, $\mathcal{U} = \mathcal{U} \setminus \mathcal{V}_{\xi_i}$.
- 6: **end while**

Proposition 6.1. For any $v_i \in \mathcal{V}$, $A_{v_i} \subset B(v_i, \mathcal{V})$. If $v_i \notin \mathcal{V}_{v_j}$ for any $v_j \in \mathcal{V} \setminus \{v_i\}$, then $A_{v_i} = B(v_i, \mathcal{V})$.

Proof. According to the proof of Lemma 6.1(iii), it is easy to see that if $u \in A_{v_i}$, then $u \in B(v_i, \mathcal{V})$. Given the condition $v_i \notin \mathcal{V}_{v_j}$ for any $v_j \in \mathcal{V} \setminus \{v_i\}$, we prove that if $u \in B(v_i, \mathcal{V})$ then $u \in \mathcal{V}_{v_i}$. According to Lemma 6.1(ii), it is equivalent to prove $\nu_u = v_i$. Suppose on the contrary $\nu_u = v_k$ and $v_k \neq v_i$. Then $[u, v_k] \cap \mathcal{V} = v_k$. Since $u \in B(v_i, \mathcal{V})$ then $[u, v_i] \cap \mathcal{V} = v_i$ and $v_i \in \mathcal{V}_u$. So we have $[v_i, v_k] \cap \mathcal{V} = \{v_i, v_k\}$ and $t_k < t_i$. Therefore $v_i \in \mathcal{V}_{v_k}$; and we obtain a contradiction. Then we must have $\nu_u = v_i$ and $u \in \mathcal{V}_{v_i}$, which completes the proof. $\square$

We now introduce the notion of an admissible covering of $\mathcal{V}$.

Definition 6.3. A covering that decomposes $\mathcal{V} = \bigcup_{i=1}^l \mathcal{U}_i$ into a union (not necessarily disjoint) of non-empty subsets $\mathcal{U}_i$ for $1 \leq i \leq l$, is called *admissible* if for each i , and there exists u_i such that $\mathcal{V}_{u_i} = \mathcal{U}_i$.

It is easy to see that $\{\mathcal{V}_{s_m}\}_{m=1}^{|S|}$ is an admissible covering of $\mathcal{V}$. However, since the sources are unknown, we propose the Multiple Sources Reduction (MSR) algorithm in Algorithm 6.1 to find an admissible covering. We have the following result.

Proposition 6.2. Consider the the MSR algorithm in Algorithm 6.1.

- (i) For each $i = 1, \dots, l$, $A_{\xi_i} = B(\xi_i, \mathcal{V})$.
- (ii) $\{\mathcal{V}_{\xi_i}\}_{i=1}^l$ is an admissible covering of $\mathcal{V}$.

- (iii) Each A_{ξ_i} contains at least one source, so $l \leq |S|$. If $l = |S|$, then each A_{ξ_i} contains exactly one source.
- (iv) $l = |S|$ if and only if for each distinct pair $s_i, s_j \in S$, $\nu_{s_i} \notin \mathcal{V}_{s_j}$.

Proof.

- (i) Since $\xi_i = \arg \min_{v_k \in \mathcal{U}} t_k$, for any $v_j \in \mathcal{U} \setminus \{\xi_i\}$, we have $\xi_i \notin \mathcal{V}_{v_j}$. Suppose there exists $v_j \in \mathcal{V} \setminus \mathcal{U}$ such that $\xi_i \in \mathcal{V}_{v_j}$. Since $v_j \in \mathcal{V}_{\xi_k}$ for some $k < i$, from Lemma 6.1(v), $\mathcal{V}_{v_j} \subset \mathcal{V}_{\xi_k}$ and we obtain a contradiction with $\xi_i \notin \mathcal{U}$. The claim then follows from Proposition 6.1.
- (ii) It follows directly that each $\mathcal{V}_{\xi_i}$ is non-empty and $\mathcal{V} = \bigcup_{i=1}^l \mathcal{V}_{\xi_i}$ with $l \leq |\mathcal{V}|$.
- (iii) If $\xi_1 \in \mathcal{V}_{s_1}$, then $\xi_1 = \nu_{s_1}$. According to Lemma 6.1(ii), $s_1 \in A_{\xi_1}$. If $\mathcal{U} \neq \emptyset$ and suppose $\xi_i \in \mathcal{V}_{s_i}$, then from Lemma 6.1(vi), we must have $\nu_{s_i} \notin \bigcup_{j=1}^{i-1} \mathcal{V}_{\xi_j}$. Therefore $\nu_{s_i} = \xi_i$ and $s_i \in A_{\xi_i}$. This procedure is then repeated and each A_{ξ_i} contains at least one source. Moreover, it is likely that some sources are not contained in any A_{ξ_i} . Refer to the example in Fig. 6.2. So $l \leq |S|$. If $l = |S|$, then each A_{ξ_i} contains exactly one source.
- (iv) By the proof of claim (iii), if for each distinct pair $s_i, s_j \in S$ such that $\nu_{s_i} \notin \mathcal{V}_{s_j}$, then we can find $\{\xi_m\}_{m=1}^{|S|}$ with $\xi_m = \nu_{s_m}$. On the contrary, if there exists $\nu_{s_i} \in \mathcal{V}_{s_j}$ then $\nu_{s_j} < \nu_{s_i}$, so we must choose ν_{s_j} before ν_{s_i} . From Lemma 6.1(vi), $\mathcal{V}_{s_i} \subset \mathcal{V}_{s_j}$ thus $\mathcal{V}_{s_i} \cap \mathcal{U} \setminus \mathcal{V}_{s_j} = \emptyset$. Hence $s_i \in A_{\nu_{s_j}}$ (cf. Fig. 6.2(a)) or s_i will not be contained at any $\{A_{\xi_k}\}_{k=1}^l$ (cf. Fig. 6.2(b)). This implies that $l < |S|$.

□

Finding ξ_i in Algorithm 6.1 can be accomplished by maintaining a min-heap for $\mathbf{T}$, which incurs an overall time complexity of $O(|\mathcal{V}|)$ for the whole algorithm. Finding A_{ξ_i} involves BFS, which incurs a time complexity of $O(|V \setminus \bigcup_{k < i} A_{\xi_k}|)$. Therefore, the overall time complexity of MSR is $O(|V|)$.

Proposition 6.2(iii) indicates that if we assume that there is only a single source in each source candidate cluster A_{ξ_i} , then we are underestimating the number of

FIG. 6.2: Two cases where $l < |S|$. The observed infected nodes are labeled with their infection timestamps, and $|S| = 2$. Since $\xi_1 = v_1 = \nu_{s_1}$ and $\mathcal{V}_{\xi_1} = \mathcal{V}$ then we have $l = 1 < |S| = 2$ and $\nu_{s_2} \in \mathcal{V}_{s_1}$. In (a), $A_{\xi_1} = \{s_1, s_2, v_1\}$. In (b), $A_{\xi_1} = \{s_1, v_1\}$.

Algorithm 6.2 Source-Set Size Estimation (SSSE)[119]

Input: Adjacency matrix of the tree T .

Output: A set of disjoint and connected subtrees of T , denoted by $\{T_i\}_{i=1}^K$.

- 1: Compute the average pairwise distance $\bar{d}$ of T , and find a path $P = [u, v]$ with the maximal length.
 - 2: **if** $|P| > \bar{d}$ **then**
 - 3: Define a function $s_P(\cdot)$ on P : For each $w \in P$, find the leaves (with degree 1) in T that are closest to w compared with other nodes in $P \setminus w$. Then $s_P(w)$ is defined to be the sum of the distances between w and each of these leaves. If $s_P(w)$ is a local minimum and w' is a neighbor of w in P , then delete the edge connecting w and w' . T is divided into two subtrees T_1 and T_2 .
 - 4: **end if**
 - 5: For T_1 and T_2 , repeat the above partition procedure until the diameters of the subtrees are less than $\bar{d}$ or $s_P(\cdot)$ does not have a local minimum.
-

sources (see Fig. 6.2 for an example). Therefore we adopt an additional procedure developed in [119] to further partition each A_{ξ_i} and $\mathcal{V}_{\xi_i}$. The paper [119] has proposed an algorithm, which we call the Source-Set Size Estimation (SSSE) algorithm, to estimate the number of misinformation sources in a tree by partitioning the tree into several covering components. We briefly summarize the SSSE algorithm in Algorithm 6.2.

The time complexity of SSSE is $O(|V|)$. From Lemma 6.1(iii) we see that each A_{ξ_i} forms a connected subtree. So we apply the SSSE algorithm on each A_{ξ_i} to obtain a new covering for $\mathcal{V}$, which are denoted by $\{C_i\}_{i=1}^L$, where $L \geq l$. At the same time $\{\mathcal{V}_{\xi_i}\}_{i=1}^l$ is also partitioned as $\{O_i\}_{i=1}^L$. Our simulations indicate that it is now possible for $L > |S|$ as SSSE may over-estimate the number of sources in each A_{ξ_i} .

