Social Media Analytics with Graph Convolutional Networks

by

Wanyu Lin

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Social Media Analytics with Graph Convolutional Networks

Wanyu Lin
Doctor of Philosophy
Graduate Department of Electrical and Computer Engineering
University of Toronto
2020

Online social media, such as Facebook, has become a norm in our social and personal lives. The explicit or implicit social relationships established on social networks can be leveraged to market products or make recommendations. However, the open nature of online social media provides a favorable environment for malicious users to spread incorrect information, either for financial gains or to increase social influence. Therefore, social media analysis, such as social trust investigation, has attracted increasing attention from multiple disciplines. On the other hand, graph convolutional neural networks (GCNs) recently have shown to be powerful in learning on graphs. Their advantages provide great potential to analyze online social networks represented as graph data. In this dissertation, we investigate significant research problems in the context of graph convolutional networks, tackling complex social media analysis.

We begin by reviewing some key concepts and unique properties and principles of graph convolutional networks and network representation learning — a fundamental step for analyzing social networks. Then we present *Guardian* to evaluate social trust in online social networks. More specifically, *Guardian* is an end-to-end framework that stacks multiple trust convolutional layers, designed to discover hidden and predictive latent factors of trust in online social networks. With *Guardian*, we can effectively and efficiently estimate the value of trustworthiness between any two users who are not explicitly connected. Many real-world relationships can be represented as networks with positive and negative links, called *signed networks*, where the sign of links may indicate trust or distrust, and like or dislike relationships. Thus, complementary to *Guardian*,

we propose *SiGCN*, to learn low-dimensional representations of signed networks. With *SiGCN*, we can learn effective user embedding for downstream signed network analysis. Finally, to investigate the robustness of GCNs, we also study the adversarial attacks on graph-structured data, particularly mounting attacks against link prediction algorithms based on GCNs. Evaluations and validations to our frameworks are not only analytical but also experimental. We conducted extensive experiments on benchmarking datasets, and our experimental results demonstrated that our approach effectively achieves the best possible performance for accuracy, efficiency, and scalability.

To my family

Acknowledgements

This thesis represents not only my work at the keyboard but also a milestone in my research career. It becomes a reality with the support and help of many individuals. I would like to extend my sincere thanks to all of them.

First and foremost I wish to express my sincere appreciation to my supervisor, Professor Baochun Li, for his continued support, encouragement and patient guidance throughout my time as his student. Baochun has been supportive not only by providing the research assistantship, but also academically and emotionally through the rough road to finish this dissertation. I have been extremely lucky to have a supervisor who has provided me with great freedom to pursue the cutting-edge research that I am truly interested in. I remember he used to say something like "high-quality research deserves your time and efforts", which has been a source of inspiration for me to do delicate research. Baochun convincingly guided and encouraged me to be professional and to keep doing the right things, especially in those moments when things do not go as expected. Without his persistent support, the goal of this dissertation would not have been realized. Thank Baochun for teaching me so many good principles of doing exquisite work, which I still can keep following throughout my entire life.

My thanks and appreciation also go to my collaborators, Shengxiang Ji and Zhaolin Gao. They were undergraduate summer students in 2018 and 2019, respectively. We acknowledge them for assistance in conducting the experimental evaluation in this study. I would also like to thank my Ph.D. oral examination committee members, Professor Ben Liang, Professor Cristiana Amza, and Professor Song Guo for their valuable feedback and very detailed comments on this dissertation. I would like to thank Professor Christopher Yip for serving the committee chair of my defense. I am grateful to all members of the iQua research group for an enjoyable and collaborative work environment and our friendship: Hao Wang, Siqi Ji, Yinan Liu, Hao Lan, Salma Emara, Chen Ying, Tianhang Zheng, Sijia Chen, Yufei Kang, Guiyang Liu, and Ruixin Yao. Many thanks to every one of you.

Special thanks to my good friend Rui Liu, who are always ready to give help and advice. I would also like to express my gratitude to my good friends in Toronto: Tian Chen, Xinyue Zhang, and Jin Xu. I always remember the good memories we have enjoyed in different places. To my roommate, Ziqi Guo, thank you for being there whenever I needed a friend.

Finally, I give my deepest gratitude to my family, living in Jinjiang, Zhangzhou, Guangzhou, and Hong Kong. Thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my parents who supported

me emotionally and financially, to my sisters, Yaxuan, Binbin, and Anni, for listening, offering me advice, and supporting me through this entire process. To my cute nephew Yiding Liu, my nieces Ye Ding and Siyuan Ding, thank you for sharing your funny stories, your beautiful smile, which always makes my day.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The algorithmic and architectural design and implementations, for modern social media analysis using graph convolutional networks, are motivated by a close examination on graph convolutional networks on learning graphs. Graph convolutional networks have been shown remarkable performance compared with the challenges and difficulties in the traditional approaches of analyzing social media data. In this chapter, we first introduce the background and discuss the motivations of our work, then present the main contributions of the dissertation, and finally outline the rest of the dissertation.

1.1 Background

Online social media, such as Twitter and Facebook, has become popular as a medium for producing and disseminating information, and connecting like-minded people. Individuals routinely share their opinions and life experiences in social networking sites, which holds great promise to reach large audiences. Enterprises and governments can leverage the explicit or implicit social relationships established on these online social networks for delivering their services to citizens and customers. Specifically, enterprises and governments analyze social media networks to perform analytics, interest analysis, and sentimental analysis or find influencers to aid in making more informed decisions.

However, the open nature and the popularity of online social networks provide a

favorable environment for malicious users to spread negative or malicious information, either for financial gains [22] or to increase social influence [6]. Therefore, establishing trust communities – the environment where users can share their opinions and life experience in an open and interactive way without concerns about privacy leakage – has become essential in online social networks. In general, *social trust* has been regarded as a foundation for building trust communities, which has been studied in many different disciplines including psychology [17], sociology [58], economics [33], and computer science [71]. Derived from psychology and sociology [59], "user" trust can be defined as "a subjective expectation a user has about another's future behavior." Online marketplaces are often operated based on mutual trust, which enables the transacting peers to release the concerns about the uncertainty and risk inherent in the public environment [68]. In this context, trust is established based on past interactions between users.

Estimates of social trustworthiness in online social networks help indicate to what extent a user could expect someone else to perform given actions [85]. Therefore, it has many applications. [24] was proposed for semantic web content filtering, by combining provenance information and trust annotations in semantic web-based social networks. Golbeck *et al.* [25] integrated the value of social trust to create predictive movie recommendations. In trust-aware recommender systems, social trust value may be a good supplement to the sparse rating data and therefore may provide more accurate prediction rating [30]. With the importance of social trust, in the first part of this dissertation, we mainly focus on analyzing social trust in online social networks, in particular, evaluating the pairwise trust relationship between any two users who are not directly connected within online social networks.

In essence, an online social network is modeled as a graph, where nodes represent users and edges denote the explicit social relationships. For the sake of clarity, an example network is shown in Fig. 1.1, where the numbers are associated trustworthiness of the established relationships in online social networks. More specifically, 1 represents the

lowest trustworthiness, and 3 denotes the highest trustworthiness (the actual trustworthiness domain is typically application-specific). Therefore, the problem of social trust evaluation is to measure the trustworthiness of any two users, for example, A and E, given the established trust relationships in the network.

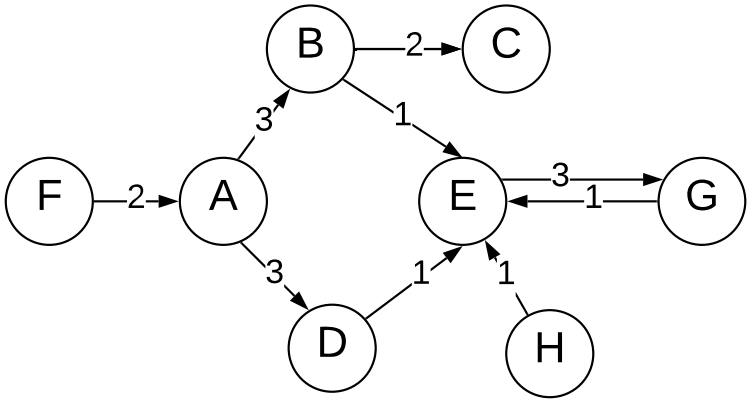


Figure 1.1: Network graph: nodes represent users, directional edges denote trust relationships of the trustor-trustee pairs, and the numbers are the associated trustworthiness (1 for the lowest trustworthiness and 3 for the highest trustworthiness).

To date, an extensive amount of work on evaluating pairwise social trust has been reported in the literature [45, 50–52, 84, 85]. For example, OpinionWalk [50] evaluated trust relationships by performing path searches throughout the network. Discounting and combining operators are designed to model trust propagation and aggregation along network paths. These hand-crafted rules rely heavily upon the knowledge of domain experts and may be difficult to be generalized to different domains. Liu *et al.* [51] proposed NeuralWalk, a framework based on neural networks, to learn trust propagation and aggregation rules with machine learning techniques. However, it required a significant amount of computation resources for its matrix operations, which is not scalable to real-world online social networks.

Apart from modeling social relationships with a value of social trust, many real-world

relationships on social media often reflect a mixture of positive and negative interactions. For example, people can be friends (positive interactions) or foes (negative interactions). These social networks can be represented as network graphs with positive and negative links, called *signed networks*. In general, *signed networks* in social media represent social relationships among users, where positive links indicate trust, like or friendships, and negative links show distrust, dislike, or foes. For example, Epinions[1], a product review site, allows users to establish both positive (trust) and negative (distrust) links to other users. Such a network can be modeled as a graph, where nodes represent users, an edge with a $+1$ sign denotes a positive link, and -1 represents a negative link, shown in Fig. 1.2. Both positive and negative links in such networks convey a much richer set of information that can be leveraged to enhance network mining tasks, such as link sign prediction, node ranking, and community detection [73].

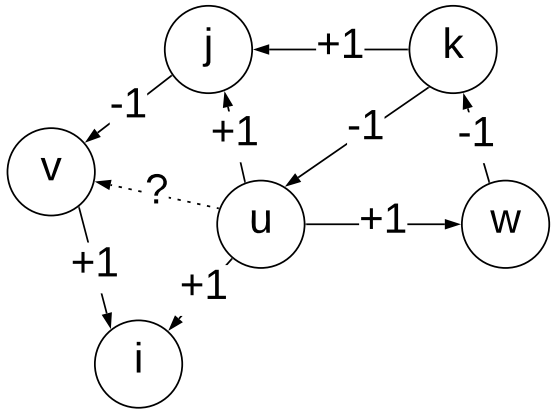


Figure 1.2: A signed directed network: an example. $+1$ represents a positive link, -1 denotes a negative link, and the arrow denotes the direction of the link.

Evidence from existing achievements in social network analysis suggests that considering both positive and negative links can significantly improve the performance of social network analysis [43]. Therefore, it is crucial to modeling social relationships with a mixture of positive and negative links and conducting analysis on these signed networks.

[1] `https://epinions.com`

Complementary to the first component of the dissertation, we shift our focus to the problem of signed network analysis. With the prevalence of signed networks, many theories from social psychology have been developed to characterize the social phenomenon in signed networks. Structural balance theory [12, 32] and status theory [44] have been proven to be very helpful in mining signed social networks. These formed social theories and the increasing availability of signed networks have significantly advanced signed network analysis. In particular, an extensive amount of work on signed network embedding — a methodology to learn low-dimensional dense representations of nodes, facilitating signed network analysis — has been reported in the literature.

Concretely, the task of signed network embedding is to learn a mapping function, which is able to encode each user with a dense vector, given a signed social network. Such a vector is informative that preserves the original network information, including network structure, established relationships (positive connection or negative connection) and user attributes, therefore may be effective for downstream signed network analysis. For example, as shown in Fig. 1.2, with learned dense vectors, we can infer whether the link from user u to v is positive or negative, given the established relationships (illustrated with solid arrows) in the network graph.

With relatively few exceptions (e.g., BESIDE [16]), research on representation learning of signed networks has focused on characterizing social structural balance theory (e.g., SIGNet [35], SIDE [39], SiNE [79], SNE [87]), which defines an organizing principle for signed links on signed networks and implies that cycles with an even number of negative signs are more plausible, thus should be more prevalent in real-world networks. However, structural balance theory is initially defined for signed undirected networks [44], and may not effectively capture the direction information of links for signed directed networks. A new approach that explicitly designed for signed directed networks, considering efficacy, efficiency and scalability, thus leads to the second component of the dissertation.

In general, a social network can be represented as a graph, which models a set of

nodes/ users and their relationships/ edges. Due to the expressive capability of graphs, there has been receiving increasing attention on analyzing graphs with machine learning. Graph convolutional neural networks (GCNs) are deep learning-based models that operate on the graph, a non-Euclidean data structure [31, 41, 86]. Designed explicitly for graph-structured data, GCNs have shown to achieve state-of-the-art performance in many graph learning tasks, such as node classification, graph classification, and link prediction [31,41,86]. The above difficulties in analyzing large-scale social networks motivate the study of graph convolutional networks on social media analytics.

1.2 Motivation

1.2.1 Advantages of Graph Convolutional Networks

In essence, graph convolutional networks were designed to capture the homophily nature of network graphs [31, 41], indicating that more similar users are more likely to connect with each other. More specifically, feature/attribute information from local graph neighborhoods is iteratively aggregated. By stacking multiple convolutions and transformations, local information can be propagated throughout the entire graph.

Among others, a particular variant of GCNs falls into the category of inductive representation learning algorithms that generate node embeddings by aggregating features from a node's local neighborhood [31,86]. These inductive learning algorithms best target large graphs. They are capable of inductively generating embeddings for unseen nodes based on their features and local graph neighbors. Their key computational workhorse is the notion of localized graph convolutions. More concretely, parameters of multiple localized convolutions are shared across all nodes, making the parameter complexity independent of the input graph size.

This range of efficiency provides great potential to social media analytics, of which the social networks typically are very large and can be represented as graph-structured data. Though the advantages of graph convolutional networks on graph learning tasks

are obvious, there are difficulties for these models to be directly applied to analyze such complex social networks. These issues are enumerated as follows.

1.2.2 Challenges in Social Trust Evaluation

In online social networks, social trust can be similarly represented as graph data, including both social network structures and associated trust relationships between users. Thus, given the advantages graph convolutional networks, excellent opportunities may exist in use of the graph convolutional neural networks to evaluate social trust relationships between pairs of users.

Existing trust evaluation approaches were designed based on the propagative and composable nature of social trust in online social networks. In particular, *the propagative nature of social trust* refers to the fact that trust may be passed from one user to another, creating chains of social trust that connects two users who are not explicitly connected [76]. The *composable nature of social trust* refers to the fact that trust needs to be aggregated if several chains of social trust exist [76]. In a nutshell, trust propagation and aggregation rules are the keys to effectively evaluate pairwise social trust in online social networks.

Yet, evaluating social trust using graph convolutional neural networks to capture trust propagation and aggregation rules is quite challenging. Online social networks not only contain the social graph structure (social connections between users) but also include pairwise social trust relationships. In this context, the first challenge is how social connections and associated trust relationships can be represented jointly so that the propagative nature and composable nature of social trust are able to be captured simultaneously. In addition, social trust is typically asymmetric; one user may trust someone else more than she is trusted back. Therefore, the second challenge is how to characterize such an asymmetric property in social trust.

1.2.3 Challenges in Signed Network Embedding

Though GCNs have been proved to be capable of effectively and efficiently encoding the structural information of the network and node feature information jointly, for those unsigned networks (networks contain positive links only) [31,41]. In the context of signed directed networks representation learning, excellent opportunities may exist in the use of the GCNs to capture the node features and the graph structure, including sign and direction information of the links.