Finally, for each $1 \leq i \leq L$, we apply GSSI to find a source estimate $\hat{s}_i \in C_i$ using O_i as an approximation for $\mathcal{I}_{s_i}$. Note that different sources may have different diffusion parameters (t_0, μ, σ^2) and GSSI is able to estimate these parameters for

Algorithm 6.3 Source Candidate Clustering and Estimation (SCCE)

Input: Adjacency matrix of the graph G , observation $\mathcal{V} = \{v_1, \dots, v_n\}$ and $\mathbf{T} = [t_1, \dots, t_n]'$.

Output: $\hat{S} = \{\hat{s}_i\}_{i=1}^L$

- 1: Let $\mathcal{U} = \mathcal{V}$, and $i = 1$
- 2: **while** $\mathcal{U} \neq \emptyset$ **do**
- 3: $\xi_i = \arg \min_{v_k \in \mathcal{U}} t_k$.
- 4: Find one BFS tree rooted at ξ_i and then obtain $B(\xi_i, \mathcal{V})$ and $\mathcal{V}_{\xi_i}$ on this tree.
Set $A_{\xi_i} = B(\xi_i, \mathcal{V})$.
- 5: Set $l = i$, $i = i + 1$, $\mathcal{U} = \mathcal{U} \setminus \mathcal{V}_{\xi_i}$.
- 6: **end while**
- 7: For each $1 \leq i \leq l$, run SSSE algorithm to partition $\mathcal{V}_{\xi_i}$ and A_{ξ_i} on the BFS tree rooted at ξ_i . Finally we obtain $\{C_i\}_{i=1}^L$ and $\{O_i\}_{i=1}^L$.
- 8: **for** every C_i and O_i **do**
- 9: Run GSSI algorithm. Compute f_s in Algorithm 5.1 for each $s \in C_i \setminus \{\xi_i\}$.
- 10: Pick $\hat{s}_i$ to be a node s that minimizes f_s .
- 11: **end for**

each possible source. An example to illustrate the steps involved in the estimation of multiple sources on a tree is given in Fig. 6.1.

6.2 Multiple Sources Estimation for General Graphs

We now heuristically extend the sources estimation procedure for trees described in the previous section to general graphs, and summarized it in Algorithm 6.3, where $\{\hat{s}_i\}_{i=1}^L$ is the set of estimated sources. First, we make use of the BFS heuristic in generalizing the MSR algorithm to a general graph before applying SSSE. Notice that for general graphs, it is possible that before applying the SSSE algorithm, the number of source candidate clusters $l > |S|$ and $A_{\xi_i} \cap S = \emptyset$ for some values of i . The generalized MSR and SSSE steps involve a time complexity of $O(|V|^2)$ since we find BFS trees rooted at each ξ_i . For each subgraph we run GSSI, whose time complexity depends on the number of nodes and observed nodes of the subgraph. The worst case complexity is incurred when $L = 1$. Therefore the overall time complexity of SCCE is $O(|V|^3 + |V||\mathcal{V}|^3)$.

6.3 Experimental Evaluations for Multiple Sources Estimation

We evaluate the performance of our SCCE algorithm with the same graphs used in Section 5.5. We compare SCCE with the method proposed in [57], which we call the BDSL algorithm. To quantify the performance of SCCE and BDSL, we adopt the performance metric (with a slight modification for fairer comparison) proposed in [36]: We first match the estimated sources $\{\hat{s}_i\}_{i=1}^L$ with the actual sources $\{s_i\}_{i=1}^{|S|}$ so that the sum of the error distances between each estimated source and its match is minimized. Denote this matching as π . If we incorrectly estimate the number of misinformation sources, i.e., $L \neq |S|$, we add a penalty $\eta||S|-L|$. Denote $d_{\max}$ and $\bar{d}$ as the diameter and average pairwise distance of the graph. Since the distance between a source and a random node does not exceed $d_{\max}$, a reasonable choice of penalty $\eta \leq d_{\max}$. We then define the average error to be

$$\Delta = \frac{1}{\mathcal{L}} \left(\sum_{i=1}^{\min(L, |S|)} d(s_i, \hat{s}_{\pi(i)}) + \eta||S|-L| \right), \quad (6.2)$$

where $d(u, v)$ is the length of the shortest path in G between u and v , and

$$\mathcal{L} = \begin{cases} \min(L, |S|) & \text{if } \eta = 0, \\ \max(L, |S|) & \text{if } 0 < \eta \leq d_{\max}. \end{cases}$$

For different applications, we may assign different values to η depending on how important it is to estimate correctly the number of misinformation sources. In this chapter, we consider $\eta = 0, \bar{d}$ and the extreme case where $\eta = d_{\max}$. The BDSL algorithm cannot estimate the number of sources, so we assume it has prior knowledge of $|S|$. Therefore for BDSL, we have $\mathcal{L} = L = |S|$.

We first perform simulations on ER(500) and BA(500). For each simulation, we let the number of sources be uniformly chosen from $\{2, 3, 4\}$. From Proposition 6.2(iv), we see that if two sources are close to each other, then it is likely that one of them cannot be identified. So we choose sources such that the distance between each pair of sources is no less than the average pairwise distance of the tree, which is a

reasonable assumption in real applications.¹ For the diffusion process, let the start times of all sources be identical, which is an assumption required by BDSL, and propagation delays along all edges follow a truncated Gaussian distribution $\mathcal{N}(2, 1)$. Notice that BDSL requires to know the mean of the Gaussian distribution used while our algorithm does not. For the BDSL algorithm, since we uniformly choose $|S|$ from $\{2, 3, 4\}$, the average L is 3, which is used as a benchmark to compare how well SCCE performs in estimating the number of sources. We randomly choose a portion of nodes as observed nodes for the multiple sources estimation. We average Δ in (6.2), and L over 300 simulations and plot against the fraction of timestamps as shown in Fig. 6.3. The plots in Fig. 6.3(a) and (b) show that SCCE performs better than BDSL for both B-A and E-R trees although SCCE requires less information. For SCCE algorithm, from Fig. 6.3(c) we see that $L > |S|$ in some cases due to the introduction of the additional partition procedure developed in the SSSE algorithm; and the average L is close to the average $|S|$ if we observe enough timestamps.

We next perform simulations on some synthetic graphs and real networks (cf. Table 5.1 for some properties of these graphs). Let the number of sources be uniformly chosen from $\{2, 3\}$, so that its average value is 2.5. For the truncated Gaussian distribution used to simulate the misinformation spreading, we set $(\mu, \sigma^2) = (3, 1)$. Since in real applications, $|S|$ is typically not very large, we make the additional assumption that $|S| \leq 5$ when running the SCCE algorithm by controlling the number of iterations in the MSR step. Simulation results in Figs. 6.4 to 6.6 indicate that in terms of average Δ , SCCE performs best in the cases where $\eta = 0$ and $\bar{d}$. In most cases SCCE also performs better than BDSL even when $\eta = d_{\max}$. For the performance of average L , we see that generally SCCE finds more source estimates as the fraction of timestamps increases. Sometimes we may over-estimate the number of sources. On E-R and B-A graphs, SCCE is able to find estimates that are on average within 2 hops of the real sources when the fraction of timestamps is at least 20%. For the Enron and Facebook networks, BDSL has poor estimation accuracy, while in the case of $\eta \leq \bar{d}$, SCCE produce estimates that are on average within 2.5 hops of the real sources with no less than 30% of nodes being observed.

¹Note that no estimation method can find all the sources correctly with probability one. Therefore, in real applications, source estimation methods are used to narrow down the potential candidate sources so that further investigation on them and their neighbors within a small radius can be conducted. Sources that are close to each other can then be considered as a single source under our estimation framework.

FIG. 6.3: Multiple sources estimation on (a) BA(500) and (b) ER(500).

FIG. 6.4: Multiple sources estimation on (a) BA(500, 16) and (b) ER(500, 16).

We also apply our algorithm to estimate multiple sources in malware propagation using the model and parameters discussed in Section 5.5. We see that the propagation model of malware validates our SI and Gaussian assumption. However, we want to see whether our algorithm can be used on this application. We randomly

FIG. 6.5: Multiple sources estimation on (a) Enron network and (b) Facebook network.

FIG. 6.6: Performance of the average L for different graphs.

and uniformly choose 2 or 3 malware sources and collect infection timestamps. Simulation results are shown in Fig. 6.7. The plots show that BDSL performs poorly even with increasing fraction of timestamps; we believe this is because the estimation of the mean propagation delays are not accurate. For SCCE, if the penalty η does not exceed $\bar{d}$, we are able to estimate sources that have an average error of 2.5 even with only 10% timestamps. The performance does not improve much as the fraction of timestamps increases. We believe this is because the diffusion model we have assumed does not match the characteristics of malware propagation.

In previous simulations, the number of sources is set to be relatively small (no greater than 3 on average). Intuitively, the performance of our SCCE algorithm will get worse as the number of sources increases. From Proposition 6.2(iv) we see that if several sources are close to each other, then it is likely that some of them cannot be identified. Therefore, we are more likely to underestimate $|S|$ if

FIG. 6.7: Malware sources estimation on Facebook network.