However, representation learning in signed networks using graph convolutional neural networks is quite challenging. GCNs were originally designed to capture the homophily nature of unsigned networks [31, 41], indicating that more similar users are more likely to connect with each other. The notion of homophily is not applicable when both positive and negative links are jointly considered for signed network representation learning. In this context, the first challenge is how the network connections and associated sign information can be jointly represented in the embedding space. Though signed GCNs were proposed to address the problem of representation learning on signed networks (e.g., SGCN [19]), they were designed to use structural balance theory where the direction information of links are not explicitly aggregated and propagated, thus may not work very well in signed directed networks. Therefore, the second challenge is how to effectively characterize the direction information of links on signed directed networks.

1.2.4 The Robustness of Graph Convolutional Networks

Experimental evaluation results demonstrated that our approaches effectively achieve the best possible performance of analyzing social networks in the literature, in terms of accuracy, efficiency, and scalability. However, as effective as they may be, recent studies have also shown that neural networks, in general, are vulnerable to malicious adversaries, who are able to craft specific sets of *adversarial examples* so that neural network models will generate desired outputs of their choice. Typically, these selected

adversarial inputs are derived from regular inputs by introducing minor — yet carefully selected — perturbations. Such adversarial attacks have been widely demonstrated with high success rates in the contexts of image recognition [10] and malware detection [27]. Interestingly, it remains unclear how effective such adversarial attacks may be for link prediction algorithms based on graph neural networks.

Link prediction refers to the problem of identifying the existence of a link between two nodes in a network [54]. It is an essential problem with practical applications in a diverse set of research fields, including friend recommendation in online social networks [20], prediction, and ranking algorithms in complex networks (e.g., co-authorship graphs) [63], and criminal networks [9]. In criminal networks, for example, links between entities indicate that potential connections between these entities exist, such as having commercial ties or memberships in the same criminal organization. These potential links provide useful underlying information about network structures and may be readily detected by link prediction algorithms.

Adversarial perturbations on the graphs underlying complex networks, especially on social networking sites, are easily conceivable and quite common in practice. As link prediction may reveal connections which associated parties prefer to keep hidden – either for the sake of profit or to evade the law enforcement. For example, in online recommendation systems, fraudsters frequently manipulate online reviews to affect reader opinion in recommendation networks [15]. In a criminal network, criminals may try to hide their links to bypass the detection of criminal groups [9, 60]. Therefore, in the third part of this dissertation, we move our focus on the adversarial attacks on graph-structured data, particularly mounting attacks for link prediction algorithms based on graph neural networks.

1.2.5 Challenges on Attacking a Graph

Attacking the graph in a complex network effectively involves several non-trivial challenges. *First*, due to the inherent learning characteristics of graph convolutional networks, our problem of crafting perturbations on graph data contains dependent variables as the adjacency matrix and node information matrix are coupled, and existing solutions of gradient-based approaches are not applicable. *Second*, unlike existing adversarial attacks in the domain of image recognition [10] consisting of continuous data, graph-structured data are typically discrete and combinatorial. With the data's discreteness and the large number of model parameters of graph neural networks, solving the optimization problem for a complex network graph is highly challenging. *Finally*, adversarial perturbations — such as adversarial images in the context of image recognition — should not be noticeable by humans in general. Yet, in complex network graphs, the notion of "unnoticeable changes" needs to be clearly defined first.

1.3 Contributions

In general, the objectives of this dissertation are to investigate the problem of online social network analysis with a focus on trust evaluation of any two users, representation learning when the social networks are represented as signed network graphs. To be able to address the robustness of used techniques, graph convolutional networks, we also study the adversarial attacks on graph-structured data, particularly mounting attacks against link prediction based on graph convolutional neural networks. Following such objectives, this dissertation makes the following contributions.

1.3.1 Evaluating Trust in Online Social Networks

We first present *Guardian*, to address the challenges in social trust evaluation based on graph convolutional neural networks. With *Guardian*, we are able to effectively and efficiently estimate the value of trustworthiness between any two users who are not explicitly

connected, given the social network structure and associated trust relationships between users. *Guardian* is an end-to-end framework that stacks multiple trust convolutional layers, which is designed to discover hidden and predictive latent factors of trust in online social networks.

The key component of *Guardian* is the trust convolutional layer, which employs the notion of localized graph convolutions [31]. It is designed to capture the propagative nature and composable nature of social trust. The parameters to be learned in each layer are shared across all users, making the parameter complexity of our proposed framework independent of the size of the input network graph. In particular, in order to capture the asymmetric property of the social trust, each of our trust convolutional layers consists of two components: popularity trust propagation and engagement trust propagation. The former is used to learn the extent that a user is trusted by the others, while the latter is for capturing the willingness that a user trusts the others. Finally, by stacking a fully-connected layer, *Guardian* is able to explicitly represent both popularity trust and engagement trust of individual users in a collaborative manner. As such, effective pairwise trust relationships can be established. Extensive experimental results on benchmarking datasets demonstrated that *Guardian* can speedup trust evaluation by up to $2,827\times$ with comparable accuracy, as compared to the state-of-the-art in the literature.

1.3.2 Signed Network Embedding Based on Status Theory

Then we propose *SiGCN*, to address the challenges in representation learning of signed directed networks based on status theory via graph convolutional neural networks. With *SiGCN*, we can effectively and efficiently learn low-dimensional dense representations for users, such that the representations of users in the embedding space are effective for downstream signed network analysis. *SiGCN* is a new framework that learns representations of users for signed directed networks with graph convolutional neural networks. *SiGCN* is designed based on *status theory*, a fundamental theory from social psychology

that provides an organizing principle of signed links for signed directed networks [44]. In signed directed networks, social status can be represented in many different ways, such as the rankings of nodes in social networks, and it represents the prestige/trustworthiness of nodes [73]. These relative levels of status can be propagated and aggregated throughout the networks.

The key component of *SiGCN* is the status convolutional layer, which employs the notion of localized graph convolutions [31,41]. In order to capture the direction information of links, each of our status convolutional layers consists of two components: *receptive-based status* aggregation and *generative-based status* aggregation. A plausible analogy for these two components can be represented in the context of social networks: receptive-based status aggregation is used to characterize the status that a user endorsed by the others (e.g., the popularity of the user), while generative-based status aggregation is for capturing the extent that a user is willing to endorse to the others (e.g., the deference of the user to the others). By doing so, *SiGCN* can distill comprehensive information from the diverse relationships in a signed graph based on the theory of status. Further, by stacking fully-connected layers, *SiGCN* can obtain a status score for each node. The relative relationship between any two entities, therefore, can also be established with their status scores.

An extensive array of experimental results on benchmarking datasets demonstrated that *SiGCN* can speedup representation learning for link sign prediction by up to 6.5× as compared with the baselines. More specifically, *SiGCN* achieves up to a 4× speedup, as well as comparable accuracy as compared to BESIDE [16]. *SiGCN* also increases its accuracy by up to 18.8% compared with SIDE [39], and achieves state-of-the-art robustness and scalability compared to the literature. We also show that *SiGCN* can learn effective status scores of each node, which can be used for link sign prediction and node ranking and yield state-of-the-art performance.

1.3.3 Adversarial Attacks on Link Prediction Algorithms

Finally, we mount adversarial attacks on link prediction via applying existing evasion attacks in adversarial machine learning, to systematically study the ability of an "adversary" to manipulate link prediction. We first formulate the problem of crafting adversarial examples to deceive graph convolutional neural networks-based link prediction models as an optimization problem. In particular, we focus on evasion attacks against a state-of-the-art link prediction algorithm, called SEAL [89], which learns missing/unobserved links from local enclosing subgraphs. Essentially, SEAL is proposed based on a Υ-decaying heuristic theory, which shows that graph structure features can be well approximated from the local subgraphs and is able to unify a wide range of heuristics in a single framework. In this regard, we can envision that the mounted attacks may be transferred to the heuristics which can be well incorporated into the Υ-decaying heuristic framework.

Inspired by Zugner *et al.* [95], we first propose a greedy heuristic that perturbs the network graph incrementally by manipulating the graph structure. We then propose an efficient variant that utilizes the link formation mechanism and the Υ-decaying heuristic theory. To validate the effectiveness of our crafted attacks, we use real-world datasets to perform an extensive array of experiments. Our results have shown convincing evidence that the performance of link prediction in SEAL has been negatively affected by a significant margin using our adversarial attack, even with very limited knowledge of complex network graphs. More importantly, our experimental results have also shown that our adversarial attack can be readily transferred to several link prediction heuristics in the literature.

1.4 Overview and Roadmap

The remainder of this dissertation is organized as follows.

Chapter 2 introduces the necessary knowledge about network representation learn-

ing/network embedding, and graph convolutional networks.

Chapter 3 presents our study on the problem of evaluating trust in online social networks with graph convolutional networks. This chapter is based on our work published in IEEE International Conference on Computer Communications [47], collaborated with Zhaolin Gao, and Baochun Li.

Chapter 4 presents our study on the problem of signed network embedding based on status theory with graph convolutional networks. This chapter is based on our work submitted to ACM International Conference on Information and Knowledge Management [49], collaborated with Baochun Li.

Chapter 5 extends our scope to the context of adversarial attacks on link prediction algorithms based on graph convolutional networks. This chapter is based on our work published in ACM ASIA Conference on Computer and Communications Security [48], collaborated with Shengxiang Ji, and Baochun Li.

Chapter 6 concludes this dissertation, with a summary of our work and a discussion on future directions.

Chapter 2

Preliminaries

In this chapter, we review some important background material regarding key concepts from graph neural networks and network embedding/representation learning. This material is included in a separate introductory chapter, since it forms the basis for much of the development in the remainder of this dissertation. Other background knowledge — such as social-psychological theories used in mining signed networks, heuristic link prediction algorithms, graph neural network-based link prediction, adversarial attacks, and their transferability — is more localized to particular chapters in the dissertation. This chapter is intended to focus only on the minimal subset of ideas required to understand our contributions and discussion in the remainder of this dissertation, rather than to provide a comprehensive overview of the topics in this dissertation.

2.1 Representation Learning on Graphs

The goal of network representation learning is to learn low-dimensional representations for all nodes, which can be used for many different tasks of network analysis, such as link prediction [89], node classification, and community detection.

Network Embedding/Network Representation Learning: Formally, given a network $\mathcal{G} = (\mathcal{V}, \mathcal{A})$[1], the task of network representation learning is to learn a mapping function

[1]We are aware of the drawbacks of elaborate notations. Each chapter is a self-contained unit, in which all the relevant notations are fully explained. Although, whenever appropriate, individual notations are

$f : v \rightarrow x_v$, where $x_v \in \mathcal{R}^d$ is the learned representation of user v with dimension d. The transformation function f preserves the original network information, including network structure and node features, such that any representations of users, in the embedding space, are effective for downstream network analysis.

To transform networks from original network space to embedding space, different models can be adopted to incorporate different types of information or address different goals. The commonly used models include matrix factorization, random walk (e.g, node2vec [28], Line [74], and DeepWalk [66]), graph neural networks and their variations. Among these models, walk-based approaches and matrix factorization-based approaches usually suffer from several drawbacks. Firstly, the number of model parameters grows linearly with the size of the input graph. In other words, they may be inefficient for large graphs. Secondly, these methods lack the ability of generalization, which means that they could not be directly used in dynamic graphs or be generalized to new graphs. In this dissertation, we focus on graph neural network-based approaches, which are computationally efficient and can be generalized to new graphs/unseen nodes.

We do not attempt to provide a comprehensive literature review on graph neural network-based representation learning models. Instead, we selectively provide the baseline methods adopted by top performers on node classification tasks, such as graph convolutional networks proposed by Kipf&Welling [41] and GraphSage [31], an inductive framework for large graphs, either in terms of their simplicity or expressiveness. Furthermore, we think that these methods are of great value, not the least because they lead to state-of-the-art performance in the literature and can readily be applied to many tasks learning with graph-structured data.

---

used in more than one chapter, the corresponding definitions will be repeated in each of the chapters concerned.

2.2 Graph Convolutional Networks

Graph neural networks are designed specifically to generalize existing neural networks for processing graph-structured data. Formally, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{X})$ is given with $n = |\mathcal{V}|$ vertices, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ is the set of vertices, $\mathcal{A} \in \{0,1\}^{n \times n}$ is the adjacency matrix representing the connections, and $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}^T \in \mathcal{R}^{n \times m}$ is the feature matrix of vertices, and $x_v \in \mathcal{R}^m$ is the m-dimensional feature vector of vertex v.

Given a set of labeled nodes $\mathcal{V}_l \subset \mathcal{V}$, with class labels from $\mathcal{Y} = \{y_1, y_2, y_3, \cdots, y_K\}$ and a set of unlabeled nodes $\mathcal{V}_u \subset \mathcal{V}/\mathcal{V}_l$, the goal of node classification is to map each node $v \in \mathcal{V}$ to one class in $\mathcal{Y}$.

Graph convolutional neural networks (GCNs) is a generalization of traditional convolutional neural networks to the graph domain. In [41], the GCN model applied for semi-supervised classification is a two-layer GCN followed by a softmax classifier on the output features:

$$Z = \mathbf{softmax}(\hat{\mathcal{A}}\mathbf{ReLU}(\hat{\mathcal{A}}\mathcal{X}\Theta^{(0)})\Theta^{(1)}) \tag{2.1}$$

where $\tilde{\mathcal{A}} = \mathcal{A} + I$, $\tilde{\mathcal{D}}_{ii} = \sum_j \tilde{\mathcal{A}}_{ij}$, $\hat{\mathcal{A}} = \tilde{\mathcal{D}}^{-\frac{1}{2}}\tilde{\mathcal{A}}\tilde{\mathcal{D}}^{-\frac{1}{2}}$, $\mathbf{softmax}(x_i) = \frac{1}{Z}\exp(x_i)$ with $Z = \sum_i \exp(x_i)$. The optimization loss function is defined as the cross-entropy error over all labeled samples:

$$\mathcal{L}_{\mathbf{ce}} = -\sum_{i \in \mathcal{V}_l}\sum_{k=1}^{K} Y_{ik} \ln Z_{ik} \tag{2.2}$$

where $\mathcal{V}_l$ is the set of node indices that have labels, and K is the number of classes/labels.

Discussion In essence, for semi-supervised learning to work, a certain assumption, called the *smoothness assumption*, has to hold. It implies that if two inputs x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2. Semi-supervised GCN has been proved to perform very well on many classification tasks. These can be explained as that graph convolution is a special form of Laplacian smooth-

ing, which computes the new representation of a vertex by averaging over itself and its neighbors.

2.3 Inductive Representation Learning on Graphs

In [41], the GCN model is inherently transductive that focuses on embedding nodes from a single fixed graph, and does not naturally generalize to unseen nodes. GraphSage, by contrast, is an inductive framework that leverages node feature information (e.g., text attributes) to efficiently generate node embeddings for previously unseen data.

This inductive representation learning algorithm was given by Hamilton&Ying [31] and its forward process is described in **procedure GraphSage**. Essentially, GraphSage follows a neighborhood aggregation strategy, where it iteratively updates the representation of a node by aggregating representations of its neighbors. After l iterations of aggregation, a node's representation captures the structural information within its l-hop network neighborhood. We call the operator for aggregating information from the local graph as aggregation operator, or aggregator. The choice of aggregator, denoted as **AGGREGATE**, is crucial. A number of architectures for **AGGREGATE** have been proposed.

In the pooling variant of GraphSage, **AGGREGATE** has been formulated as:

$$\mathbf{AGGREGATE}_l^{\mathbf{pool}} = \mathbf{max}\left(\{\sigma(W_{\mathbf{pool}}h_u^l + b), \forall u \in \mathcal{N}(v)\}\right) \tag{2.3}$$

where **max** denotes the element-wise **max** operator and σ is a nonlinear activation function. By applying the max-pooling operator to each of the computed features, the model is able to capture different aspects of the neighborhood set. Note that, any symmetric vector function, such as element-wise **mean** operator, could be used in place of the **max** operator.

For mean aggregator, the node embedding update process (line $5-6$ in **procedure GraphSage**) can be replaced as:

$$h_v^l \leftarrow \sigma\left(W \cdot \textbf{MEAN}(\{h_v^{l-1}\} \cup \{h_u^{l-1}, \forall u \in \mathcal{N}(v)\}\right) \tag{2.4}$$

1: **procedure GraphSage**: EMBEDDING GENERATION (I.E, FORWARD PROPAGATION)

INPUT: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; input features x_v, $\forall v \in \mathcal{V}$; depth L, weight matrices $\mathcal{W}^l$, $l \in [1, L]$; non-linearity σ; differentiable aggregator functions $\textbf{AGGREGATE}_l, l \in [1, L]$; neighborhood function $\mathcal{N} : v \rightarrow 2^{|\mathcal{V}|}$

OUTPUT: Vector representations z_v, $\forall v \in \mathcal{V}$

2: $h_v^0 \leftarrow x_v$, $\forall v \in \mathcal{V}$

3: **for** $l = 1 \cdots L$ **do**

4: **for all** $v \in \mathcal{V}$ **do**

5: $h_{\mathcal{N}(v)}^l \leftarrow \textbf{AGGREGATE}_l\left(\{h_u^{l-1}, \forall u \in \mathcal{N}(v)\}\right)$;

6: $h_v^l \leftarrow \sigma\left(W^l \cdot \textbf{CONCAT}(h_v^{l-1}, h_{\mathcal{N}(v)}^l)\right)$

7: $h_v^l \leftarrow h_v^l / ||h_v^l||_2, \forall v \in \mathcal{V}$

8: $z_v \leftarrow h_v^L, \forall v \in \mathcal{V}$

Instead of training individual embeddings for each node, **GraphSage** learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. The parameters of **GraphSage** can be learned in a fully unsupervised setting. To do so, a graph-based loss function is applied to the output representations, $z_v, \forall v \in \mathcal{V}$, and the weight matrices, $W^l, \forall l \in \{1, ..., L\}$, and parameters of the aggregator functions are tuned via stochastic gradient descent. For supervised setting, the parameters can be learned according to the classification cross-entropy loss, shown in Eq. 2.2.

Applications Graph neural networks are widely being used in various applications, such as social network prediction [19, 31, 41], recommendation systems [21], and traffic forecasting [80].

Chapter 3

Evaluating Trust in Online Social Networks

Due to the popularity and open nature of online social networks, *social trust* has become an important concern in online social networks. Therefore, it is helpful to evaluate the pairwise trust relationship between any two users who are not directly connected within online social networks. In this chapter, we propose *Guardian*, a new end-to-end framework that learns latent factors in social trust with graph convolutional neural networks. *Guardian* is designed to incorporate social network structures and trust relationships to estimate social trust between any two users.

Highlights of our original contributions in this chapter are as follows. *First*, we introduce a principled methodology to jointly capture both social connections and associated trust relationships of the users within online social networks. *Second*, we propose a new approach to jointly characterize the popularity trust and engagement trust of users so that the asymmetric property of the social trust can be captured implicitly. *Third*, we demonstrate the effectiveness and efficiency of our proposed framework using two online social networks from different domains — Advogato and Pretty Good Privacy. Our extensive array of experiments on benchmarking datasets demonstrated that *Guardian* can speedup trust evaluation by up to $2,827\times$ with comparable accuracy as compared to NeuralWalk [51], and increase accuracy by up to 18.8% and 19.8% compared with

Matri [85] and OpinionWalk [50], respectively.

The remainder of this chapter is organized as follows. We first introduce some background, and formulate social trust evaluation problem in Sec. 3.1. In Sec. 3.2, we present the details of our framework designed to evaluate social trust in online social networks effectively and efficiently. In Sec. 3.3, we present an extensive array of experimental results to evaluate the performance of our framework. Sec. 3.4 discusses related work and Sec. 3.5 concludes this chapter.

3.1 Problem Setup

We consider a social trust evaluation problem in an online social network, which is modeled as a directed graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where any vertex $u, v \in \mathcal{V}$ represent users, and $e_{u\to v} \in \mathcal{E}$ denotes the observed trust relationships. $w_{u\to v}$ measures the trustworthiness of the trustor-trustee pair $\langle u, v\rangle$, where the trustworthiness domain is typically application-specific. For example, in Epinion[1], $w \in \{\text{Trust, Distrust}\}$, while in Advogato[2] and in Pretty-Good-Privacy[3] (PGP), $w \in \{\text{Observer, Apprentice, Journeyer, Master}\}$. Let $\mathcal{W} = \{\langle u, v\rangle, w_{u\to v} | e_{u\to v} \in \mathcal{E}\}$ be the set of observed trust relationships in the given online social graph. $\tilde{\mathcal{W}} = \{\langle u, v\rangle, \tilde{w}_{u\to v} | \tilde{e}_{u\to v} \notin \mathcal{E}\}$ denotes the set of unobserved/missing trust relationships that are to be evaluated.

Notably, as in most existing online social networks, trustworthiness is represented by categorical values. In this context, the social trust evaluation problem is equivalent to a social trust prediction problem. We can define $|w|$ to be the total number of types of trustworthiness, which is application-specific. For example, in PGP or Advogato, $|w| = 4$.

Before we formulate the problem of pairwise social trust evaluation, we introduce some important notations and necessary properties of social trust to facilitate a better understanding of the problem and our solution. For any user $u \in \mathcal{V}$, let $N_O(u)$ be the

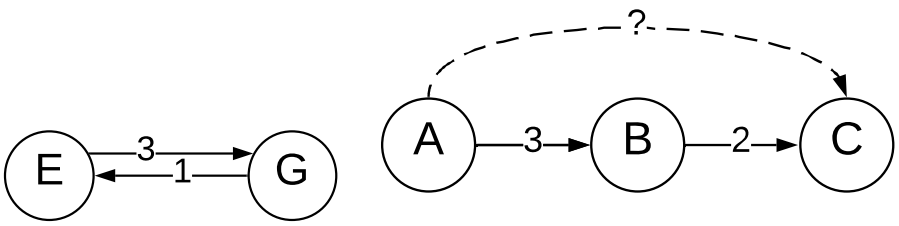
[1] https://snap.stanford.edu/data/soc-Epinions1.html
[2] http://www.trustlet.org/datasets/advogato/
[3] http://networkrepository.com/arenas_pgp.php

Table 3.1: Notations

Notation	Descriptions
$w_{u \to v}$	the trustworthiness of v from the perspective of u
$\mathcal{W}$	the set of observed pairwise trust relationships
$\|w\|$	the number of trustworthiness types
$N_O(u)$	the set of observed trustees whom u endorses her trust on (out-neighbors of u)
$N_I(u)$	the set of observed trustors who endorse trust on u (in-neighbors of u)
$x[u]$	initial embedding of user u
D_e	the dimension of initial embedding vector
pTr, eTr	the popularity trust and the engagement trust
$h_I[u]$	the latent factor of the popularity trust from in-neighbors $N_I(u)$ of user u
$h_O[u]$	the latent factor of the engagement trust from out-neighbors $N_O(u)$ of user u
$h[u]$	the trust latent factor of user u
$\tilde{h}_{u \to v}$	the pairwise trustworthiness latent factor
$\tilde{w}_{u \to v}$	predicted trust relationship
$\otimes$	the concatenation operator of two vectors
$\oplus$	the mean aggregator
σ	non-linear activation functions, e.g., $\tanh(\cdot)$, $\text{softmax}(\cdot)$
W, b	the model parameters (weight matrices and bias) in *Guardian*

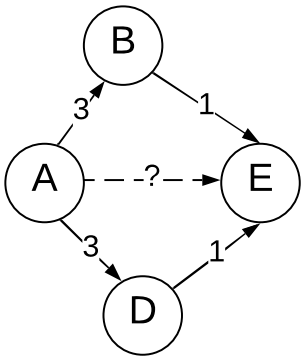
set of observed trustees whom u endorses her trust on (out-neighbors of u), $N_I(u)$ be the set of observed trustors who endorse trust on u (in-neighbors of u). In this sense, we can define $|N_I(u)|$ and $|N_O(u)|$ to represent in-degree and out-degree of u, respectively. The mathematical notations used in this chapter are summarized in Table 3.1.

In the literature [71], widely used trust properties include the propagative nature, composable nature, and asymmetric property. For the sake of clarity, we use the same example in Chapter 1 to illustrate the properties of social trust, which will also be used throughout this chapter.



(a) Asymmetry: the trustworthiness of user G from the perspective of user E (E → G) is different from that of E from G (G → E), even though they endorse trust explicitly to each other.

(b) Propagative nature: user A trusts user B with a trust value of 3 and user B trusts user C with 2, user A trusts user C with 2. In this example, A → B → C forms a trust chain for A → C.



(c) Composable nature: to establish a trust relationship for A → E, there exist two trust chains that need to be aggregated. Because both the trust value for B → E and D → E are 1, it is unlikely for A → E to achieve a high trust value.

Figure 3.1: The property illustrations of social trust: an example.

Fig. 3.1b illustrates the propagative nature of social trust. Since user A trusts user B with a trust value of 3 and user B trusts user C with 2, user A trusts user C with 2. In this example, A → B → C forms a trust chain (path) for A → C. To establish a trust relationship for A → E as shown in Fig. 3.1c, there exist two trust chains that need to be aggregated. Because both the trust value for B → E and D → E are 1, it is unlikely for A → E to achieve a high trust value. In other words, while evaluating the trust value between any two users, the trust chains in between should be considered and

aggregated in an effective way, such as the weighted mean. The asymmetric property of the social trust can also be illustrated in Fig. 3.1a: the trustworthiness of user G from the perspective of user E (E $\rightarrow$ G) is different from that of E from G (G $\rightarrow$ E), even though they endorse trust explicitly to each other.

With the aforementioned notations and definitions, we can now formally define the problem of *social trust evaluation* (or *prediction*). Given an online social network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, the social trust evaluation problem aims to evaluate (or *predict*) the trustworthiness of the trustor-trustee pair $\tilde{w}_{u \rightarrow v}$, where $u, v \in \mathcal{V}$, $u \neq v$ and $\tilde{e}_{u \rightarrow v} \notin \mathcal{E}$.

3.2 *Guardian*: Proposed Framework

We now present *Guardian*, our proposed framework for social trust evaluation, the architecture of which is illustrated in Fig. 3.2. There are three components in the framework: (1) an embedding layer that offers an initialization of user embeddings; 2) multiple trust convolutional layers that refine the popularity trust embedding and engagement trust embedding by injecting high-order social trust relationships; and 3) a prediction layer that consists of a fully-connected layer followed by a softmax function. It first transforms the latent representations of users into the latent factor of trust, and then outputs the probability of the prediction. In what follows, we first conceptually discuss the efficiency and effectiveness of our proposed framework, and then discuss more influential factors for social trust evaluation and limitations of our framework.

3.2.1 Embedding Layer

With the recent emergence of representation learning, the network embedding technique has been extensively studied to discover and encode network structural properties into a low-dimensional latent space. More formally, network embedding learns a representation vector $x[u] \in \mathcal{R}^{D_e \times 1}$ for each user u in the network graph $\mathcal{G}$. In *Guardian*, we use a pre-trained embedding layer to map each user into a D_e-dimensional representation. To

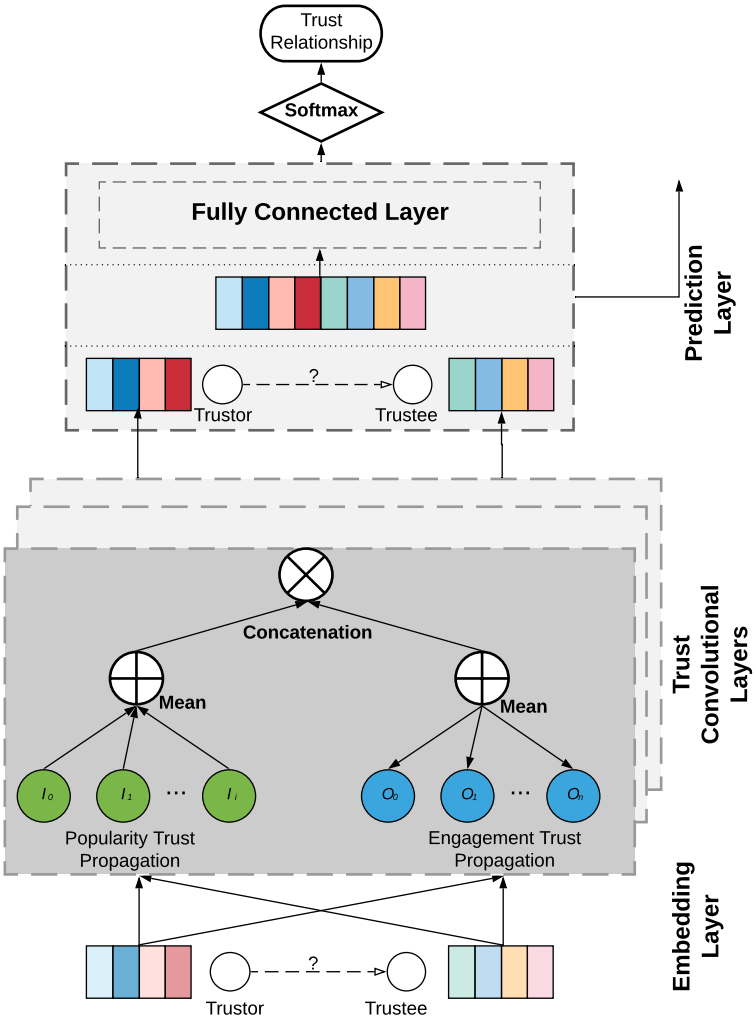


Figure 3.2: Illustration of *Guardian* framework: (1) an embedding layer that offers an initialization of user embeddings; 2) multiple trust convolutional layers that refine the popularity trust embedding and engagement trust embedding by injecting high-order social trust relationships; and 3) a prediction layer that consists of a fully-connected layer followed by a softmax function. To optimize *Guardian*, the full pipeline in our architecture is used.

be more specific, user embeddings are generated based on users' social friendship, which were created whenever two users interact. As indicated by social correlation theories [55, 57], users' social behaviors, such as interactive behaviors with others, are similar to or influenced by their directly connected friends. Widely used pre-trained models for user embeddings are node2vec [28] and DeepWalk [66].

It is worth noting that these representations serve as an initial state for user embed-

dings, to be optimized in an end-to-end fashion. In *Guardian*, we refine the embeddings by propagating them along the online social network graph. With a specially designed transformation layer, these refined user embeddings can be transformed into pairwise trustor-trustee embeddings for trustworthiness prediction.

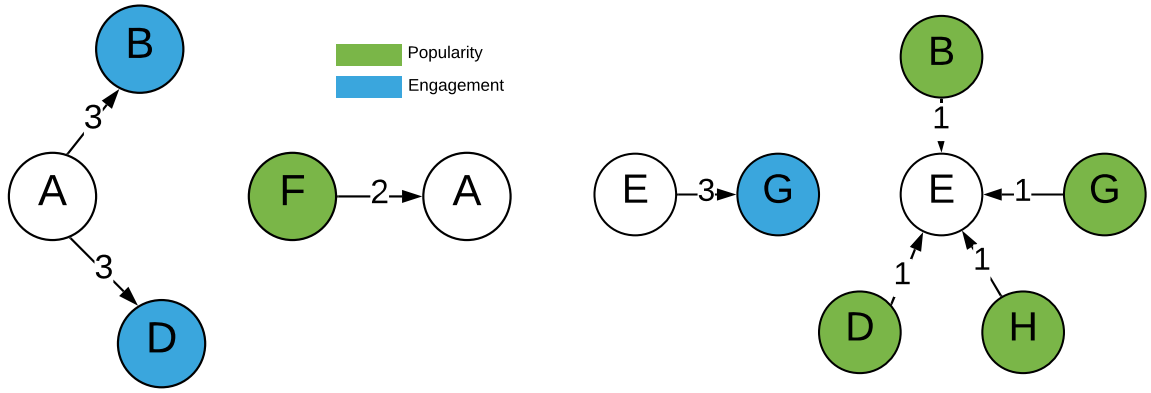
3.2.2 Trust Convolutional Layers

As an online social network graph contains not only social connections between users but also trust interactions between the trustors and the trustees, we provide a principled approach to jointly capture the social connections and associated trust relationships for learning the embeddings $h[u]$ of the users. In particular, due to the asymmetric property of social trust, a user can assume different roles, either as a trustor or a trustee. To be able to capture the asymmetric property of social trust, we first separate pairwise trust interactions into two groups: *popularity interactions* and *engagement interactions*. Popularity-based interactions refer to the trustworthiness of a user as observed by the others. In this sense, the more a user is trusted by the others, the more popularity-based trust this user gains. Similarly, engagement-based interactions refer to the trustworthiness of the others from a user's perspective. The popularity trust indicates the extent that a user is trusted by the others, while the engagement trust reveals the willingness that a user trusts the others.