FIG. 6.8: Performance of average Δ and average L for different $|S|$ on B-A tree BA(500, 16) and B-A graph BA(500).

there are more sources in the network. Another reason can be that, on average the number of observations used for locating each source is also decreased. We perform simulations on BA(500) and BA(500, 16). Let $(\mu, \sigma^2) = (3, 1)$, $\eta = \bar{d}$, and the fraction of timestamps be 30%. We average Δ in (6.2), and L over 300 simulations and plot against the number of sources as shown in Fig. 6.8. The average pairwise distances of BA(500) and BA(500, 16) are 6.4 and 2.46 respectively. The plots show that the performance of SCCE is degraded significantly when the number of sources becomes large.

Finally, we evaluate the performance of SCCE with a sample of real data provided by SNAP.² First developed in the work [120], the Higgs dataset has been built after monitoring the spreading processes on Twitter before, during and after the announcement of the discovery of a new particle with the features of the elusive

²<https://snap.stanford.edu/data/higgs-twitter.html>

Higgs boson on 4 July 2012. Before the announcement on 1 July 2012, there were some rumors about the discovery of a Higgs-like boson at Tevatron. Such rumors were posted on the Twitter social network and quickly spread among Twitter users around the world. The dataset provides a social network with 456,626 users and 14,855,842 connections. In addition, a retweet network including the users and timestamps of retweet events is also given. We extract a connected sub-network with 5,078 users and 26,449 connections. All the users in this sub-network posted or retweeted the rumors related to the Higgs boson discovery. We also collect 4,254 users' first retweet timestamps and the total spanned time is about 39 hours starting from 1 July 2012. We then model the sub-network as an undirected graph (cf. Table 5.1), regard the timestamps as our observations, and the first several users who posted the rumors independently as misinformation sources.

The diffusion of the retweets is heterogeneous and unknown to us, which means that our assumptions for the diffusion model are not satisfied. However, we still apply our SCCE algorithm to estimate the sources given the graph topology and timestamp observations. Algorithms such as GAU and BDSL require more information such as the mean of the propagation delays, which need to be learned from historical data. From the dataset we find three misinformation sources, let $S = \{s_1, s_2, s_3\}$. In the extracted graph we have $d(s_1, s_2) = 2$, $d(s_2, s_3) = 2$, and $d(s_1, s_3) = 3$. These three misinformation sources have different start times. We observe that s_1 first initiated the diffusion process, then after about 7 and 15 hours, s_2 and s_3 started their diffusion, respectively. We then randomly choose a fraction of timestamps to perform multiple sources estimation using SCCE. After obtaining the estimated sources, we compute the average Δ and average L to show its performance. To test the BDSL algorithm, we learn the mean time of a tweet spreading across an edge from 50,000 historical tweets, which yields an estimate of about 4 hours. We average over 50 experiments. The comparison between SCCE and BDSL is shown in Table 6.1. We see that SCCE performs well on this real data. Comparatively, BDSL has poorer accuracy since the start times of the sources differ significantly from each other, and the mean propagation delays are not accurately known. The average number of estimated sources L obtained by SCCE is 2.2 with 10% timestamps, 3.1 with 20% timestamps and 3.3 with 30% timestamps.

To summarize, with the same fraction of timestamps, the performance of multiple sources estimation is not as good as single source estimation. This makes sense as

Fraction of timestamps	Average Δ			
	SCCE ($\eta = 0$)	SCCE ($\eta = \bar{d}$)	SCCE ($\eta = d_{\max}$)	BDSL
10%	0.73	1.88	3.2	5.6
20%	1.10	1.72	1.92	5.3
30%	0.96	1.68	1.89	5.4

TABLE 6.1: Comparison between SCCE and BDSL on real data.

Graph	Error reduction(%)	
	$\eta = \bar{d}$	$\eta = d_{\max}$
ER(500)	61.2	46.9
BA(500)	54.4	40.4
ER(500, 16)	34.4	14.1
BA(500, 16)	27.8	5.2
Enron	47.8	18.1
Facebook	45.6	12.0
Facebook (Malware)	49.5	43.8
Twitter	68.9	65.0

TABLE 6.2: Percent of reduced average error distance of SCCE compared with BDSL.

the number of timestamps used for each source estimation is reduced. A summary comparison of SCCE (with $\eta = \bar{d}$ and $d_{\max}$) with BDSL when 30% timestamps are observed is provided in Table 6.2.

6.4 Conclusion

In Chapter 5 and this chapter, we have developed algorithms to estimate infection sources using a subset of timestamps for the cases where there is a single source, and where there are multiple sources and the number of sources is unknown. We adopt a Gaussian spreading model with unknown diffusion parameters, which are estimated as part of our inference algorithms. In the single source case, we introduce a new heuristic that optimizes over a parametrized family of Gromov matrices to more accurately identify the source in a general graph. In the multiple sources estimation problem, we develop theory that allows us to partition the observed infected nodes into observation clusters. We then apply our single source algorithm in each cluster. Experimental evaluations with synthetic and real-world data suggest that our approaches can find the infection sources to within a small number of hops from the true sources with a small number of timestamp observations.

Chapter 7

Fake News Detection

In this chapter we consider a special case of misinformation identification: automatic fake news detection based on the statement text and additional metadata.

7.1 Proposed model

7.1.1 Feature Extraction

Given a piece of news S_i , we first obtain its numerical representation: a feature vector x_i . We write

$$x_i = [a'_i, b'_i, c'_i, d'_i]' \quad (7.1)$$

where a_i , b_i , c_i and d_i represent features from the statement text, textual metadata, numerical vector, and miscellaneous information respectively. The main statement consisting of a sequence of words should be a commonly available data and plays the most important role in fake news detection, Therefore a_i is always required. Other components can be optional. For example, textual metadata may include the topic and speaker's information such as name, job and context, the historic credit of the speaker can be represented by a numerical vector, and d_i can be a mixture of sentiment features and semantic features such as Term Frequency-Inverse Document Frequency (TFIDF), Linguistic Inquiry and Word Count (LIWC).

To extract such features, we may obtain a_i using a fine-tuned BERT model. For b_i , we average the word embedding vectors of the words that compose the textual metadata. We perform feature scaling methods such as standardization and mean normalization on numerical data to obtain c_i . For d_i , we use sentiment analysis models such as VADER [121] and TextBlob¹ on the statement.

7.1.2 The Hybrid Learning Model

After collecting a training set of feature vectors $X = \{x_1, x_2, \dots, x_N\}$ and associated truthfulness labels $\{l_1, l_2, \dots, l_N \mid l_i \in \mathcal{L}\}$, we first learn a simple neural network which we call NN1 whose output is a vector of likelihood scores corresponding to truthfulness classes in $\mathcal{L}$ (see Fig. 7.2(a)). We apply a softmax layer at the end, so that NN1 returns the probabilities of each class in $\mathcal{L}$. When training NN1, we use the cross-entropy as the loss function to perform the backpropagation. Then given a new feature vector $x_k \notin X$, we have

$$y_k = \text{NN1}(x_k). \quad (7.2)$$

Then the truthfulness label can be simply estimated by

$$\hat{l}_k = \arg \max_{l_k \in \mathcal{L}} y_k(l_k). \quad (7.3)$$

NN1 allows us to make prediction based on the content of each individual news. However, we also would like to exploit co-relations among different news. In order to do so, we propose to train another neural network, which we now describe as NN2. We need the following notion.

Definition 7.1. Let $\mathcal{R} = \{1, 2, \dots, R\}$ be a discrete set of classes. A *label mapping function* (see Fig. 7.1 for an example) is a symmetric function:

$$\phi : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{R}. \quad (7.4)$$

We would like to learn a graph using the labeled data. The label mapping function allows us to obtain a “label of the edge weight” given two labeled news.

¹<https://textblob.readthedocs.io/>

FIG. 7.1: An example to illustrate the label-mapping function in (7.4). Black numbers are truthfulness labels while red numbers are relationship labels.

FIG. 7.2: Two neural networks used in our model. NN1 is used to model the truthfulness of a piece of news, NN2 models the relationship between a pair of news.

More specifically, we group the training set X into news pairs $\{(x_i, x_j)\}$. For each pair (x_i, x_j) , we associated the *relationship labels* $\{r_{ij} = \phi(l_i, l_j)\}$, so that we can build-up a neural network NN2 to learn the relationship label of any pair of news. The architecture of NN2 is summarized in Fig. 7.2(b). Note that (x_i, x_j) is mapped to a vector combining x_i and x_j .

We make a few remarks regarding NN2:

1. The symmetry of ϕ guarantees $r_{ij} = r_{ji}$ for the training set. However, the symmetry may break on the testing set when we make predictions.
2. The size of training set of NN2 is of the order N^2 which can be large. We may extract a subset for training. One example is that we pair two pieces of news only if their topics are relevant.

3. The distribution of relationship classes can be imbalanced even when the distribution of truthfulness labels is balanced. From Fig. 7.1 we see that the relationship label ratio is 2 : 1 : 1 : 2. For this, we propose to resample the training dataset or use the weighted cross-entropy loss function when training NN2.