In what follows, we consider two types of trust aggregation to characterize the popularity trust and engagement trust, represented as $h_I[u]$ and $h_O[u]$, respectively. For each of them, we use mean-aggregator to aggregate its associated trust interactions with its neighbors. It is worth mentioning that, mean-aggregator is the main operation of aggregating information from local graph neighborhoods [31, 41].

Let's see an example in our example social network graph, originally shown in Fig. 1.1. With our trust model, the popularity interactions of user A and E are depicted in green in Fig. 3.3, while the engagement interactions are shown in blue. More specifically, for

user E, there are four incoming neighbors, all of which have a trust value of 1. The popularity trust of E is, therefore, 1 by averaging over its incoming trust relationships, and the engagement trust of E is 3. Similarly, the popularity trust of A is 2, as there is only one incoming neighbor with trust value 2, while its engagement trust is 3, averaging over its two outgoing trust relationships.



(a) The popularity trust of A is 2, as there is only one incoming neighbor with trust value 2, while its engagement trust is 3, averaging over its two outgoing trust relationships.

(b) For user E, there are four incoming neighbors, all of which have a trust value of 1. The popularity trust of E is, therefore, 1 by averaging over its incoming trust relationships, and the engagement trust of E is 3.

Figure 3.3: The popularity trust and the engagement trust: an example. With our trust model, the popularity interactions of user A and E are depicted in green, while the engagement interactions are shown in blue.

***P**opularity **Tr**ust Propagation* (pTr). Intuitively, the incoming social connections and associated trust relationships provide direct evidence on the popularity trust of a user in online social networks. We build upon this basis to propagate the popularity trust between connected users.

In particular, to model categorical trustworthiness, we first use one-hot encoding to represent each type of trustworthiness. Taking the Advogato dataset as an example, for trustworthiness $w_{u \to v} \in \{\text{Observer, Apprentice, Journeyer, Master}\}$, we model them as the following one-hot representations: $[0,0,0,1]^T$, $[0,0,1,0]^T$, $[0,1,0,0]^T$, and $[1,0,0,0]^T$. Then *Guardian* employs a linear transformation to convert the one-hot encodings into dense vector embeddings through Eq. (3.1) and Eq. (3.4). For a trust relationship with

trustworthiness $w_{u\leftarrow v}$ (u is being a trustee in this trust relationship), we model the popularity trust of u as observed by v with a combination of v's embedding $x[v]$ and the embedding of associated trustworthiness $e_{w_{u\leftarrow v}}$.

$$e_{w_{u\leftarrow v}} = W_{u\leftarrow v} \cdot w_{u\leftarrow v} \tag{3.1}$$

$$\mathrm{pTr}_{u\leftarrow v} = x[v] \otimes e_{w_{u\leftarrow v}} \tag{3.2}$$

where $W_{u\leftarrow v} \in \mathcal{R}^{D_e \times D_{|w|}}$ is a trainable transformation matrix, $\otimes$ denotes the concatenation operation between two vectors, and $|w|$ denotes the number of trustworthiness types.

We now take the element-wise mean of the vectors in $\{\mathrm{pTr}_{u\leftarrow v}, \forall v \in N_I(u)\}$. This mean-based aggregator is a linear approximation of a localized spectral convolution [41], as the following function:

$$h_I[u] = \frac{1}{N_I(u)} \cdot \sum_{v \in N_I(u)} \mathrm{pTr}_{u\leftarrow v} \tag{3.3}$$

***E**ngagement **Tr**ust Propagation* (eTr). Accordingly, we characterize the engagement trust of a user through its outgoing social connections and associated trust relationships. We build upon this basis to perform the propagation and aggregation of engagement trust between the connected users. Thus, the engagement trust of user u can be captured by the following functions:

$$e_{w_{u\rightarrow v}} = W_{u\rightarrow v} \cdot w_{u\rightarrow v} \tag{3.4}$$

$$\mathrm{eTr}_{u\rightarrow v} = x[v] \otimes e_{w_{u\rightarrow v}} \tag{3.5}$$

$$h_O[u] = \frac{1}{N_O(u)} \cdot \sum_{v \in N_O(u)} \mathrm{eTr}_{u \to v} \tag{3.6}$$

where $\mathrm{eTr}_{u \to v}$ denotes the involvement trust of the user u to user v in online social networks.

Learning Trust Latent Factors of Users. In order to learn better latent factors of users for downstream trustworthiness prediction, the popularity trust and engagement trust are needed to be considered jointly. Here, we propose to combine these two types of trust through a standard fully connected (FC) layer, where $h_I[u]$ and $h_O[u]$ are concatenated before feeding into the FC. Formally, the latent factor of user u, $h[u]$, can be characterized as follows:

$$h[u] = \sigma\left(W \cdot (h_I[u] \otimes h_O[u]) + b\right) \tag{3.7}$$

where W is a trainable transformation matrix, b is a learnable bias, and σ denotes the non-linear activation function.

Higher-order Trust Propagation. By stacking l trust convolutional layers, a user is capable of receiving the social trust (the popularity trust and engagement trust) propagated from its l-hop neighbors. In the l-th step, the representation of user u is recursively formulated as Eq. (3.8) - Eq. (4.12):

$$\mathrm{pTr}^l_{u \leftarrow v} = h^{l-1}[v] \otimes \{W^l_{u \leftarrow v} \cdot w_{u \leftarrow v}\} \tag{3.8}$$

$$\mathrm{eTr}^l_{u \to v} = h^{l-1}[v] \otimes \{W^l_{u \to v} \cdot w_{u \to v}\} \tag{3.9}$$

$$h^l_I[u] = \frac{1}{N_I(u)} \cdot \sum_{v \in N_I(u)} \mathrm{pTr}^l_{u \leftarrow v} \tag{3.10}$$

$$h_O^l[u] = \frac{1}{N_O(u)} \cdot \sum_{v \in N_O(u)} \mathrm{eTr}_{u \to v}^l \tag{3.11}$$

$$h^l[u] = \sigma\left(W^l \cdot (h_I^l[u] \otimes h_O^l[u]) + b^l\right) \tag{3.12}$$

where $h^0[u] = x[u]$ is the pre-trained embedding of user u obtained in the embedding layer, $w_{u \to v}$ and $w_{u \leftarrow v}$ are the observed trust relationships, and $W_{u \leftarrow v}^l$, $W_{u \to v}^l$, W^l, and b^l are the model trainable parameters, to be optimized in an end-to-end fashion with *Guardian*. Note that, by stacking multiple trust convolutional layers, we not only enrich the initial user embedding with its propagated popularity trust and engagement trust in online social networks, but also allow controlling the range of trust propagation by adjusting l.

3.2.3 Prediction Layer

In order to learn the latent factor of trust relationship, we first concatenate the latent embeddings of the trustor and the trustee, and then fit them to a standard fully-connected (FC) layer followed by a softmax layer. Formally, the latent representation of the trustor-trustee pair is formulated as Eq. (3.13), where W_{fc} is a trainable weight matrix defined in the FC layer, and σ is the softmax function, defined as $\mathrm{softmax}(x_i) = \frac{\exp(x_i)}{Z}$ with $Z = \sum_i \exp(x_i)$.

$$\tilde{h}_{u \to v} = \sigma\left(W_{fc} \cdot (h[u] \otimes h[v])\right) \tag{3.13}$$

The advantage of using concatenation lies in its simplicity and expressiveness, which have been shown in a recent work of graph convolutional neural networks [31]. In addition, the fully connected layer leads to a more effective representation of a trust relationship for prediction, as this step explicitly injects the popularity trust and the engagement trust of individual users in a collaborative fashion. The outcome of this step is the probabilistic

prediction values of the trustworthiness. As a consequence, the trustworthiness of user v from the perspective of user u is computed as $\tilde{w}_{u\to v} = \underset{j}{\text{argmax}}(\tilde{h}_{u\to v})$. Note that $\tilde{w}_{u\to v} \neq \tilde{w}_{v\to u}$, due to the asymmetric property of social trust in online social networks. The detailed forward propagation algorithm of *Guardian* is shown as **procedure Guardian**.

3.2.4 Model Training

To learn the model parameters in *Guardian*, we define the objective function as the cross-entropy loss between the predicted values and the ground-truth trustworthiness from the observed set $\mathcal{W}$. Formally, it is formulated as:

$$\mathcal{L} = -\frac{1}{|\mathcal{W}|} \sum_{(\langle u,v\rangle, w_{u\to v})\in\mathcal{W}} \log \tilde{h}_{u\to v, w_{u\to v}} + \lambda \cdot ||\Theta||_2^2 \tag{3.14}$$

where $\mathcal{W} = \{\langle u, v\rangle, w_{u\to v}\}$ is the set of observed trustor-trustee pairs and associated trust relationships, $\Theta = \{\{W^l_{u\leftarrow v}, W^l_{u\to v}, W^l, b^l\}_{l=1}^{L}, W_{fc}\}$ denotes all trainable model parameters, and λ controls the L_2 regularization strength to prevent over-fitting. In particular, we adopt Adam [40] as the optimizer in our implementation, as it has been shown to be effective in updating the model parameters [31].

3.2.5 Analysis and Discussions

Different from the state-of-the-art trust evaluation solutions in the literature [50, 51], our framework does not have any assumptions on the existence of paths between the trustor and the trustee while we compute the pairwise trustworthiness values. This reflects the real-world situation where some of the users are new in the society and may not have any social connections with the other users. However, these newly added users are still able to trust the existing users who have a significant popularity trust (e.g., the authenticated/official users) to some extent. Surprisingly, *Guardian* can still achieve the best prediction accuracy even if we do not make any assumptions, which, as shown in Sec. 3.3, can be empirically verified later.

procedure Guardian: TRUST RELATIONSHIP PREDICTION (I.E, FORWARD PROPAGATION)
INPUT: Observed Network Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, $\mathcal{W} = \{\langle u, v\rangle, w_{u\to v}\}$
OUTPUT: The prediction vector of trust relationship $\tilde{h}_{u\to v}$, for all $\langle u, v\rangle \in \mathcal{W}$
Generate initial states of user embeddings for $\mathcal{G}$
$h^0[u] \leftarrow x[u]$, for all $u \in \mathcal{V}$
▷ Trust latent factors of observed users
for all $u \in \mathcal{V}$ **do**
  for $l = 1 \cdots L$ **do**
    ▷ Popularity Trust
    $h_I^l[u] = \frac{1}{N_I(u)} \cdot \sum_{i \in N_I(u)} \mathrm{pTr}_{u\leftarrow i}^l$
    ▷ Engagement Trust
    $h_O^l[u] = \frac{1}{N_O(u)} \cdot \sum_{i \in N_O(u)} \mathrm{eTr}_{u\to i}^l$
    $h^l[u] = \sigma\left(W^l \cdot (h_I^l[u] \otimes h_O^l[u]) + b^l\right)$
    ▷ Trust relationship prediction vector
for all $\langle u, v\rangle \in \mathcal{W}$ **do**
  $h[u] \leftarrow h^L[u]$
  $h[v] \leftarrow h^L[v]$
  $\tilde{h}_{u\to v} = \sigma\left(W_{fc} \cdot (h[u] \otimes h[v])\right)$

The key computational operations of our framework are the notion of localized graph convolutions [31]. To be able to implicitly capture the asymmetric property of social trust, each trust convolutional layer learns how to aggregate the popularity trust and engagement trust of users from a small graph neighborhood in the social graph. By applying multiple trust convolutional layers that aggregate the trust information from the local neighborhood of users, our approach can obtain the popularity trust and engagement trust of users from their local network topology.

Theoretically, our proposed framework inherits the capability of inductively generating embeddings for unseen users based on their features and local graph neighbors. At the prediction step, we are able to compute the embeddings for users that were not seen in the training phase. More specifically, the inductive capability allows us to train on a subgraph to obtain the model parameters. It is able to estimate the pairwise trustworthiness for users that were not seen during the training phase. Sec. 3.3 empirically verified that training on a subgraph containing 40% could achieve favorable performance, i.e., increases in the training set size did not seem to much help.

It is also worth mentioning that parameters of our proposed trust convolutional layers are shared across all users, making the parameter complexity of *Guardian* independent of the input graph size. It is easy to computer embedding of new users that get added into the graph over time, as it does not require any retraining process as the pre-trained parameters can be used for inference for the unseen users. The computation cost of our framework main comes from the localized graph convolutions, of which the complexity is coming from the model parameter complexity. Sec. 3.3 empirically verified the efficiency and scalability of our framework.

Incorporating context-aware features. Except for the social network graph and associated trust interactions, context is also an important influential factor for social trust evaluation [61]. In different contexts, trust relationships are typically different. For example, user A trusts user B for movie recommendations, while A may not trust B for restaurant recommendations. Movies and restaurants here represent different contexts. Therefore, it is crucial to distinguish between the different contexts of trust. Our framework can be readily extended to incorporate such context-aware features to further improve prediction accuracy, e.g, concatenating context features and graph structure embedding as the initial representation of a user.

Limitations. One important property of the social trust is that it is dynamic. More precisely, social trust can increase or decrease with new interactions and observations. It may also decay with time. A more recent interaction or observation may be more important than those that have happened earlier. It is intriguing to find out how our proposed framework responds to dynamics in social trust relationships, which will be left as our future work.

Table 3.2: Statistical Description of Advogato and PGP Datasets.

Dataset	# of Nodes	# of Edges	Avg. Degree	Diameter
Advogato	6,541	51,127	19.2	4.82
PGP	38,546	317,979	16.5	7.7

3.3 Experimental Evaluation

3.3.1 Description of Datasets Used

In our experiments, we choose two widely used, real-world and benchmarking datasets for performance comparisons of different trust evaluation models [51]. The first dataset is Advogato, which is an online social network for open source developers. To allow users to certify each other, this network provides four different levels of trustworthiness. More specifically, the types of trustworthiness are {Observer, Apprentice, Journeyer, Master}.

The second dataset is Pretty-Good-Privacy (PGP), an encryption program that provides cryptographic privacy and authentication for data communication by adopting the concept of "web of trust." Similarly, the web of trust in PGP dataset contains four different levels of trustworthiness. The statistics of these two datasets are presented in Table 3.2.

3.3.2 Experimental Settings

Baselines for comparisons. To demonstrate the effectiveness, we compared *Guardian*, our proposed framework against three groups of methods including traditional walk-based approach, matrix factorization-based approach, and deep neural network-based approach. For each group, we selected a representative baseline and below we will detail them. All experiments run 20 times to ensure statistical significance.

OpinionWalk [50]: This approach modeled the pairwise trustworthiness using statistical distributions in three-valued subjective logic. In order to establish a trust rela-

tionship between two indirectly connected users, it walked throughout the network in a breadth-first search manner. In particular, trust propagation and aggregation along the social paths were modeled with its predefined discounting and combining operators.

Matri [85]: This methodology was proposed to combine trust tendency and trust propagation under a collective matrix factorization framework. Under this framework, the trustor and trustee are mapped into a joint latent space. The trustworthiness of each trustor-trustee pair is modeled as the similarity (measured by the inner product of two vectors) between the latent vector of the trustor and the latent factor of the trustee in the learned latent space.

NeuralWalk [51]: This model was the state-of-the-art trust evaluation solution in the literature, in terms of its prediction accuracy. Its core is to learn single-hop trust propagation and aggregation rules with a neural network architecture, WalkNet. By iteratively executing the learning process of WalkNet multiple times, NeuralWalk is able to evaluate multi-hop social trust within online social networks.

Evaluation metrics. In order to evaluate the effectiveness of our proposed framework, two popular metrics were adopted to evaluate the prediction accuracy, including F1-score and Mean Absolute Error (MAE). All results are reported based on the results of 20 runs. Note that, larger values of F1-score, smaller values of MAE indicate better prediction accuracy. A small improvement in these evaluation metrics implies a significant influence on the quality of prediction. For efficiency and scalability, we used the average wall-clock time over 20 runs.

All the experiments were performed on a machine with Intel Core i7-9700K 8-core 3.6GHz CPU, 32GB RAM, 500GB SSD, and GeForce GTX 1660 Ti GPU.

Data preprocessing. We followed the data preprocessing as reported in NeuralWalk [51]. Specifically, as OpinionWalk is deductive, there is no need to separate the datasets for training and inference. Instead, we randomly selected $1,000$ trustor-trustee pairs for each dataset to statistically compare OpinionWalk with our framework. As for Matri,

NeuralWalk, and *Guardian*, we randomly split each dataset into two portions: 80% of the trustor-trustee pairs to constitute the training set, and the remaining 20% as the test set. More precisely, the 20% of trustor-trustee pairs were removed from the network graph to compose the training set.

OpinionWalk [50] and Matri [85] mapped four trustworthiness levels into scalar values, aka {Observer: 0.1, Apprentice: 0.4, Journeyer: 0.7, Master: 0.9}, and they used MAE as their performance metric. Similarly, to be comparable, we did the same mapping for NeuralWalk and *Guardian* (both are categorical classifiers) to obtain the model MAE. Regarding F1-score, the outputs of OpinionWalk and Matri were rounded to the nearest categorical values, aka {Observer: 0, Apprentice: 1, Journeyer: 2, Master: 3}.

As illustrated in Sec. 3.2, to model categorical trustworthiness, we used one-hot encoding to represent each type of trustworthiness. As the benchmark datasets we used all contain four different types of trustworthiness, we transformed {Observer, Apprentice, Journeyer, Master} as following one-hot representations: $\{[0,0,0,1]^T$, $[0,0,1,0]^T$, $[0,1,0,0]^T$, and $[1,0,0,0]^T\}$. Note that, our framework can be readily generalized to any application domains containing an arbitrary number of trustworthiness levels.

Parameter settings. We implemented our proposed framework in Pytorch[4]. node2vec [28] was used to generate the initial embeddings for each user[5]. The embedding dimension was fixed to 128 for all datasets. In terms of hyperparameters, we applied a grid search for hyperparameters: the learning rate was tuned amongst $\{0.001, 0.005, 0.01, 0.05\}$, the coefficient of L_2 normalization was searched in $\{10^{-5}, 10^{-4}\}$, and the dropout ratio was in $\{0.0, 0.1, \ldots, 0.8\}$. We used the Xavier initializer [23] to initialize the model parameters. In addition, early stopping strategy was performed, *i.e.*, premature stopping if training loss does not increase for 10 successive epochs. Without specification, we report the results of three trust propagation layers $[32, 64, 32]$, learning rate of 0.01, dropout ratio of 0.0 and normalization coefficient of 10^{-5}. The detailed parameter settings for

[4]https://pytorch.org

Table 3.3: Prediction Accuracy on Advogato

Approaches	F1-Score	MAE
Guardian	**74.3**%	0.082
NeuralWalk	74.0%	0.081
OpinionWalk	64.3%	0.228
Matri	65.6%	0.127

OpinionWalk, NeuralWalk, and Matri refer to [50, 51, 85], respectively.

3.3.3 Performance Comparisons

Effectiveness. The Advogato dataset is used to evaluate the performance of different approaches. The results are reported in Table 3.3. *Guardian* offers the best F1-score with 0.3% improvement on NeuralWalk — the state-of-the-art solution — and even higher improvement on Matri, about 8.7%. As F1-score is scaled between 0 and 1, the increases in performance are significant. In terms of MAE, NeuralWalk and *Guardian* achieved approximately the same prediction accuracy, which implies the powerful learning capability of machine learning techniques.

To test that *Guardian* does not rely on datasets, we also evaluated our framework on PGP. We were not able to report the performance of NeuralWalk on PGP, as it ran out of the memory after one out of three iterations on our machine. As shown in Table 3.4, *Guardian* consistently offers the best F1-score by increasing the accuracy 18.8% for Matri and 19.8% for OpinionWalk. The results reported successfully verify that our proposed trust convolutional layers are able to characterize the trust latent factors of users to establish effective social trust.

Matri was not able to offer comparable performance on two datasets, which indicates that either the collected matrix for factorization or the inner product in the learned latent

[5] As the benchmarking datasets do not contain context information, we do not consider context-aware feature in our experiments.

Table 3.4: Prediction Accuracy on PGP

Approaches	F1-Score	MAE
Guardian	**87.1%**	**0.083**
NeuralWalk	–	–
OpinionWalk	67.3%	0.249
Matri	68.3%	0.122

Table 3.5: Training and Inference Time (on the full test set)

Process	Approach	Advogato	PGP
Training Time (s)	*Guardian*	**28.580**	**304.370**
	Matri	176.395	1,593.285
Inference Time (s)	*Guardian*	0.102	0.583
	Matri	0.008	0.044

space of users may not be sufficient to capture the complex relations among the trustors and trustees. We also observed that OpinionWalk achieved the worst performance on both datasets, which shows that the path-search manner or the predefined trust propagation and aggregation rules may not be effective to provide accurate estimations.

Efficiency. For efficiency comparisons, we evaluated different approaches on the same machine as listed above. Because OpinionWalk is a deductive method - evaluating one trustor-trustee pair at a time, it does not generate any model parameters/user latent space for new trustor-trustee pair evaluation. In other words, the time for trust evaluation increases linearly with the number of trustor-trustee pairs to be evaluated. As such, we report the average runtime for evaluating 1,000 trustor-trustee pairs in Fig. 3.4 but exclude this method from the following discussions.

We compared the total runtime of three approaches (Matri, NeuralWalk and *Guardian*)

[6]Due to RAM issue, we are not able to reproduce NeuralWalk on PGP with our machine.

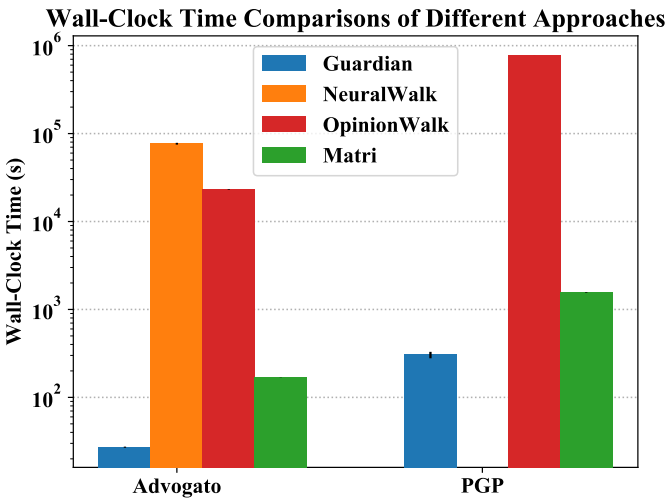
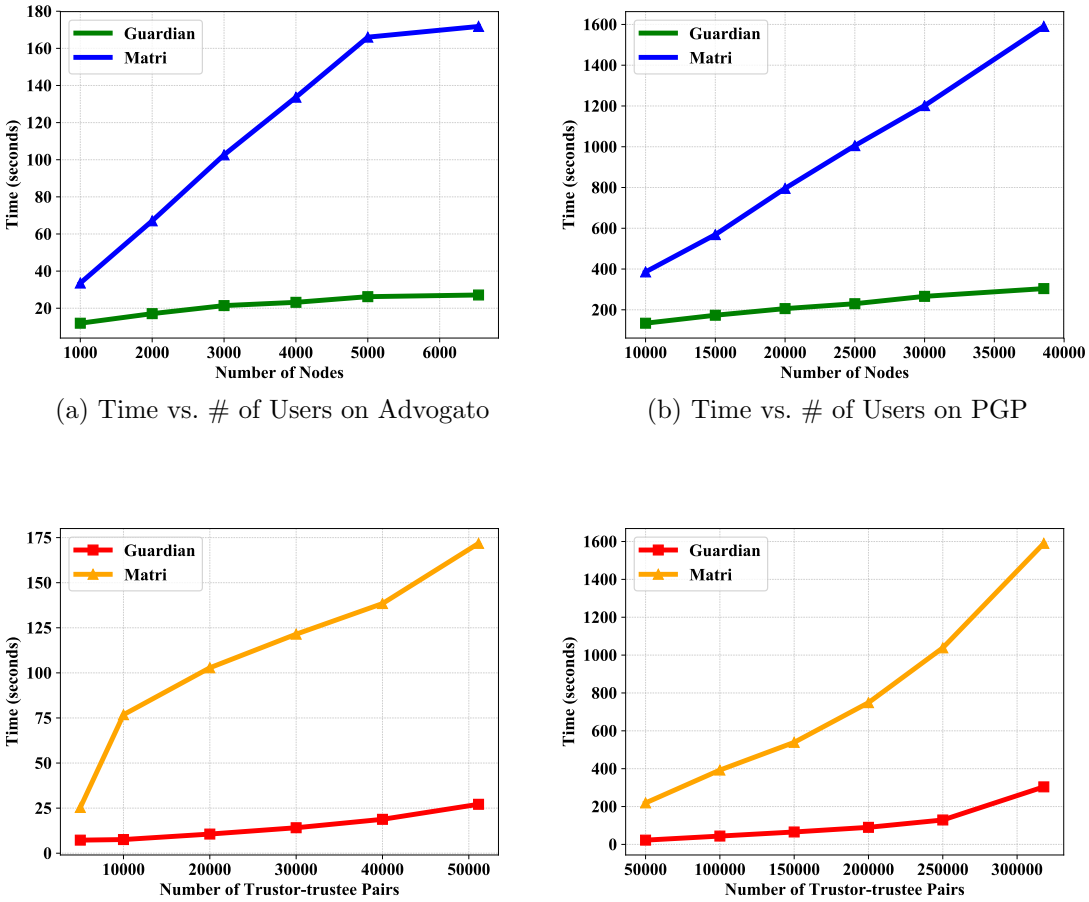


Figure 3.4: Wall-clock time on Advogato and PGP[6].

and the results are also shown in Fig. 3.4. It is worth noting that *Guardian* consistently outperforms all other baselines on all datasets. In particular, *Guardian* shortens the processing time significantly on NeuralWalk by $2,827\times$. Note that, when we run NeuralWalk on PGP, running one of three iterations has already cost us around 52 hours before it ran out of memory. Comparing to Matri, *Guardian* is $6.17\times$ and $5.23\times$ faster on Advogato and PGP respectively. It demonstrates that our proposed trust convolutional layer greatly speeds up trust evaluation process in online social networks and shows its promising that can be applied to large-scale network applications.

To enhance the understanding of the time cost for training[7]and inference respectively, we measured the time used for these two processes separately on both datasets. Table 3.5 summarizes the training and inference time for different approaches. As reported, *Guardian* is $6.17\times$ faster than Matri on the training phase, which shows the total time cost of Matri mainly comes from its matrix factorization phase. We also noticed that our framework bears longer inference time as compared with Matri. However, Matri can not evaluate the trust relationship for users that were not seen during the learning phase, while our proposed *Guardian* is an inductive model that can be generalized to unseen users. Therefore, a retraining of the dataset is needed for newly added users for

Matri. *Guardian*, on the other hand, does not require retraining because the pre-trained parameters can be saved for later inference.



(a) Time vs. # of Users on Advogato

(b) Time vs. # of Users on PGP

(c) Time vs. # of Pairs on Advogato

(d) Time vs. # of Pairs on PGP

Figure 3.5: Scalability: *Guardian* vs. Matri.

Scalability. The scalability of *Guardian* is evaluated by measuring the wall-clock times with a different number of users and a different number of trustor-trustee pairs, respectively. Both of the selected users and pairs are subgraphs from the main graph of the dataset, and each node in the subgraphs has at least one edge (no singleton node). We observe that the results, shown in Fig. 3.5, are consistent with the complexity discussions in Sec. 3.2.5.

More specifically, as Fig. 3.5a and Fig. 3.5b show, the wall-clock time of Matri in-

Table 3.6: Robustness with Different Sizes of Training Set on Advogato

Approach	Training Set(%)	F1-Score	MAE
Guardian	80%	$\mathbf{74.3\% \pm 0.4\%}$	$\mathbf{0.082 \pm 0.002}$
	60%	$\mathbf{72.9\% \pm 0.2\%}$	$\mathbf{0.087 \pm 0.001}$
	40%	$\mathbf{70.7\% \pm 0.1\%}$	$\mathbf{0.094 \pm 0.001}$
Matri	80%	$65.6\% \pm 0.4\%$	0.127 ± 0.001
	60%	$63.9\% \pm 0.3\%$	0.132 ± 0.001
	40%	$61.7\% \pm 0.3\%$	0.139 ± 0.001

creases sharply with the number of users while *Guardian* consistently performs well as the number of users increases. The computation cost of our framework main comes from the localized graph convolutions, of which the complexity depends on the model parameter complexity. Because the parameters of our proposed trust convolutional layers are shared across all users, making the parameter complexity of the trust convolution operation independent of the number of users. we observe a slightly increase on runtime when increasing the number of users.

For the increasing number of trustee-trustor pairs, the time of Matri increases dramatically and shows similar trends on both datasets, shown in Fig. 3.5c and Fig. 3.5d. It is noteworthy that *Guardian* consistently performs well on all benchmarking datasets, indicating that *Guardian* is more scalable and can readily be generalized to large-scale network applications. Note that, the wall-clock time of *Guardian* tends to increase slightly with the increase of the input graph size. That can be explained by the fact that the computation cost of Guardian main comes from the localized trust convolution. The size of neighborhood increases when the number of users increases, leading to more local aggregation computations.

Robustness. We evaluated the approaches with different training and test set ratio to

[7]For simplicity, in this dissertation, we described the factorization phase of Matri as training phase.

Table 3.7: Robustness with Different Sizes of Training Set on PGP

Approach	Training Set(%)	F1-Score	MAE
Guardian	80%	$\mathbf{87.1\% \pm 0.1\%}$	$\mathbf{0.083 \pm 0.001}$
	60%	$\mathbf{86.5\% \pm 0.1\%}$	$\mathbf{0.088 \pm 0.001}$
	40%	$\mathbf{85.3\% \pm 0.2\%}$	$\mathbf{0.096 \pm 0.001}$
Matri	80%	$68.3\% \pm 0.7\%$	0.122 ± 0.0003
	60%	$64.7\% \pm 0.1\%$	0.131 ± 0.0004
	40%	$60.5\% \pm 0.1\%$	0.144 ± 0.0001

measure their robustness. The portions of the training set were set as 80%, 60%, 40% of the entire dataset. Table 3.6 and Table 3.7 show the evaluation results of both datasets. As reported, *Guardian* has a minor performance decrease of 3.6% for Advogato and 1.9% for PGP when the size of the training set is reduced to 40% of the entire graph, while Matri has a decrease of 3.9% and 7.8%, respectively. This indicates that our proposed framework has better robustness, with respect to the size of the training set. Notably, *Guardian* also consistently offers the best prediction accuracy, even when the model was trained with 40% training data as compared to Matri with 80% training data. Training on a subgraph containing 40% could achieve favorable performance, i.e., increases in the training set size did not seem to much help. This further suggests that the proposed framework inherits the capability of inductively estimating the pairwise trustworthiness for users that were not seen during the training phase, which allows us to train on a subgraph to obtain the model parameters.

3.4 Related Work

In this section, we present and discuss some related works on pairwise social trust evaluation and recent advancements in applying convolutional neural networks to graph-structured data.