The design of NN2 is based on the intuition that if we know the relationship of a pair of news and the truthfulness of one of them, we are able to estimate the truthfulness of the other. Suppose we want to estimate the truthfulness of a piece of news, which has a collection of “neighbors”. Each neighbor represents a piece of related news with known truthfulness and can provide useful information to our estimation. Therefore, we are likely to give more accurate estimation by considering all the information provided by such neighbors. For example, each neighbor provides an estimation of the truthfulness based on its own truthfulness and relationship with the news. We can then use mechanisms such as majority voting to give the final estimation. Specifically, suppose l_i and r_{ij} are given, we can estimate l_j by:

$$\begin{aligned}
 P(l_j \mid l_i, r_{ij}) &\propto P(l_i, l_j, r_{ij}) \\
 &\propto P(r_{ij} \mid l_i, l_j) P(l_j \mid l_i) \\
 &\propto P(r_{ij} \mid l_i, l_j) \quad \text{if } P(l_j \mid l_i) = 1/L \text{ for any } l_j \in \mathcal{L}.
 \end{aligned} \tag{7.5}$$

From (7.4) we see that $P(r_{ij} \mid l_i, l_j)$ is either 0 or 1. However, it is likely that $P(l_j \mid l_i, r_{ij}) = P(l_k \mid l_i, r_{ij}) \neq 0$ for some $l_j \neq l_k$. See Table 7.1 for an example of computing $P(l_j \mid l_i, r_{ij})$ given the label-mapping function in Fig. 7.1.

Now given $x_k \notin X$, we first obtain

$$\hat{r}_{ik} = \text{NN2}(x_i, x_k) \tag{7.6}$$

which is similar for $\hat{r}_{ki}$ for any $x_i \in X$. We then compute another likelihood vector z_k , the probability of truthfulness label $l_k \in \mathcal{L}$ is given by

$$z_k(l_k) \propto \sum_{x_i \in X} (P(l_k \mid l_i, \hat{r}_{ik}) + P(l_k \mid l_i, \hat{r}_{ki})). \tag{7.7}$$

The vector z_k is normalized such that $\sum_{l_k \in \mathcal{L}} z_k(l_k) = 1$. We have actually constructed a star graph whose internal node is x_k and the set of leaves is X . The

		r_{ij}			
l_i	$P(l_j \mid l_i, r_{ij})$	1	2	3	4
	1	[1, 0, 0, 0]	[0, 1, 0, 0]	[0, 0, 1, 0]	[0, 0, 0, 1]
	2	[0, 1, 0, 0]	[1, 0, 0, 0]	[0, 0, 0, 1]	[0, 0, 1, 0]
	3	[0, 0, 1, 0]	[0, 0, 0, 1]	[1, 0, 0, 0]	[0, 1, 0, 0]
	4	[0, 0, 0, 1]	[0, 0, 1, 0]	[0, 1, 0, 0]	[1, 0, 0, 0]

TABLE 7.1: An example to compute $P(l_j \mid l_i, r_{ij})$ where $l_i, l_j \in \mathcal{L}$ and $r_{ij} \in \mathcal{R}$ given the label-mapping function in Fig. 7.1.

weight of a directed edge (x_k, x_i) is $\hat{r}_{ki}$, which represents the relationship between x_k and x_i . We regard the truthfulness label l_i as the graph signal on node x_i and we intend to recover the graph signal on x_k based on the knowledge of the graph topology and known graph signals on nodes on X . The concept of smoothness is often used to solve such graph signal recovery problem [28, 122]. In [28, 122], each edge weight is positive and represents the similarity between the graph signals on two adjacent nodes. Therefore, we can recover unknown graph signals by minimizing graph smoothness defined in (2.3). In (7.7), we borrow the similar idea. However, in our problem, graph signal values and edge weights are categorical variables. We instead define $z_k(l_k)$ in (7.7) as the representation of smoothness. We then aim to recover unknown graph signals by maximizing $z_k(l_k)$.

Therefore, we obtain two likelihood vectors y_k and z_k which are then combined to predict the final truthfulness label:

$$\hat{l}_k = \arg \max_{l_k \in \mathcal{L}} (z_k(l_k) + \lambda y_k(l_k)) \quad (7.8)$$

where λ is a hyperparameter which should be tuned on the development set. See Fig. 7.3 for a summary of our hybrid model. We expect that our hybrid model (7.8) can make label prediction for each news based on its own features as well as its related news with known labels.

FIG. 7.3: Our hybrid fake News detection model, which combines a deep learning model and a graph model. Note that X is the training set while $x_k \notin X$ is a testing sample.

Dataset Division			Speaker Affiliations		
Training set	Validation set	Testing set	Democrats	Democrats	Others
10,269	1,284	1,283	4,150	5,687	2,185
Labels Size					
Pants on fire	False	Barely-true	Half-true	Mostly-true	True
1,050	2,511	2,108	2,638	2,466	2,063

TABLE 7.2: The LIAR dataset statistics.

7.2 Performance Evaluation

7.2.1 Dataset

We evaluate our model using the LIAR dataset [65]. The dataset contains 12,836 news collected from POLITIFACT.COM. There are six fine-grained truthfulness labels take discrete values from 1 to 6 corresponding to *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*. Each news encompasses a short statement and auxiliary metadata. Metadata includes topic and speaker profile such as speaker name, job title, party affiliation, state, speech location, and credit history. The credit history is 5-dimensional vector representing a speaker’s historical records of inaccurate statements on label classes (5, 4, 3, 2, 1). Annotation is based on evaluations by professional editors. Some examples in the dataset are presented in Table 7.3. The statistics are listed in the Table 7.2. Although the statements in the dataset are politics-related only, they are not merely from democrats and republicans, but also from other non-politic individuals or organizations. The distribution of labels in the LIAR dataset is relatively well-balanced.

Statement: McCain opposed a requirement that the government buy American-made motorcycles. And he said all buy-American provisions were quote 'disgraceful.'
Topic: Federal budget
Speaker: Barack Obama
Job Title: President
State: Illinois
Party Affiliation: Democrats
Speech Location: A radio advertisement
Credit History: (70, 71, 160, 163, 9)
Label: True

Statement: When Mitt Romney was governor of Massachusetts, we didn't just slow the rate of growth of our government, we actually cut it.
Topic: History, state-budget
Speaker: Mitt Romney
Job Title: Former governor
State: Massachusetts
Party Affiliation: Republican
Speech Location: An interview with CBN News
Credit History: (34, 32, 58, 33, 19)
Label: False

TABLE 7.3: Two samples from the LIAR dataset.

7.2.2 Experiment Settings and Results

We first obtain the feature vector of a piece of news as discussed in Section 7.1.1. For the statement text, we apply the pre-trained BERT_{BASE} model [94] which has 12 layers (i.e. Transformer blocks). The dimension of the final hidden vector is 748. For the fine-tuning, we adopt the hyperparameters suggested in [94]. The batch size is 64, learning rate is $2e - 3$ and the number of epochs is 3. For textual metadata, we apply the Word2Vec tool [90] and average word embedding vectors of all the tokenized texts to obtain a 300-dimensional feature vector. For credit history, we perform standardization over the whole dataset to redistribute the features with zero mean normal distribution. We extract the sentiment feature of statement using VADER [121] and TextBlod². From [66, 67, 69] we learn that credit history greatly contributes to a successful prediction. Therefore we first test the case where only standardized credit history is used. We adopt NN1 model with 1 hidden layer and 30 hidden units, apply the ReLU activation function, and set the dropout probability to 0.1. The results in Table 7.4 show that we the best prediction accuracy is obtained using only credit history. Integrating other features

²<https://textblob.readthedocs.io/>

Model	Valid	Test
MM [69]	0.474	0.442
LSTM [67]	0.392	0.399
LSTM+Att [67]	0.407	0.415
MMFD [66]	-	0.388
NN1 (7.3)	0.485	0.446

TABLE 7.4: Accuracy on LIAR dataset when only credit history is used in our NN1 model. Accuracies of other models in the literature are also given. Best accuracy in validation and testing sets are bold. “Att” is short for “attention”.

even decrease the accuracy possibly because they introduce additional noise. We check the original credit history and find that there are 2439 samples in which the summation of credit history is 1. Therefore, if one speaker only contributed one statement and his credit history is $(0, 0, 0, 0, 1)$, we will know that the truthfulness label should be pants-fire. In [65], the author subtracted the current label from the credit history when using this meta-data vector in prediction experiments. Actually this kind of credibility information are not often available during predictions [68]. In view of this, in the following discussions we will not use credit history information.