3.4.1 Pairwise Social Trust Evaluation

Walk-based approaches: In the past decade, most of the existing trust evaluation models were based on the trust propagation along the paths from the trustor to the trustee. For example, ModelTrust [56] and TidalTrust [25] evaluated the pairwise trustworthiness by searching the paths throughout the network. The propagated trust from multiple paths, between the trustor and the trustee, then are aggregated to be the estimated value of trust. Aiming for higher accurate trust evaluation, AssessTrust [52] and OpinionWalk [50] modeled the value of trust using statistical distributions in three-valued subjective logic. In particular, in order to establish a trust relationship between two indirectly connected users, OpinionWalk [50] walked throughout the network in a breadth-first search manner and modeled the trust propagation and aggregation via its predefined discounting and combining operators.

Matrix factorization-based approaches: [91] and Matri [85] are matrix factorization-based approaches, which are proposed to analyze the observed trustworthiness to identify the unobserved/missing trust relationships. In this category, the trustor-trustee pairs were analogous to user-item pairs in a recommender system. In general, the matrix factorization methods are used to map the trustors and the trustees to a joint latent factor space, so that the trustworthiness of the trustor-trustee pairs can be modeled as their inner products in that space. In particular, Matri [85] was designed to combine trust tendency and trust propagation under a collective matrix factorization framework, while [91] further considered the similarity of users' trust rating habits. Since these approaches are inherently transductive, expensive re-training process may be required to estimate the trust values for users that were not seen during the training phase.

Neural network-based approach: In contrast to the aforementioned approaches, NeuralWalk [51] was designed to capture the trust propagation and aggregation rules using machine learning techniques. The main component of this model is WalkNet, a neural network architecture, that was designed to model single-hop trust propagation and ag-

gregation. By iteratively employing WalkNet, NeuralWalk is capable of establishing a trust relationship between the trustor and the trustee, as long as there exists at least one social path from the trustor to the trustee. Even though NeuralWalk can achieve state-of-the-art prediction accuracy in the literature, it is highly inefficient due to the massive matrix operations for training and test set selection.

3.4.2 Graph Convolutional Neural Networks

More recently, graph convolutional neural networks (GCNs) have been proven to be capable of learning on graph structure data [31, 41, 89], leading to new state-of-the-art results on benchmarks such as node classification and link prediction. These GCN-based approaches consistently outperformed techniques based upon matrix factorization or random walks (e.g, node2vec [28], Line [74], and DeepWalk [66]).Their success has led to a surge of interest in applying GCN-based frameworks to applications ranging from recommendation systems [86], drug design [93], to social influence prediction [67].

Despite the compelling success achieved by previous work, little attention has been paid to social trust evaluation with graph convolutional neural networks. Here we fill this gap and show the effectiveness and efficiency of graph convolutional neural networks-based representation learning for social trust evaluation.

3.5 Summary

In this chapter, we devised a new framework *Guardian*, to model social trust for trust evaluation. In this framework, we explicitly incorporated the popularity trust and engagement trust into the latent representations of users to learn effective trust relationships. The key of *Guardian* is the newly proposed trust convolutional layer, which is able to jointly capture social graph structure and associated trust interactions. Extensive experiments on two real-world datasets have demonstrated the rationality and effectiveness of our proposed *Guardian*. In the meanwhile, it enjoys high efficiency due to the notion of

localized graph convolutions. In the future, we are interested in improving *Guardian* by incorporating the attention mechanism during trust propagation. Moreover, we will investigate the capability of *Guardian* to address trust dynamics. It will also be interesting to incorporate the context-aware information to further enhance prediction performance.

Chapter 4

Signed Network Embedding Based on Status Theory

Apart from modeling social relationships with a value of social trust, many real-world relationships on social media often reflect a mixture of positive and negative interactions. For example, people can be friends (positive interactions) or foes (negative interactions). In this context, online social networks can be represented as signed graphs, containing positive and negative links. Complementary to chapter 3, we shift our focus to the problem of signed network analysis in this chapter. Except for the difference of relationship modeling, this chapter studies the use of graph convolutional network to generate signed network embeddings, which encodes the network topology — including sign and direction information of links — on signed networks. Such representations are useful to carry out various tasks in signed network analysis, such as link sign prediction and node ranking.

In this chapter, we propose *SiGCN*, a new framework that learns representations for users on signed networks with graph convolutional neural networks. *SiGCN* is designed based on *status theory*, a social-psychological theory specifically developed for directed networks. Different with *Guardian* proposed in chapter 3, *SiGCN* can learn effective representations and obtain the status score of each user. According to the principles of status theory, the link sign can be derived from the relative difference of users' status scores. Except for the link sign prediction task, the learned status score can be used for

node ranking tasks.

Highlights of our original contributions are as follows. *First*, we introduce a principled methodology to jointly capture both the graph structure and associated sign information of links within signed directed networks. *Second*, we utilize status theory to capture the status of individual users such that both sign and direction information of links can be captured in the embedding space. *Third*, we demonstrate the effectiveness and efficiency of our proposed framework using four signed directed networks from different domains. Our extensive array of experiments on benchmarking datasets demonstrated that *SiGCN* can speedup representation learning for link sign prediction by up to 6.5× as compared with the baselines. More specifically, *SiGCN* speeds up to 4× faster and achieves comparable accuracy as compared to BESIDE [16], increases accuracy by up to 18.8% compared with SIDE [39], and achieves the state-of-the-art robustness and scalability as reported in the literature. We also show that *SiGCN* can learn effective status score of each user, which can be used for link sign prediction and node ranking and yield state-of-the-art performance.

The remainder of this chapter is organized as follows. We first introduce some background knowledge, particularly about the social-psychological theories used in mining signed networks in Sec. 4.1. Then we formulate the problem of network representation learning on signed directed networks in Sec. 4.2. In Sec. 4.3, we illustrate the details of our framework designed to learn signed network embedding effectively and efficiently. In Sec. 4.4, we present an extensive array of experimental results to evaluate the performance of our framework. Sec. 4.5 discusses related work and Sec. 4.6 concludes this chapter.

4.1 Preliminaries

Before we formulate the problem of network representation learning on signed directed networks, we introduce some necessary definitions to facilitate a better understanding of

the problem and our proposed solution.

Definition 4.1.1. Social status: Status is broadly defined as the position of users, either communities or individuals, in a social hierarchy that results from accumulated acts of deference. It has been widely recognized by the sociologists that status is fundamentally rooted in the accumulation of deference behaviors [26, 69]. In signed social networks, social status can be represented in many different ways, such as the rankings of nodes in social networks, and it represents the prestige/trustworthiness of nodes [73].

Definition 4.1.2. Status theory: Status theory defines an organizing principle for signed links on signed directed networks. In the theory of status [44], a positive link $e_{u\to v}$ implies that v has a higher status from the perspective of u (shown in Fig. 4.1c), while a negative link $e_{u\to v}$ indicates that v is regarded as having a lower status, as shown in Fig. 4.1d. In signed networks, these relative levels of status can be propagated and aggregated throughout the networks.

Definition 4.1.3. Structural balance theory: Balance theory [12] classifies cycles in a signed network as being balanced or unbalanced. It implies that cycles with an even number of negative signs are more plausible, hence should be more prevalent in real networks. For simplicity, we illustrate balanced structures with triangles. More specifically, balanced triangles with three positives, shown in Fig. 4.1a, capture the notion that "the friend of my friend is my friend," while those with two negatives, shown in Fig. 4.1b, implies that "the enemy of my enemy is my friend." Balance theory was initially developed for undirected networks.

Comparisons of two theories. Status theory and balance theory both provide insights into ways in which users use linking mechanisms in social computing applications. Status theory is specialized in directional links, as it posits a status differential from the source node of a link to its target node. Balance theory was initially proposed for undirected networks, though it has been widely applied to directed networks (e.g., SIDE [39],

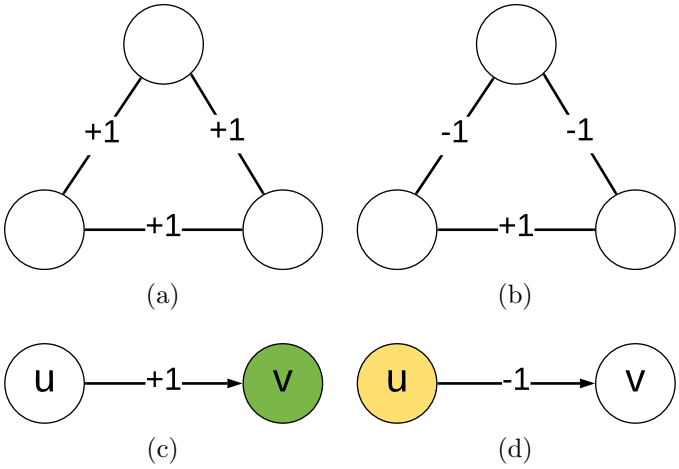


Figure 4.1: (a) (b) Balanced triangle examples: balanced triangle with zero negative and balanced triangle with two negatives, respectively; (c) (d) Status theory illustration: a positive link $e_{u\to v}$ implies that v has a higher status from the perspective of u. A negative link $e_{u\to v}$ indicates that v is regarded as having a lower status.

SigNet [35]). Structural balance theory can be viewed as modeling like and dislike relationships [44], while in some important domains, such as Epinions and Wikipedia, a positive link from u to v can be interpreted as "v has higher status than I do" and a negative link can be viewed as a model of "v has lower status than I do" [29]. In these domains, status theory has been proved to have more expressiveness than balance theory [44].

4.2 Problem Setup

In this chapter, we consider a signed directed network, which is modeled as a signed directed graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}^+$, $\mathcal{E}^-$ represent the sets of positive links and negative links, respectively. Any node $v \in \mathcal{V}$ represents an user in the social network and each link $e_{u\to v} \in \{\mathcal{E}^+ \cup \mathcal{E}^-\}$ is a directed link from u to v associated with a positive or negative sign. More precisely, $+1$ represents a positive link and -1 denotes a negative link. As nodes typically represent users in social media, we use the terms "node" and "user," "links/edges" and "relationships" interchangeably in this dissertation. To differentiate direction information between any

two users, we define the source node as the creator of a link, while the target node as the receiver of a link. This reflects real-world application domains, such as rating reviewers on Epinions and voting for Adminship on Wikipedia (WikiRfA) [82], where both ratings and votes are created by the source nodes of the links.

For any node $u \in \mathcal{V}$, let $N_O(u)$ be the set of out-neighbors of node u, and $N_I(u)$ be the set of in-neighbors of u. In this sense, we can define $|N_I(u)|$ and $|N_O(u)|$ to represent in-degree and out-degree of u, respectively. The mathematical notations used in this chapter are summarized in Table 4.1.

Signed Network Embedding/Network Representation Learning: Given a signed directed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-)$, the task of signed network representation learning is to learn a mapping function $f : u \rightarrow S[u]$, where $S[u] \in \mathcal{R}^d$ is the learned representation of user u with dimension d. The transformation function f preserves the original network information, including network structure, and both sign and direction information of the links, such that any representations of users, in the embedding space, are effective for downstream signed network analysis, e.g., link sign prediction and node ranking. An example of a signed directed network is shown in Fig. 1.2, which will be used throughout this chapter.

4.3 *SiGCN: Proposed Framework*

In this section, we first introduce the limitations of balance theory on signed directed networks. Then, we introduce our proposed signed graph convolutional network with status theory, *SiGCN*, a dedicated effort towards the construction of signed graph convolutional networks specialized for signed directed networks.

In particular, we examine the percentage of triangles satisfying balance theory. There are 92.4% triangles satisfying balance theory on Epinions, while on WikiRfA, only 73.62% triangles are balanced based on the balance theory. This indicates that signed graph convolution networks based on the balance theory may not work very well on signed

Table 4.1: Notations

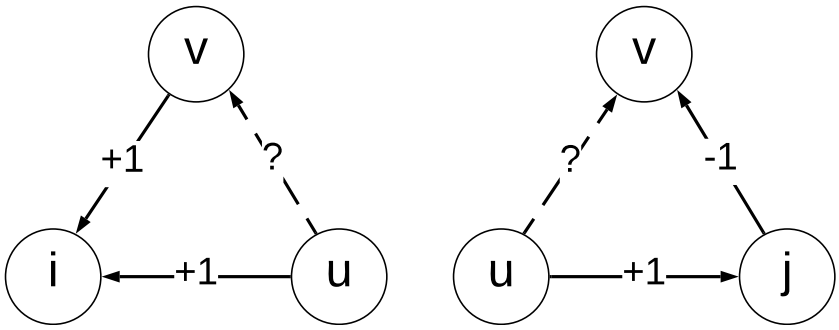
Notation	Descriptions
$N_O(u)$	the set of out-neighbors of u
$N_I(u)$	the set of in-neighbors of u
G, R	the generative status and receptive status
$S_g[u]$	the latent factor of generative status propagated from the out-neighbors of user u
$S_r[u]$	the latent factor of receptive status propagated from in-neighbors of user u
$S[u]$	the latent factor of status for user u
$\text{sign}_{u\to v}$	observed sign of link $e_{u\to v}$
$\tilde{\text{sign}}_{u\to v}$	predicted sign of link $e_{u\to v}$
$\otimes$	the concatenation operator of two vectors
$\oplus$	mean aggregator
σ	non-linear activation functions, e.g., sigmoid$(\cdot)$
W, b	the model parameters of *SiGCN*

directed networks. Let's see an example in our signed directed network graph, originally shown in Fig. 1.2. User v links positively to user i and u links positively to user i; balance theory would suggest a positive link of $e_{u\to v}$, as shown in Fig. 4.2a. However, as shown in Fig. 4.2b, user u links positively to user j and j links negatively to user v; balance theory would suggest a negative link of $e_{u\to v}$. The predicted sign of link $e_{u\to v}$ based on the balance theory with triangles are contradictory.

Before formalizing our signed graph convolutional network with status theory, we provide some intuitions behind our construction. In signed networks, both positive and negative links need to be jointly considered for network representation learning. As such, the notion of homophily becomes not applicable, which can also be explained by the theory of status. According to the theory of status, the positive in-neighbors and the negative out-neighbors of a node increase its status. In contrast, the positive out-neighbors and negative in-neighbors decrease its status. In this sense, status aggregation

and propagation in signed networks desire new principles though the status of each user is determined by its local topological connections.

To illustrate the modeling with status theory in our example, we simplify the status modeling as follows: each user was initially assigned a status of 0. If user u links positively to another user, or another user links negatively to user u, we increase user u a status of 1; otherwise, we decrease the status of u by 1. With the same rule, the status value of each user is assigned as shown in Fig. 4.3. As u has a status of -4 and v has a status of -2, status theory would suggest a positive link of $e_{u\to v}$. It is worth mentioning that status is propagated and aggregated throughout the networks, which is more complex in the real situation. In what follows, we detail our proposed framework for representation learning on signed directed networks, the architecture of which is illustrated in Fig. 4.5.



(a) Predicted as positive (b) Predicted as negative

Figure 4.2: Contradictory predictions of $e_{u\to v}$ with balance theory in our example.

4.3.1 Modeling Social Status of Users

In this chapter, we consider the richness of interactions between users within signed directed networks as *social status*. Social status can be represented in many different ways depending on the interpretations of positive and negative links, which are typically distinct across different application domains. In the context of signed social media, status theory allows deriving the relationship of any two users based on their status in the social graph.