We concatenate statement text feature, textual metadata feature and sentiment feature to obtain the full 1053-dimensional feature vector for a piece of news. For the set of truthfulness classes, we have $\mathcal{L} = \{1, 2, 3, 4, 5, 6\}$. For NN2, we simply concatenate x_i and x_j as input. We now define the label-mapping function in (7.4). We first assign each truthfulness class in $\mathcal{L}$ to a real value according to

$$\begin{aligned} g : \mathcal{L} &\rightarrow \mathcal{F} = \{-1.0, -0.6, -0.2, 0.2, 0.6, 1.0\} \\ l_i &\mapsto f_i = g(l_i). \end{aligned} \quad (7.9)$$

If $l_i = 1$, we let $f_i = -1.0$ and so on. We see that the values in $\mathcal{F}$ to some extent reflect the level of truthfulness. Similar to the example in Fig. 7.1, we let $\mathcal{R} = \{1, 2, 3, 4\}$ and

$$r_{ij} = \phi(l_i, l_j) = \begin{cases} 1 & \text{if } f_i = f_j, \\ 2 & \text{if } \text{sign}(f_i) = \text{sign}(f_j) \text{ and } f_i \neq f_j, \\ 3 & \text{if } \text{sign}(f_i) = -\text{sign}(f_j) \text{ and } f_i \neq -f_j, \\ 4 & \text{if } f_i = -f_j, \end{cases} \quad (7.10)$$

where $\text{sign}(\cdot)$ is the sign function. Then we are able to compute $P(l_j \mid l_i, r_{ij})$

Hyperparameter	NN1	NN2
Input dimension	1053	2106
Output dimension	6	4
# Hidden layer	1	1
# Hidden units	500	1000
Activation function	ReLU	ReLU
Optimizer	SGD	SGD
Momentum	0.9	0.9
Learning rate	$1e - 4$	$1e - 4$
Batch size	32	64
Dropout	0.1	0.2

TABLE 7.5: Hyperparameter settings for NN1 and NN2.

according to (7.5). For example, $P(l_j \mid l_i = 2, r_{ij} = 1) = [0, 1, 0, 0, 0, 0]$ and $P(l_j \mid l_i = 2, r_{ij} = 2) = [0.5, 0, 0.5, 0, 0, 0]$. In constructing the training set, we pair two pieces of news only if they have similar topics. For this LIAR dataset, each news has a collection of topics such as history, environment and economy. We pair two pieces of news if they have at least one common topic. For example, the pair of news in Table 7.3 have no common topic, so it is not included in the training set. Given a new statement, we also need to find its topics manually. From (7.10) we see that the relationship class ratio is approximately $1 : 2 : 2 : 1$. We also downsample the majority classes to make the training set even [123]. We generate the validation and test set similarly, and obtain a training, validation and testing set whose sizes are 160512, 42200 and 41512 respectively. During training we apply the SGD method. We strictly tune all the hyperparameters on the validation dataset. We thus train a NN2 model whose validation accuracy and testing accuracy are 0.365 and 0.363 respectively. Hyperparameters of NN1 and NN2 are summarized in Table 7.5

After obtaining NN1 and NN2, given new sample x_k , we first compute y_k from (7.2) and z_k from (7.6) and (7.7). We then estimate its label according to (7.8). The hyperparameter λ in (7.8) is tuned on the validation sets of NN1 and NN2 and is set to 0.05. This means that the introduction of NN2 and graph model greatly contributes to a success prediction. Refer Table 7.6 for experiment results of our NN1 model and hybrid model. We also include some baselines in the literature for comparison. We see that our hybrid model performs best when consider statement text together with metadata as input. We also observe that the removal of

Model	Statement		Statement+Metadata	
	Valid	Test	Valid	Test
Majority	0.204	0.208	-	-
CNN [65]	0.260	0.270	0.247	0.274
MM [69]	-	-	0.247	0.257
LSTM [67]	0.241	0.245	0.270	0.275
LSTM+Att [67]	0.250	0.255	0.274	0.279
MMFD [66]	-	0.291	-	-
NN1 (7.3)	0.308	0.294	0.312	0.298
Hybrid (7.8)	0.315	0.310	0.319	0.313
Hybrid+Opt (7.13)	0.317	0.311	0.322	0.315

TABLE 7.6: Performance comparison (accuracy) for fake news detection on LIAR dataset. There are two kinds of inputs. One is statement-only input, the other is statement combined with additional metadata (excluding credit history). “Att” indicates “Attention” and “Opt” indicates “Optimization”. Best accuracy in validation and testing sets are bold.

metadata can decrease the accuracy for all models, but our hybrid model is still the best one.

For a single piece of news, our hybrid model predicts its truthfulness by combining its internal information (feature vector) and information of relevant news (feature vectors and truthfulness labels of its neighbors). We further consider the case where a group of news is required to identify. Specifically, given a testing set $X_t = \{x_p, \dots, x_q, \dots\}$, we want to simultaneously predict their labels. Besides directly applying our hybrid model for each of them, we also make use of their relationships and introduce a global optimization over the whole testing set. We expect that such a step can modify incorrect predictions of some individuals by incorporating their neighbors’ information. We first construct a graph with the set of nodes X_t and the set of edges $\{\hat{r}_{pq}\}$ obtained from (7.6). For each $x_p \in X_t$, we predict its truthfulness labels $\hat{l}_p$ according to (7.8). To utilize graph smoothness, we map each $\hat{l}_p$ to the graph signal $\hat{f}_p$ according to (7.9) and map each $\hat{r}_{pq}$ to $\hat{w}_{pq}$ following the ascending order:

$$\begin{aligned} \mathcal{R} \rightarrow \mathcal{W} &= \{-1.0, -0.5, 0.5, 1.0\} \\ r_{pq} &\mapsto w_{pq}. \end{aligned} \tag{7.11}$$

The mapping (7.11) should consider the meaning of relationships such that the mapped weights can reflect the similarity between nodes. Let $\hat{\mathbf{f}}$ be the vector of

graph signal with $\hat{\mathbf{f}}(x_p) = \hat{f}_p$, we then solve a convex optimization problem

$$\tilde{\mathbf{f}} = \arg \min_{\mathbf{f}, \mathbf{f}(\cdot) \in \mathbb{R}} \left(\left\| \mathbf{f} - \hat{\mathbf{f}} \right\|_2 + \gamma \sum_{(x_p, x_q) \in E} |w_{pq}| (\mathbf{f}(x_p) - \text{sign}(w_{pq}) \mathbf{f}(x_q))^2 \right) \quad (7.12)$$

where the second term of the right side is the generalized graph smoothness (2.4). After obtaining $\tilde{\mathbf{f}}$, we are able to estimate the label by

$$\begin{aligned} \tilde{f}_p &= \arg \min_{f \in \mathcal{F}} \left| f - \tilde{\mathbf{f}}(x_p) \right| \\ \tilde{l}_p &= g^{-1}(\tilde{f}_p) \end{aligned} \quad (7.13)$$

where $g^{-1}(\cdot)$ is the inverse function of (7.9). We apply (7.13) on the LIAR dataset. We set $\gamma = 0.05$ in (7.12) which is tuned from the validation set. Experiment results are shown in Table 7.6 corresponding to the model “Hybrid+Opt” where “Opt” is short for “Optimization”. We see that (7.13) can further improve the prediction accuracy, though by a small amount.. However, the model (7.13) may not be used in practice. Sometimes we do not have enough testing samples and samples can be irrelevant. Another important reason is that the tuning of γ in (7.12) depends of the size of the validation set and γ can fail if the sizes of the validation set and testing set are very different.

Finally, we study the robustness of our models. Since it is difficult to identify fake news or rate the truthfulness of a statement even by professionals, there may exist some mislabeled statements. For the LIAR dataset, we randomly choose a portion of samples from the training set and shuffle their labels randomly. We then learn our models from the corrupted training set and test on clean testing set. See Fig. 7.4 for the experiment results. It shows that our NN1 model is comparatively robust to the noisy truthfulness labels. However, when the fraction of noisy labels is greater than 0.3, the performance of our hybrid model drops quickly and is worse than NN1 model. Our hybrid model is a combination of NN1, NN2 and a graph model. Noisy truthfulness labels will result in noisy relationship labels thus affect the correctness of NN2. Then for the graph model, both the graph signals and edge weights are noisy which produces inaccurate scores in (7.7). Therefore, the hybrid model is less robust than NN1 model according to (7.8). For comparison, we also test on the case where NN2 is trained on clean data. Fig. 7.4 shows that

FIG. 7.4: Performance of our models on the LIAR dataset with corrupted labels.

our hybrid model will also be robust and better than NN1. Some research on the learning with noisy labels can be found in [124, 125].

7.3 Conclusion

In this Chapter, we have developed a hybrid learning framework to identify the truthfulness of a piece of news based on statement text and metadata. We first train two neural networks to model the truthfulness of a piece of news and relationship between a pair of news respectively. A graph model is then combined to obtain a vector of likelihood scores for different truthfulness classes. We further consider the case where a collection of news is required to label. We make use of techniques in graph signal processing and introduce an optimization step to correct some wrong predictions from our hybrid model. Experiment results on the publicly LIAR dataset demonstrate that our hybrid model performs better than several machine learning models in the literature and the optimization step can further improve the performance. However, our hybrid model is not restricted to this specific problem. It would be interesting and meaningful to apply the framework to more machine learning applications.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis, we have investigated the problem of identifying misinformation and their sources. We have developed methods to recover network topology, estimate misinformation sources, and detect fake news.

In Chapter 3, we have developed a method for graph topology inference using information cascades and knowledge of some moments of the diffusion distribution across each edge, without needing to know the distribution itself. In the case of tree networks, we proposed an iterative tree inference algorithm. Simulations demonstrate that our method outperforms the tree reconstruction algorithm in [19]. We have studied how the moments of the propagation time between two nodes in a general graph behave, and extended our tree inference method heuristically to general graphs. Our simulation results suggest that our graph inference algorithm performs reasonably well, if the total number of cascades is not too small compared to the size of the network, and often outperforms the NetRate algorithm in [18] and KernelCascade algorithm in [27]. Moreover, our method is suitable for time-critical applications owing to its low complexity.

In Chapter 4, we have developed a sequential source estimation algorithm for identifying a single misinformation source using a stream of infection timestamps. Our method is based on constructing a reverse propagation process, which allows us to build a particle filter for sequential estimation. Each sequential update has low

complexity, which allows the source estimate to be updated quickly whenever a new timestamp is available. Experiments on both synthetic and real-world data suggest that our approach can find the misinformation source to within a small number of hops from the true source.

In Chapter 5 and Chapter 6, we have developed algorithms to estimate infection sources using a subset of timestamps for the cases where there is a single source, and where there are multiple sources and the number of sources is unknown. We adopt a Gaussian spreading model with unknown diffusion parameters, which are estimated as part of our inference algorithms. In the single source case, we introduce a new heuristic that optimizes over a parametrized family of Gromov matrices to more accurately identify the source in a general graph. In the multiple sources estimation problem, we develop theory that allows us to partition the observed infected nodes into observation clusters. We then apply our single source algorithm in each cluster. Experimental evaluations with synthetic and real-world data suggest that our approaches can find the infection sources to within a small number of hops from the true sources with a small number of timestamp observations.

In Chapter 7, we have developed a hybrid learning framework to identify the truthfulness of a piece of news based on statement text and metadata. We first train two neural networks to model the truthfulness of a piece of news and relationship between a pair of news respectively. A graph model is then combined to obtain a vector of likelihood scores for different truthfulness classes. We further consider the case where a collection of news is required to label. We make use of techniques in graph signal processing and introduce an optimization step to correct some wrong predictions from our hybrid model. Experiment results on the publicly LIAR dataset demonstrate that our hybrid model performs better than several machine learning models in the literature and the optimization step can further improve the performance.

8.2 Future Work

For the problem of misinformation sources estimation, we require the knowledge of network topology which may not be known in advance in many applications. In Chapter 3, we have shown that it is difficult to accurately recover a network

with limited information. Therefore, we may want to investigate how incomplete or inaccurate network topology would affect the performance of identifying misinformation sources.

The works [48, 49] have developed information diffusion strategies that make it difficult for a network administrator to estimate the information source accurately. These works assume that the network administrator has access only to the infection status of each node, but not the infection timestamps. With infection timestamps, it becomes more difficult to obfuscate the identity of the source. Since our approach involves estimation of the diffusion parameters and can handle model mismatch to some extent, as shown in our simulations on malware source estimation, we expect our approach to be somewhat robust to obfuscation strategies. Designing good source obfuscation strategies when timestamp information is available, and corresponding source estimation strategies to counter them are potential interesting future research directions.

We have considered the case where the infected nodes and their timestamps are sequentially observed. However, these observed nodes may not be optimal for source identification. For example, they are far away from the actual source and close to each other. The works [126, 127] studied the problem of sequential sensor placement. Each sensor is a specifically chosen node who can reveal its infection state or infection timestamp. The authors in [126, 127] demonstrated that by using dynamic sensors, the number of sensors necessary to localize the source is reduced a lot. It would be interesting to extend our work to the sequential sensor placement case.

For the fake news detection, we have proposed a hybrid model and tested on the LIAR dataset. However, our hybrid model is not restricted to this specific problem. It would be interesting and meaningful to apply the framework to more machine learning applications.

List of Author's Awards, Patents, and Publications¹

Journal Articles

- *F. Ji, ***W. Tang**, W. P. Tay, and E. K. P. Chong, "Network topology inference using information cascades with limited statistical knowledge," *Inf. Inference: J. IMA*, 2019, accepted.
- F. Ji, **W. Tang**, and W. P. Tay, "On the properties of Gromov matrices and their applications in network inference," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2624-2638, May 2019.
- **W. Tang**, F. Ji, and W. P. Tay, "Estimating infection sources in networks using partial timestamps," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 2, pp. 3035-3049, Dec. 2018.

Conference Proceedings

- F. Ji, **W. Tang**, and W. P. Tay, "Properties and applications of Gromov matrices in network inference," in *Proc. IEEE Workshop Stat. Signal Process.*, Jun. 2018.
- **W. Tang**, F. Ji, and W. P. Tay, "Multiple sources identification in networks with partial timestamps," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Nov. 2017.

¹The superscript * indicates joint first authors

- F. Ji, **W. Tang**, W. P. Tay, and E. K. P. Chong, “Inferring network topology from information cascades,” in *Proc. IEEE Int. Symp. on Inf. Theory*, Jun. 2017.
- **W. Tang** and W. P. Tay, “A particle filter for sequential infection source estimation,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2017.

Bibliography

- [1] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proc. ACM Int. Conf. Knowl. Discov. Data Min.*, pages 611–617, 2006. ISBN 1-59593-339-5. [1](#)
- [2] H. Berghel. The code red worm. *Commun. ACM*, 44(12):15–19, 2001. [1](#)
- [3] R. Richardson and CSI Director. Csi computer crime and security survey. *Comput. Secur. Inst.*, 1:1–30, 2008.
- [4] Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. *J. Econ. Perspect.*, 31(2):211–36, 2017.
- [5] Vian Bakir and Andrew McStay. Fake news and the economy of emotions: Problems, causes, solutions. *Digit. J.*, 6(2):154–175, 2018. [1](#)
- [6] Eytan Bakshy, Jake M. Hofman, Winter A. Mason, and Duncan J. Watts. Everyone’s an influencer: quantifying influence on Twitter. In *Proc. 4th ACM International Conference on Web Search and Data Mining*, 2011. ISBN 978-1-4503-0493-1. [1](#)
- [7] C. Stephen. China’s panic-buying of salt. <http://www.marketwatch.com/story/what-chinas-panic-buying-of-salt-means-2011-03-20>. Accessed March 20, 2011. [1](#)
- [8] C. Kang and A. Goldman. In Washington pizzeria attack, fake news brought real guns. <https://www.nytimes.com/2016/12/05/business/media/comet-ping-pong-pizza-shooting-fake-news-consequences.html>, 2016. [1](#)
- [9] A. Guille, H. Hacid, C. Favre, and D. A. Zighed. Information diffusion in online social networks: A survey. *ACM SIGMOD Rec.*, 42(2):17–28, July 2013. [1](#)

- [10] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86(14):3200–3203, Apr 2001. [1](#)
- [11] B.S. Manoj, A. Chakraborty, and R. Singh. *Complex Networks: A Networking and Signal Processing Perspective*. Prentice Hall, 2018. [2](#)
- [12] M. EJ Newman. The structure and function of complex networks. *SIAM Rev.*, 45(2):167 – 256, 2003. [2](#)
- [13] Luciano Da Fontoura Costa, Osvaldo N Oliveira Jr, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana, and Luis Enrique Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Adv. Phys.*, 60(3):329 – 412, 2011. [2](#)
- [14] Dirk Helbing, Dirk Brockmann, Thomas Chadeaux, Karsten Donnay, Ulf Blanke, Olivia Woolley-Meza, Mehdi Moussaid, Anders Johansson, Jens Krause, Sebastian Schutte, and M. Perc. Saving human lives: What complexity science and information systems can contribute. *J. Stat. Phys.*, 158(3):735 – 781, 2015.
- [15] Marko Gosak, Rene Markovič, Jurij Dolenšek, Marjan Slak Rupnik, Marko Marhl, Andraž Stožer, and Matjaž Perc. Network science of biological systems at different scales: a review. *Phys. Life Rev.*, 24:118 – 135, 2018.
- [16] A Guille, H Hacid, C Favre, and Djamel A. Zighed. Information diffusion in online social networks: A survey. *SIGMOD Rec.*, 42(2):17 – 28, July 2013. [2](#), [3](#)
- [17] M. G. Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 1019 – 1028, 2010. [2](#), [3](#), [28](#), [34](#)
- [18] M. G. Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proc. ACM Int. Conf. Mach. Learn.*, pages 561 – 568, 2011. [2](#), [3](#), [8](#), [15](#), [30](#), [34](#), [36](#), [39](#), [40](#), [101](#)
- [19] B. Abrahao, F. Chierichetti, R. Kleinberg, and A. Panconesi. Trace complexity of network inference. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, pages 491 – 499, 2013. [2](#), [3](#), [8](#), [30](#), [32](#), [40](#), [101](#)