In particular, to be able to capture the social status of each user, we first separate

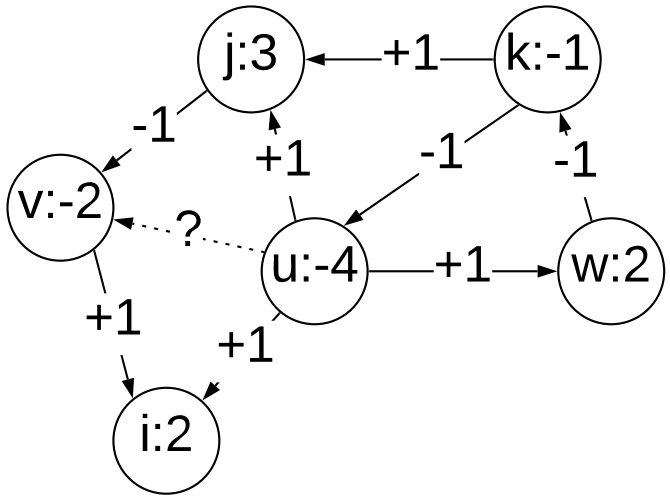


Figure 4.3: Predicted as positive: modeling with status theory in our example.

pairwise interactions of users into two groups: *receptive interactions* and *generative interactions*. Taking the Epinions, a who-trust-whom dataset, as an example, receptive interactions can be interpreted as the measurement of the popularity of a user. In this context, some users are more likely trusted by others, such as the officials, who are referred to as having higher popularity in a society. Therefore, the social status of these users is relatively higher. According to the theory of status, receptive interactions with positives increase the social status of the receiver of the links, whereas those with negatives decrease the social status of the receiver.

Similarly, generative interactions are to measure the engagement of a user. In the context of Epinions, users are more likely to trust others with higher social status, hence their social status is relatively lower than those users they trust. Generative interactions with negatives increase the social status of the creator of the link, while those with positives lower the social status of the creator.

4.3.2 Status Convolutional Layers

Accordingly, we consider two types of status aggregation to characterize the receptive-based status and generative-based status, represented as $S_r[u]$ and $S_g[u]$, respectively.

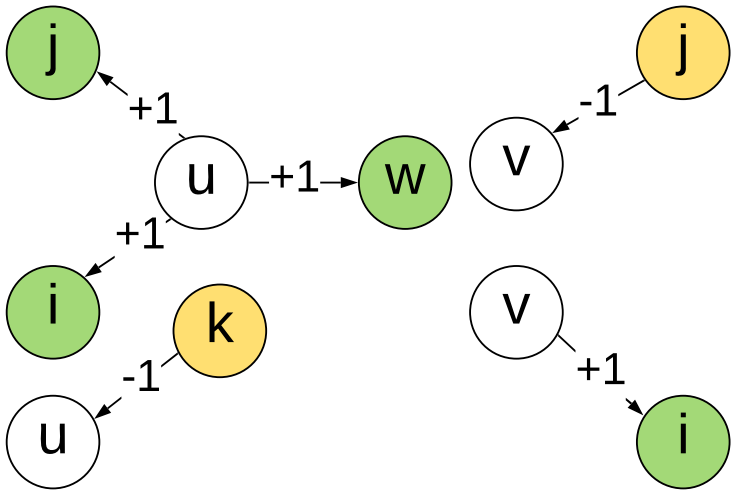


Figure 4.4: The receptive interactions (in yellow) and generative interactions (in green): an example.

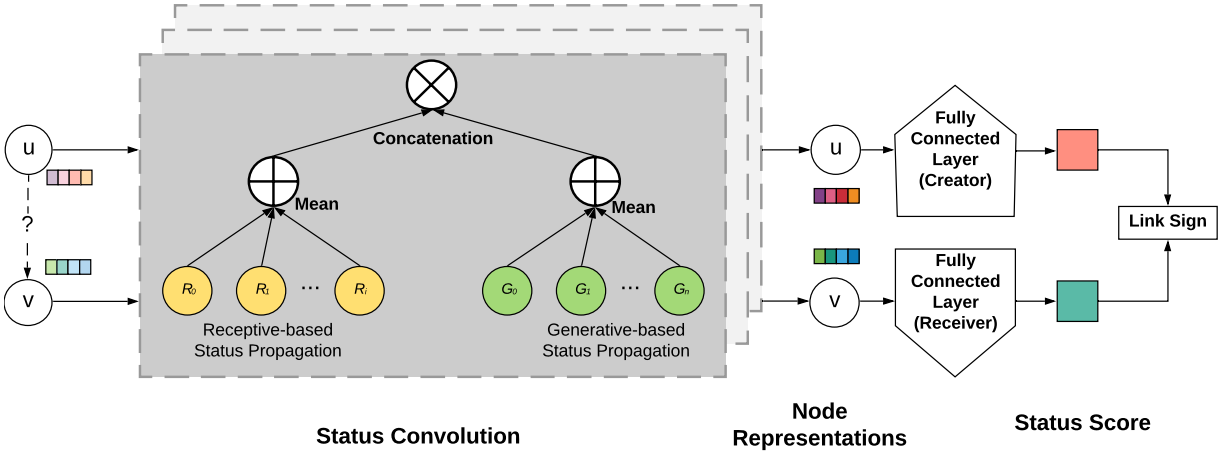


Figure 4.5: Illustration of *SiGCN* framework.

For each of them, we use mean-aggregator to aggregate its associated interactions with its neighbors. It is worth mentioning that, mean-aggregator is the main operation of aggregating information from local graph neighborhoods [31, 41].

Let's see an example in our example social network graph, shown in Fig. 1.2. With our link modeling, the receptive interactions of user u and v are depicted in yellow in Fig. 4.4, while the generative interactions are shown in green. More specifically, for user u, there are three outgoing neighbors. The generative-based status of u is, therefore, 1

by averaging over its outgoing links, and the receptive-based status of u is -1. Similarly, the receptive-based status of v is -1 while its engagement is 1.

***R**eceptive-Based Status Propagation* (R). Intuitively, the incoming social connections and associated receptive link with sign information provide direct evidence on the popularity of a user in online social networks. We build upon this basis to propagate the receptive-based status between connected users.

In particular, to model sign information of the links, we first use one-hot encoding to represent positive and negative links, respectively. More specifically, we model positive and negative links as the following one-hot representations: $[1, 0]^T$, and $[0, 1]^T$. Then *SiGCN* employs a linear transformation to convert the one-hot encodings into dense vector embeddings through Eq. (4.1) and Eq. (4.4). For a link with sign $\text{sign}_{u\leftarrow v}$ (v is the creator of the link), we model the receptive-based status of u created by v as a combination of v's feature vector $x[v]$ and the embedding of sign information $e_{\text{sign}_{u\leftarrow v}}$.

$$e_{\text{sign}_{u\leftarrow v}} = W_{u\leftarrow v} \cdot \text{sign}_{u\leftarrow v} \tag{4.1}$$

$$\text{R}_{u\leftarrow v} = x[v] \otimes e_{\text{sign}_{u\leftarrow v}} \tag{4.2}$$

where $W_{u\leftarrow v} \in \mathcal{R}^{D_e \times 2}$ is a trainable transformation matrix, $\otimes$ denotes the concatenation operation between two vectors.

We now take the element-wise mean of the vectors in $\{\text{R}_{u\leftarrow v}, \forall v \in N_I(u)\}$. This mean-based aggregator is a linear approximation of a localized spectral convolution [41], as the following function:

$$S_r[u] = \frac{1}{N_I(u)} \cdot \sum_{v \in N_I(u)} \text{R}_{u\leftarrow v} \tag{4.3}$$

***G**enerative-Based status Propagation* (G). Accordingly, we characterize the engagement of a user through its outgoing social connections and associated sign information of

generative links. We build upon this basis to perform the propagation and aggregation of the generative-based status between the connected users. Thus, the generative-based status of user u can be captured by the following functions:

$$e_{\text{sign}_{u\to v}} = W_{u\to v} \cdot \text{sign}_{u\to v} \tag{4.4}$$

$$\text{G}_{u\to v} = x[v] \otimes e_{\text{sign}_{u\to v}} \tag{4.5}$$

$$S_g[u] = \frac{1}{N_O(u)} \cdot \sum_{v\in N_O(u)} \text{G}_{u\to v} \tag{4.6}$$

where $\text{G}_{u\to v}$ denotes the engagement of user u to user v in signed social networks.

Learning Status Latent Factors of Users. In order to learn better latent factors of users for downstream signed directed network analysis, the receptive-based status and generative-based status are needed to be considered jointly. Here, we propose to combine these two types of status through a standard fully connected (FC) layer, where $S_r[u]$ and $S_g[u]$ are concatenated before feeding into the FC. Formally, the status latent factor of user u, $S[u]$, can be characterized as follows:

$$S[u] = W \cdot (S_r[u] \otimes S_g[u]) + b \tag{4.7}$$

where W is a trainable transformation matrix, b is a learnable bias, and $\otimes$ represents the concatenation operator. The advantage of using concatenation lies in its simplicity and expressiveness, which have been shown in a recent work of graph convolutional neural networks [31].

Higher-order Status Propagation. By stacking l status convolutional layers, a user is capable of receiving the status (the generative-based status and receptive-based status) propagated from its l-hop neighbors. In the l-th step, the representation of user u is recursively formulated as Eq. (4.8) - Eq. (4.12):

$$\mathrm{R}^{l}_{u\leftarrow v} = S^{l-1}[v] \otimes \{W^{l}_{u\leftarrow v} \cdot \mathrm{sign}_{u\leftarrow v}\} \tag{4.8}$$

$$\mathrm{G}^{l}_{u\rightarrow v} = S^{l-1}[v] \otimes \{W^{l}_{u\rightarrow v} \cdot \mathrm{sign}_{u\rightarrow v}\} \tag{4.9}$$

$$S^{l}_{r}[u] = \frac{1}{N_I(u)} \cdot \sum_{v\in N_I(u)} \mathrm{R}^{l}_{u\leftarrow v} \tag{4.10}$$

$$S^{l}_{g}[u] = \frac{1}{N_O(u)} \cdot \sum_{v\in N_O(u)} \mathrm{G}^{l}_{u\rightarrow v} \tag{4.11}$$

$$S^{l}[u] = W^{l} \cdot (S^{l}_{r}[u] \otimes S^{l}_{g}[u]) + b^{l} \tag{4.12}$$

where $S^0[u] = x[u]$ is the feature vector of node u, $\mathrm{sign}_{u\rightarrow v}$ and $\mathrm{sign}_{u\leftarrow v}$ are the observed signed link, and $W^{l}_{u\leftarrow v}$, $W^{l}_{u\rightarrow v}$, W^{l}, and b^{l} are the model trainable parameters, to be optimized in an end-to-end fashion with *SiGCN*. Note that, by stacking multiple status convolutional layers, we not only enrich user embedding with its receptive-based status and generative-based status in signed social networks, but also allow controlling the range of status propagation throughout the graph by adjusting l.

4.3.3 Modeling Relationships of Users Based on Status Theory

In the theory of status, a signed link, from creator u to receiver v, can be interpreted as the intention of u in creating the link to v [29]. Status theory characterizes the sign of the links from the relative difference of their status score.

In order to further model status score for each user, we fit the latent factor of the user status to fully-connected (FC) layers. In particular, we use two different fully-connected layers to learn the status scores for the creator and receiver, respectively. Formally, the status scores are formulated as Eq. (4.13) and Eq. (4.14), where W_{src} and W_{rev} are trainable weight matrices defined in two FC layers, and b_{src} and b_{rev} are corresponding

biases. The FC layers lead to more effective representations of user status, as this step explicitly injects the receptive-based status and generative-based status of individual users in a collaborative fashion.

$$S_{\text{src}}[u] \leftarrow W_{\text{src}} \times S^L[u] + b_{\text{src}} \tag{4.13}$$

$$S_{\text{rev}}[v] \leftarrow W_{\text{rev}} \times S^L[v] + b_{\text{rev}} \tag{4.14}$$

With status theory, the sign of a link can be captured by its relative differential of the creator and receiver. Formally, it is formulated as:

$$S_{\Delta_{u\to v}} = S_{\text{src}}[u] - S_{\text{rev}}[v] \tag{4.15}$$

Therefore, the sign of link $e_{u\to v}$ can be computed as:

$$\tilde{\text{sign}}_{e_{u\to v}} = \begin{cases} -1 & \text{if } \sigma(S_{\Delta_{u\to v}}) > 0.5 \\ +1 & \text{otherwise} \end{cases}$$

where σ is a sigmoid function to normalize the status differential to the range of $(0, 1)$, defined as $\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$. The detailed forward propagation algorithm of *SiGCN* is shown as **procedure SiGCN**.

4.3.4 Model Training

We define an objective function to learn the model parameters in *SiGCN*. In particular, the objective function contains two terms, both of which are to encourage the representations to be able to understand associated status so as to obtain sign information of the links based on status theory. More specifically, the first term is to minimize the status differential of positive links, while the second term is to maximize the status differential of negative links. The overall objective function is formalized as follows:

$$\begin{aligned}\mathcal{L} = (\text{sign}_{e_{u\to v}}) \cdot \frac{1}{|\mathcal{E}^+|} \sum_{e_{u\to v}\in\mathcal{E}^+} \sigma(S_{\Delta_{u\to v}}) \\ +(\text{sign}_{e_{u\to v}}) \cdot \frac{1}{|\mathcal{E}^-|} \sum_{e_{u\to v}\in\mathcal{E}^-} \sigma(S_{\Delta_{u\to v}}) \\ +\lambda \cdot ||\Theta||_2^2\end{aligned} \tag{4.16}$$

where $\text{sign}_{e_{u\to v}}$ represents the ground-truth (Eq. 4.17), and $\Theta = \{\{W^l_{u\leftarrow v}, W^l_{u\to v}, W^l\}_{l=1}^L, W_{\text{src}}, W_{\text{rev}}\}$ denotes all trainable model parameters, and λ controls the L_2 regularization strength to prevent over-fitting. In particular, we adopt Adam [40] as the optimizer in our implementation, as it has been shown to be effective in updating the model parameters [31].

$$\text{sign}_{e_{u\to v}} = \begin{cases} -1 & \text{if } e_{u\to v} \in \mathcal{E}^- \\ +1 & \text{if } e_{u\to v} \in \mathcal{E}^+ \end{cases} \tag{4.17}$$

---

procedure SiGCN: Representation Generation (i.e. Forward Propagation)
  $S^0[u] \leftarrow x[u]$, for all $u \in \mathcal{V}$, where X is node feature matrix
  ▷ Status latent factors of observed users
  for all $u \in \mathcal{V}$ **do**
    for $l = 1 \cdots L$ **do**
      ▷ Receptive Status
      $S_r^l[u] = \frac{1}{N_I(u)} \cdot \sum_{i\in N_I(u)} \text{R}^l_{u\leftarrow i}$
      ▷ Generative Status
      $S_g^l[u] = \frac{1}{N_O(u)} \cdot \sum_{i\in N_O(u)} \text{G}^l_{u\to i}$
      $S^l[u] = W^l \cdot [S_r^l[u] \otimes S_g^l[u]] + b^l$
  ▷ Status Score
  for all $\langle u, v\rangle \in \mathcal{W}$ **do**
    $S_{\text{src}}[u] \leftarrow W_{\text{src}} \times S^L[u] + b_{\text{src}}$
    $S_{\text{rev}}[v] \leftarrow W_{\text{rev}} \times S^L[v] + b_{\text{rev}}$
    $S_{\Delta_{u\to v}} = \sigma(S_{\text{src}}[u] - S_{\text{rev}}[v])$

Analysis. The key computational operations of our framework are the notion of localized graph convolutions [31,86], which can be made inductive. At the representation

Table 4.2: Statistical Description of the Datasets.

Dataset	# of Nodes	# of Edges	+ Edges (%)	- Edges (%)
Epinions	131,828	841,372	85.30	14.70
Slashdot	82,140	549,202	77.40	22.60
WikiRfA	11,258	179,418	77.92	22.08
WikiElec	7,126	104,167	78.78	21.22

generation step, we are able to compute embeddings for nodes that were not in the training set. This allows us to train on a subgraph to obtain model parameters, and then generate embeddings for nodes that have not been observed during training. Specifically, each status convolutional layer learns how to aggregate social status of users from a small graph neighborhood in the social graph. Therefore, *SiGCN* not only can compute nodes embeddings but also can evaluate status scores that can be used for node ranking as well as link sign prediction. It is worth mentioning that parameters of our proposed status convolutional layers are shared across all users, making the parameter complexity of *SiGCN* independent of the input graph size. Sec. 4.4 empirically verified the efficiency and scalability of our model. In addition, as *SiGCN* is an inductive learning model, it is able to estimate the sign of links between any two users that were not seen during the training phase.