- [20] Seth A. Myers and Jure Leskovec. On the convexity of latent social network inference. In *Proc. Conf. Neural Inf. Process. Syst.*, pages 1741 – 1749, 2010. [2](#), [34](#)
- [21] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. *SIGMETRICS Perform. Eval. Rev.*, 40:211 – 222, June 2012. [2](#), [3](#)
- [22] Manuel Gomez-Rodriguez, Le Song, Hadi Daneshmand, and Bernhard Schölkopf. Estimating diffusion networks: Recovery conditions, sample complexity and soft-thresholding algorithm. *J. Mach. Learn. Res.*, 17(90):1 – 29, 2016. [3](#)
- [23] Sanda Martinčić-Ipšić, Edvin Močibob, and M. Perc. Link prediction on Twitter. *PLoS ONE*, 12(7):e0181079, 2017. [3](#)
- [24] Mahdi Jalili, Yasin Orouskhani, Milad Asgari, Nazanin Alipourfard, and M. Perc. Link prediction in multiplex online social networks. *R. Soc. Open Sci.*, 4(2):160863. [3](#)
- [25] D. Gruhl, R. Guha, D. L. Nowell, and A. Tomkins. Information diffusion through blogspace. In *Proc. ACM Int. Conf. World Wide Web*, 2004. [3](#)
- [26] Damon Centola. The spread of behavior in an online social network experiment. *Science*, 329(5996):1194 – 1197, 2010. [3](#)
- [27] Nan Du, Le Song, Ming Yuan, and Alex J. Smola. Learning networks of heterogeneous influence. In *Proc. Conf. Neural Inf. Process. Syst.*, pages 2780 – 2788. 2012. [4](#), [8](#), [34](#), [40](#), [101](#)
- [28] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83 – 98, May 2013. [4](#), [93](#)
- [29] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning Laplacian matrix in smooth graph signal representations. *IEEE Trans. Signal Process.*, 64(23):6160 – 6173, Dec 2016. [4](#)
- [30] V. Kalofolias. How to learn a graph from smooth signals. In *Proc. Int. Conf. Artificial Intell. Stat.*, pages 920 – 929, May 2016.

- [31] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro. Network topology inference from spectral templates. *IEEE Trans. Signal Inf. Process. Netw.*, 3:467 – 483, 2017. [4](#)
- [32] D. Shah and T. Zaman. Rumors in a network: Who’s the culprit? *IEEE Trans. Inf. Theory*, 57(8):5163–5181, 2011. [4](#), [5](#), [9](#), [54](#)
- [33] K. Zhu and L. Ying. Information source detection in the SIR model: a sample path based approach. In *Proc. Inf. Theory Appl. Workshop*, 2013. [4](#), [5](#)
- [34] W. Luo and W. P. Tay. Finding an infection source under the SIS model. In *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, pages 2930–2934, 2013. [4](#), [5](#)
- [35] Devavrat Shah and Tauhid Zaman. Rumor centrality: a universal source detector. In *Proc. ACM SIGMETRICS Perform. Eval. Rev.*, volume 40, pages 199–210, 2012. [5](#)
- [36] W. Luo, W. P. Tay, and M. Leng. Identifying infection sources and regions in large networks. *IEEE Trans. Signal Process.*, 61(11):2850–2865, 2013. [5](#), [81](#)
- [37] V. Fioriti, M. Chinnici, and J. Palomo. Predicting the sources of an outbreak with a spectral technique. *Appl. Math. Sci.*, 8(133-136):6775–6782, 2014.
- [38] B. A. Prakash, J. Vreeken, and C. Faloutsos. Spotting culprits in epidemics: How many and which ones? In *IEEE Int. Conf. Data Mining*, pages 11 – 20, Brussels, Belgium, 2012.
- [39] F. Ji, W. P. Tay, and L. Varshney. An algorithmic framework for estimating rumor sources with different start times. *IEEE Trans. Signal Process.*, 65(10):2517–2530, May 2017. [5](#)
- [40] W. Luo, W. P. Tay, and M. Leng. How to identify an infection source with limited observations. *IEEE J. Sel. Top. Signal Process.*, 8(4):586–597, 2014. [5](#)
- [41] Z Wang, W Dong, W Zhang, and C W Tan. Rumor source detection with multiple observations: Fundamental limits and algorithms. In *Proc. ACM SIGMETRICS*, 2014. [5](#)

- [42] K. Zhu and L. Ying. Information source detection in the SIR model: A sample-path-based approach. 24(1):408–421, Feb 2016. ISSN 1063-6692. doi: 10.1109/TNET.2014.2364972. [5](#)
- [43] F. Altarelli, A. Braunstein, L. Dall’Asta, A. Lage-Castellanos, and R. Zecchina. Bayesian inference of epidemics on networks via belief propagation. *Phys. Rev. Lett.*, 112:118701, Mar 2014. [5](#)
- [44] A. Y. Lokhov, M. Mézard, H. Ohta, and L. Zdeborová. Inferring the origin of an epidemic with a dynamic message-passing algorithm. *Phys. Rev. E*, 90: 012801, Jul 2014. [5](#)
- [45] K. Zhu and L. Ying. Source localization in networks: Trees and beyond. *arXiv preprint arXiv:1510.01814*, 2015. [5](#)
- [46] Z. Hu, X. Han, Y. Lai, and W. Wang. Optimal localization of diffusion sources in complex networks. *R. Soc. Open Sci.*, 4(4):170091, 2017. [5](#)
- [47] A. Kumar, V. S. Borkar, and N. Karamchandani. Temporally agnostic rumor-source detection. *IEEE Trans. Signal and Inf. Process. Netw.*, 3(2):316–329, June 2017. [5](#)
- [48] W. Luo, W. P. Tay, and M. Leng. Infection spreading and source identification: A hide and seek game. *IEEE Trans. Signal Process.*, 64(16):4228–4243, Aug 2016. [5](#), [103](#)
- [49] G. Fanti, P. Kairouz, S. Oh, and P. Viswanath. Spy vs. spy: Rumor source obfuscation. In *Proc. ACM SIGMETRICS Perform. Eval. Rev.*, volume 43, pages 271–284, 2015. [5](#), [103](#)
- [50] Pedro C. Pinto, Patrick Thiran, and Martin Vetterli. Locating the source of diffusion in large-scale networks. *Phys. Rev. Lett.*, 109:068702, 2012. [5](#), [9](#), [13](#), [47](#), [54](#), [56](#), [60](#), [61](#)
- [51] K. Zhu, Z. Chen, and L. Ying. Locating the contagion source in networks with partial timestamps. *Data Min. Knowl. Discov.*, pages 1–32, 2015. [5](#), [45](#), [62](#)
- [52] A. Louni and K. P. Subbalakshmi. A two-stage algorithm to estimate the source of information diffusion in social media networks. In *Proc. IEEE Conf. Comput. Commun. Workshops*, pages 329–333, April 2014. [5](#), [9](#), [13](#), [54](#)

- [53] Wenchang Tang and Wee Peng Tay. A particle filter for sequential infection source estimation. In *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, March 2017. [5](#), [9](#), [13](#)
- [54] Z. Shen, S. Cao, W. Wang, Z. Di, and H. E. Stanley. Locating the source of diffusion in complex networks by time-reversal backward spreading. *Phys. Rev. E*, 93:032301, Mar 2016. [5](#), [9](#), [60](#), [61](#)
- [55] S. Zejnilović, J. Gomes, and B. Sinopoli. Network observability and localization of the source of diffusion based on a subset of nodes. In *Proc. IEEE Annu. Allerton Conf. on Commun. Control Comput.*, pages 847–852, 2013. [5](#)
- [56] S. Zejnilović, J. Xavier, J. Gomes, and B. Sinopoli. Selecting observers for source localization via error exponents. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 2914–2918, 2015. [5](#)
- [57] L. Fu, Z. Shen, W. Wang, Y. Fan, and Z. Di. Multi-source localization on complex networks with limited observers. *Europhys. Lett.*, 113(1):18006, 2016. [6](#), [10](#), [81](#)
- [58] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *Proc. ACM Int. Conf. World Wide Web*, pages 675–684, 2011. [6](#)
- [59] Jing Ma, Wei Gao, Zhongyu Wei, Yueming Lu, and Kam-Fai Wong. Detect rumors using time series of social context information on microblogging websites. In *Proc. ACM Int. Conf. Inf. Knowl. Manag.*, pages 1751–1754, 2015. [6](#)
- [60] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6):e0128193, 2015. [6](#)
- [61] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. Truth finding on the deep web: Is the problem solved? *Proc. VLDB Endowment*, 6(2):97–108, 2012.
- [62] Ndapandula Nakashole and Tom M Mitchell. Language-aware truth assessment of fact candidates. In *Proc. Annu. Meeting Assoc. Comput. Linguist.*, pages 1009–1019, 2014.

- [63] You Wu, Pankaj K Agarwal, Chengkai Li, Jun Yang, and Cong Yu. Toward computational fact-checking. *Proc. VLDB Endowment*, 7(7):589–600, 2014. [6](#)
- [64] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explor. Newsl.*, 19(1):22–36, 2017. [6](#), [8](#)
- [65] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017. [7](#), [94](#), [96](#), [98](#)
- [66] Hamid Karimi, Proteek Roy, Sari Saba-Sadiya, and Jiliang Tang. Multi-source multi-class fake news detection. In *Proc. Int. Conf. Comput. Linguist.*, pages 1546–1557, 2018. [7](#), [95](#), [96](#), [98](#)
- [67] Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. Fake news detection through multi-perspective speaker profiles. In *Proc. Int. Joint Conf. Natur. Lang. Process.*, pages 252–256, 2017. [7](#), [95](#), [96](#), [98](#)
- [68] Angelika Kirilin and Micheal Strube. Exploiting a speakers credibility to detect fake news. In *Proc. Data Sci. J. Media*, 2018. [7](#), [96](#)
- [69] Trung Tin Pham. A study on deep learning for fake news detection. 2018. [7](#), [95](#), [96](#), [98](#)
- [70] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018. [7](#)
- [71] James Thorne and Andreas Vlachos. Automated fact checking: Task formulations, methods and future directions. *arXiv preprint arXiv:1806.07687*, 2018. [8](#)
- [72] Wenpeng Yin and Dan Roth. Twowingos: A two-wing optimization strategy for evidential claim verification. *arXiv preprint arXiv:1808.03465*, 2018.
- [73] Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. Ukp-athene: Multi-sentence textual entailment for claim verification. *arXiv preprint arXiv:1809.01479*, 2018.

- [74] James Thorne and Andreas Vlachos. Adversarial attacks against fact extraction and verification. *arXiv preprint arXiv:1903.05543*, 2019. [8](#)
- [75] Kai Shu, Suhang Wang, and Huan Liu. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*, 2017. [8](#)
- [76] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018. [8](#)
- [77] Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proc. Conf. Empir. Methods Natur. Lang. Process.*, pages 2931–2937, 2017. [8](#)
- [78] Marco L Della Vedova, Eugenio Tacchini, Stefano Moret, Gabriele Ballarin, Massimo DiPierro, and Luca de Alfaro. Automatic online fake news detection combining content and social signals. In *IEEE Conf. Open Innov. Assoc.*, pages 272–279, 2018. [8](#)
- [79] Vanessa Wei Feng and Graeme Hirst. Detecting deceptive opinions with profile compatibility. In *Proc. Int. Joint Conf. Natur. Lang. Process.*, pages 338–346, 2013. [8](#)
- [80] David M Markowitz and Jeffrey T Hancock. Linguistic traces of a scientific fraud: The case of diederik stapel. *PloS one*, 9(8):e105937, 2014. [8](#)
- [81] Ray Oshikawa, Jing Qian, and William Yang Wang. A survey on natural language processing for fake news detection. *arXiv preprint arXiv:1811.00770*, 2018. [8](#)
- [82] Karishma Sharma, Feng Qian, He Jiang, Natali Ruchansky, Ming Zhang, and Yan Liu. Combating fake news: A survey on identification and mitigation techniques. *ACM Trans. Intell. Syst. Technol.*, 10(3):21, 2019. [8](#)
- [83] L. Zheng and C. W. Tan. A probabilistic characterization of the rumor graph boundary in rumor source detection. In *Proc. IEEE Int. Conf. Digit. Signal Process.*, pages 765–769, 2015. [9](#), [54](#)

- [84] Richard Durrett. *Probability: Theory and Examples*. Duxbury Press, New York, second edition, 1995. [15](#)
- [85] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997. [16](#)
- [86] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer, 2003. [17](#)
- [87] Dengyong Zhou and Bernhard Schölkopf. A regularization framework for learning from graph data. In *ICML Workshop Statistical Relational Learning and Its Connections to Other Fields*, volume 15, pages 67–68, 2004. [17](#)
- [88] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015. [19](#)
- [89] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inf. process. Syst.*, pages 5998–6008, 2017. [19](#), [21](#)
- [90] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Adv. Neural Inf. process. Syst.*, pages 3111–3119, 2013. [19](#), [95](#)
- [91] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proc. Conf. Empir. Methods Natur. Lang. Process.*, pages 1532–1543, 2014. [19](#)
- [92] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. [19](#)
- [93] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. [19](#)
- [94] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. [20](#), [21](#), [95](#)

- [95] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018. [20](#)
- [96] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI, 2018. [20](#)
- [97] Feng Ji, Wenchang Tang, Wee Peng Tay, and Edwin K. P. Chong. Network topology inference using information cascades with limited statistical knowledge. *Inf. Inference: J. IMA*, 2019. accepted. [23](#), [26](#)
- [98] Paul Erdős and Alfréd Rényi. On random graphs I. *Publ. Math. Debrecen.*, 6:290 – 297, 1959. [28](#), [44](#)
- [99] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007. [28](#), [33](#), [34](#)
- [100] Enrique Castillo. *Extreme Value Theory in Engineering*. Academic Press, San Diego, 1988. [28](#)
- [101] Ursula Gather. Robust estimation of the mean of the exponential distribution in outlier situations. *Commun. Stat. Theory Methods*, 15(8):2323 – 2345, 1986. [29](#)
- [102] E. S. Ahmed, A. I. Volodin, and A. A. Hussein. Robust weighted likelihood estimation of exponential parameters. *IEEE Trans. Rel.*, 54(3):389 – 395, Sept. 2005. [29](#)
- [103] Mark F Schilling, Ann E Watkins, and William Watkins. Is human height bimodal? *Am. Stat.*, 56(3):223 – 229, 2002. [38](#)
- [104] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking. *IEEE Trans. Signal Process.*, 50(2):174–188, 2002. [44](#)
- [105] R. Douc and O. Cappé. Comparison of resampling schemes for particle filtering. In *Proc. IEEE Int. Symp. Image Signal Process. and Anal.*, pages 64–69, 2005. [44](#)

- [106] I. Kapovich and N. Benakli. Boundaries of hyperbolic groups. *arXiv preprint math/0202286*, 2002. [51](#)
- [107] O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *J. Multivar. Anal.*, 88(2):365–411, February 2004. [58](#)
- [108] A. Touloumis. Nonparametric stein-type shrinkage covariance matrix estimators in high-dimensional settings. *Comput. Stat. & Data Anal.*, 83:251–261, 2015. [58](#)
- [109] Feng Ji, Wenchang Tang, and Wee Peng Tay. On the properties of Gromov matrices and their applications in network inference. *IEEE Trans. Signal Process.*, 67(10):2624 – 2638, May 2019. [59](#)
- [110] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. [62](#)
- [111] Kaspersky Labs. Facebook malware poses as flash update, infects 110k users. February 2015. [65](#)
- [112] Bradley Barth. Malware attacking Facebook, LinkedIn users via malicious images. November 2016. [65](#)
- [113] W. Fan and K.H. Yeung. Online social networks-paradise of computer viruses. *Physica A: Stat. Mech. Appl.*, 390(2):189–197, 2011. [65](#)
- [114] Guanhua Yan, Guanling Chen, Stephan Eidenbenz, and Nan Li. Malware propagation in online social networks: Nature, dynamics, and defense implications. In *Proc. ACM Symp. Inf., Comput. Commun. Secur.*, pages 196–206, 2011.
- [115] M. R. Faghani, A. Matrawy, and C. H. Lung. A study of Trojan propagation in online social networks. In *Proc. Int. Conf. New Tech., Mobility and Secur.*, pages 1–5, May 2012.
- [116] Mohammad Reza Faghani and Uyen Trang Nguyen. Modeling the propagation of Trojan malware in online social networks. *arXiv preprint abs/1708.00969*, 2017. [65](#), [67](#), [68](#)
- [117] C. Anthe, P. Charzan, and E. Florio. Microsoft security intelligence report. April 2015. [67](#)

- [118] C. D. Meyer. *Matrix analysis and applied linear algebra*. Siam, 2000. [71](#)
- [119] F. Ji, W. P. Tay, and L. R. Varshney. Estimating the number of infection sources in a tree. In *Proc. IEEE Global Conf. Signal and Inf. Process.*, pages 380–384, Dec 2016. [79](#)
- [120] M. De Domenico, A. Lima, P. Mougél, and M. Musolesi. The anatomy of a scientific rumor. *Sci. Rep.*, 3:2980, 2013. [85](#)
- [121] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proc. Int. AAAI Conf. Weblogs Soc. Media*, 2014. [90](#), [95](#)
- [122] S. Chen, R. Varma, A. Singh, and J. Kovačević. Signal recovery on graphs: Fundamental limits of sampling strategies. *IEEE Trans. Signal Inf. Process. over Netw.*, 2(4):539–554, Dec 2016. [93](#)
- [123] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, 2002. [97](#)
- [124] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Adv. Neural Inf. Process. Syst.*, pages 1196–1204, 2013. [100](#)
- [125] Brendan Van Rooyen and Robert C Williamson. A theory of learning with corrupted labels. *J. Mach. Learn. Res.*, 18:228–1, 2017. [100](#)
- [126] B. Spinelli, L. E. Celis, and P. Thiran. A general framework for sensor placement in source localization. *IEEE Trans. Netw. Sci. Eng.*, 6(2):86–102, April 2019. [103](#)
- [127] S. Zejnilović, J. Gomes, and B. Sinopoli. Sequential observer selection for source localization. In *Proc. IEEE Global Conf. Signal and Inf. Process.*, pages 1220–1224, Dec 2015. [103](#)