4.4 Experimental Evaluation

4.4.1 Description of Datasets Used

To evaluate the effectiveness and efficiency of *SiGCN*, we conduct experiments on four benchmark datasets: Epinions, Slashdot [44], WikeRfA [82], and WikiElec [43], which are publicly accessible signed social network datasets. The statistics of these datasets are presented in Table 5.2.

Epinions: This is a who-trust-whom online social network, where users create signed directed relations to each other indicating trust (corresponding to positive links) or distrust relationships (represented as negative links).

Slashdot: Slashdot is a technology-related news website known for its specific user community. Users in the network designate others as "Friends" (positive links) or "foes" (negative links).

WikiRfA: This network is defined by votes for Wikipedia administrator candidates. Any member can cast a supporting, neutral or opposing vote for a Wikipedia editor. We discard neutral votes and construct a signed directed network as did in BESIDE [16] and [82]. **WikiElec:** is the elections and voting data of Wikipedia administrator. The definition of this dataset is similar to that of WikiRfA. The details on how the signed edges are defined can also refer to the website[1].

4.4.2 Link Sign Prediction Based on Learned Node Representations

Baselines. To demonstrate the effectiveness of learned embedding on link sign prediction, we compared *SiGCN*, our proposed framework against the state-of-the-art methods on signed network embedding and below we will detail them. We do not include unsigned methods (e.g., LINE [74], Node2Vec [28]) and spectral clustering algorithms based on signed Laplacian matrix (e.g., SSE [42]), since previous signed network embedding work (BESIDE [35] and SGCN [19]) has shown their superiority over these methods.

SNE [87] adopted a log-bilinear model and used random walk sampling to generate samples. SNE was designed without any specific theories of signed networks.

SiNE [79] proposed a multi-layer neural network to learn the embeddings by optimizing an objective function satisfying structural balance theory. SiNE only concentrated on the immediate neighborhoods rather than on the global balance structure.

SIDE [39] provided a linearly scalable approach with regard to the number of nodes.

[1] `https://snap.stanford.edu/data/index.html`

Table 4.3: Sign Prediction Results #1

Dataset	Metric	SNE	SiNE	SIDE	SigNet
Epinions	auc	0.8282	0.8589	0.8768	0.9071
	F1-binary	0.9206	0.9082	0.9487	0.9484
	F1-macro	0.7634	0.6968	0.7817	0.8031
	F1-micro	0.8529	0.8332	0.9094	0.9104
Slashdot	auc	0.6447	0.7736	0.7180	0.8759
	F1-binary	0.8726	0.8671	0.8675	0.8994
	F1-macro	0.4663	0.6341	0.5360	0.7579
	F1-micro	0.7740	0.7654	0.7728	0.8406
WikiRfa	auc	0.6954	0.8610	0.6824	0.7903
	F1-binary	0.8823	0.8730	0.8748	0.9012
	F1-macro	0.6503	0.7365	0.4713	0.7454
	F1-micro	0.8095	0.7746	0.7793	0.8407
WikiElec	auc	0.8169	0.7214	0.6682	0.7274
	F1-binary	0.8960	0.8787	0.8836	0.8777
	F1-macro	0.6803	0.6393	0.4833	0.5985
	F1-micro	0.8259	0.7836	0.7924	0.7596

SIDE aggregated the direction and sign information of the links along the paths based on structural balance theory. It was proposed to optimize the likelihood over both directed and undirected signed connections.

SIGNet [35] was built upon the traditional word2vec family of embedding approaches. It leveraged a targeted node sampling strategy to maintain structural balance in higher-order neighborhoods.

SGCN [19] was proposed to fill the gap between the recent advances in unsigned GCNs and the domain of singed network embedding. It was a graph convolutional network specialized for signed network analysis. Balance theory was leveraged to aggregate and propagate the information of signed networks across signed GCN layers.

Table 4.4: Sign Prediction Results #2

Dataset	Metric	SGCN-UD	SGCN-D	BESIDE-tri	BESIDE	SiGCN
Epinions	auc	0.8997	0.8926	0.9304	**0.9437**	0.9397
	F1-binary	0.9486	0.9528	0.9601	0.9637	**0.9640**
	F1-macro	0.8212	0.8079	0.8478	0.8661	**0.8722**
	F1-micro	0.9119	0.9171	0.9306	0.9368	**0.9381**
Slashdot	auc	0.8444	0.8409	0.8769	**0.9108**	0.8975
	F1-binary	0.8918	0.8942	0.9039	**0.9148**	0.9118
	F1-macro	0.7023	0.6904	0.7592	**0.7990**	0.7957
	F1-micro	0.8230	0.8245	0.8460	**0.8657**	0.8616
WikiRfA	auc	0.8512	0.8393	0.8931	0.8969	**0.8979**
	F1-binary	0.9024	0.9001	0.9081	0.9101	**0.9136**
	F1-macro	0.7411	0.7285	0.7679	0.7740	**0.7848**
	F1-micro	0.8408	0.8369	0.8526	0.8560	**0.8619**
WikiElec	auc	0.8523	0.8534	0.8981	0.9003	**0.9008**
	F1-binary	0.9058	0.9073	0.9142	0.9145	**0.9190**
	F1-macro	0.7446	0.7433	0.7723	0.7735	**0.7899**
	F1-micro	0.8463	0.8480	0.8608	0.8612	**0.8692**

BESIDE [16] was the state-of-the-art representation learning on signed directed networks in the literature. Its core is to incorporate both balance theory and status theory for signed network embedding. By incorporating both triangles and "bridge" edges, BESIDE is able to learn effective embeddings for nodes and edges on signed directed networks. In particular, **BESIDE-tri** is a component of BESIDE, which only uses triangles with balance theory to learn user embeddings.

Among the baselines, SiNE can only deal with undirected signed social networks and SGCN only evaluated their model on undirected networks, while the others were designed in the context of directed signed networks. To be comparable, we follow the

sampling method of [39] to generate associated undirected networks of the benchmark datasets. Then we evaluated SiNE over the undirected signed networks. As SGCN works for both undirected and directed networks, we run SGCN over undirected (SGCN-UD) and directed (SGCN-D) networks, respectively.

Evaluation metrics. Four standard metrics are used to measure link-sign prediction accuracy, including AUC, binary-F1, macro-F1, and micro-F1. Note that, larger values of these metrics indicate better prediction accuracy. As commonly did in the literature [16], all experiments were run 5 times to obtain the average values. For efficiency and scalability, we used the average time over 5 runs as well.

All the experiments are performed on a computer with Intel Core i7-9700K 8-core 3.6GHz CPU, GeForce GTX 1660 Ti GPU, 32GB RAM and 500GB SSD.

Parameter Settings. We implemented our proposed framework in Pytorch[2]. We split each dataset into two parts: 80% edges for training and 20% for testing. For each run, we run with different train-test splits. As there is no node attributes in the datasets, we randomly initialize the node embeddings with 64 dimensions. In terms of hyperparameters, we applied a grid search for hyperparameters: the learning rate was tuned amongst $\{0.001, 0.005, 0.01, 0.05\}$, the coefficient of L_2 normalization was searched in $\{10^{-5}, 10^{-4}\}$. The model parameters are initialized using Xavier initializer [23]. In addition, the maximum epoch is set as 600 and early stopping strategy was performed, *i.e.*, premature stopping if training loss does not increase for 10 successive epochs. Without specification, we report the results of three status convolutional layers $[32, 64, 32]$, learning rate of 0.01 and normalization coefficient of 10^{-5}.

To evaluate the performance of learned user representations on link sign prediction, we follow the method in BESIDE [16] to get link features through concatenation, *i.e.*, the feature of $e_{u\to v}$ is $[S_{\text{src}}[u] : S_{\text{rev}}[v]]$. For fair comparisons, all methods train a logistic regression model with learned link features for link sign prediction. We used the

[2]https://pytorch.org

released source code for SNE[3], SiNE[4], SIDE[5], SIGNet[6], SGCN[7], and BESIDE[8], of which parameters were initialized as in the corresponding papers.

Prediction accuracy. Table 4.4 reports the performance comparison results, where the bold scores underlined are the best and the ones underlined are the second best. We have the following observations:

SNE achieves poor performance over four datasets, demonstrating the importance of social-physiological theories on signed network analysis. The performance of SiNE is reported on undirected networks. SGCN-UD consistently outperforms than SiNE across all datasets, which indicates that graph convolutional neural networks can effectively capture the complex relationships between users on signed networks.

Compared to SIDE and SIGNet, the experimental results of SIGNet verify that maintaining structural balance in higher-order neighborhoods can improve the expressiveness of nodes representations. SGCN-D generally achieves better performance than SIGNet in Epinions and WikiElec, while performing slightly worse in Slashdot and WikiRfA. Overall, the prediction performance of SGCN and SIGNet are comparable. It makes sense since SGCN introduces structural balance in higher-order connectivity by stacking multiple graph convolutional layers. Both SGCN and SIGNet, therefore, demonstrate the importance of maintaining structural balance theory in higher-order neighborhoods.

BESIDE achieves the best performance among the baselines in all cases. Such improvements might be attributed to the cooperation of structural balance theory and status theory, which incorporating both triangle and "bridge" edges in a complementary manner. This indicates the benefits of applying status theory in directed signed networks.

SiGCN generally yields the best performance as compared with all of the baselines. Note that, a small improvement in the reported evaluation metrics implies a significant

[3] https://bitbucket.org/bookcold/sne-signed-network-embedding/src/master/
[4] http://www.public.asu.edu/~swang187/codes/SiNE.zip
[5] https://datalab.snu.ac.kr/side/
[6] https://github.com/raihan2108/signet
[7] https://github.com/benedekrozemberczki/SGCN
[8] https://github.com/yqc01/BESIDE

influence on the quality of link sign prediction. In particular, comparing to BESIDE-tri (the one only considers structural balance theory), *SiGCN* consistently performs better on all datasets, indicating that status theory can better capture the complex relationships in directed signed network. Interestingly, *SiGCN* outperforms BESIDE in most cases, even though BESIDE incorporated structural balance theory and status theory. On Slashdot, BESIDE can slightly perform better than *SiGCN*. This can be explained by the fact that balance theory — "the enemy of my enemy is my friend" and "the friend of my friend is my friend"— is more coherent with Slashdot than status theory. In a nutshell, by modeling user relationships with status and stacking multiple status propagation layers, *SiGCN* is capable of learning more expressive representations on signed directed networks.

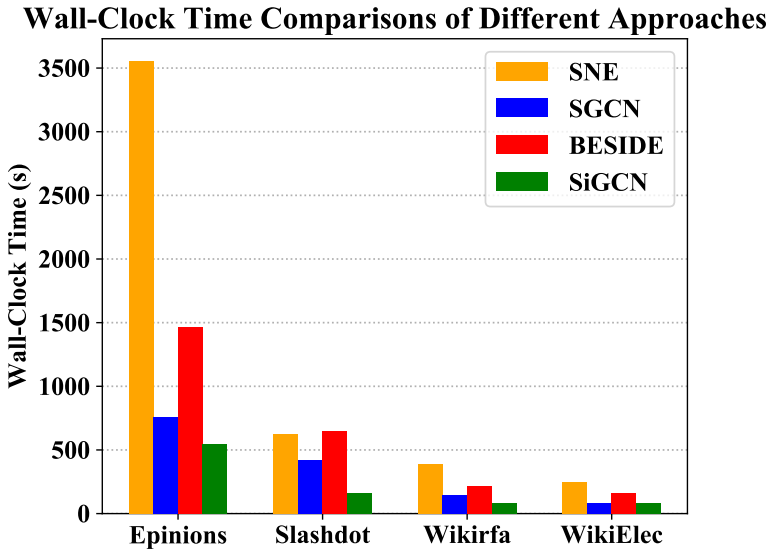


Figure 4.6: Time comparisons.

Efficiency. For efficiency evaluation, we compared the total runtime of our *SiGCN* with the baselines. As the released code of SiNE, SIGNet, and SIDE are not available on GPU, we report the comparison results with the baselines that can run on GPU. The results are shown in Fig. 5.5. It is worth noting that *SiGCN* consistently outperforms in all cases. In particular, *SiGCN* speeds up the processing time by up to 6.5× as compared with

the baselines. In particular, comparing to the state-of-the-art in the literature, BESIDE, *SiGCN* is 4× and 2.7× faster on Slashdot and Epinions, respectively. It demonstrates that our proposed status convolutional layer greatly speeds up the sign prediction process in signed networks and shows its promising that can be applied to large-scale network applications.

Robustness. We evaluated the approaches with different training and test set ratios to measure their robustness. The portions of the training set were set as 80%, 60%, 40% of the entire dataset. Due to the space limitation, we report the evaluation results on Epinions and WikiRfA, shown in Table 4.5. Both of *SiGCN* and BESIDE have good robustness. In particular, *SiGCN* has a minor F1-binary decrease of 0.0026 in Epinions and 0.0054 in WikiRfA when the size of the training set is reduced to 40% of the entire graph, while BESIDE has a decrease of 0.0041 and 0.0106, respectively. This indicates that our proposed framework has better robustness, with respect to the size of the training set.

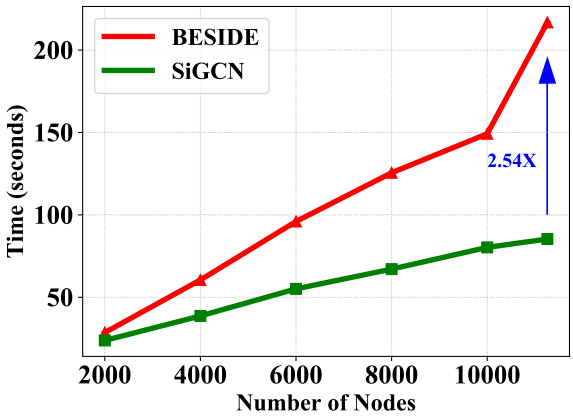
Scalability. The scalability of *SiGCN* is evaluated by measuring the wall-clock times with a different number of users and a different number of user pairs (links/edges), respectively. Both of the selected users and pairs are subgraphs from the main graph of the dataset, and each node in the subgraphs has at least one edge (no singleton node). Due to the space limitation, we only report the results on Epinions and WikiRfA, shown in Fig. 4.7.

More specifically, as Fig. 4.7a and Fig. 4.7c show, the wall-clock time of BESIDE increases sharply with the number of users while *SiGCN* consistently performs well as the number of users increases. This is because the parameters of our proposed status convolutional layers are shared across all users, making the parameter complexity of our approach independent of the number of users. For the increasing number of user pairs, the time of BESIDE increases dramatically and shows similar trends on both datasets, shown in Fig. 4.7b and Fig. 4.7d. It is noteworthy that *SiGCN* consistently performs well on all benchmarking datasets, indicating that *SiGCN* is more scalable and can readily

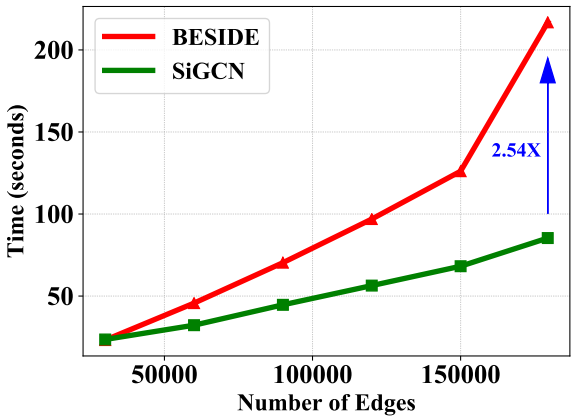
Table 4.5: Robustness

Dataset	Test Size	metric	SIDE	SGCN	BESIDE	SiGCN
Epinions	0.2	auc	0.8768	0.8926	**0.9437**	0.9396
		F1-binary	0.9487	0.9528	0.9637	**0.9642**
		F1-macro	0.7817	0.8079	0.8661	**0.8726**
		F1-micro	0.9094	0.9171	0.9368	**0.9385**
	0.4	auc	0.8788	0.8278	**0.9393**	0.9366
		F1-binary	0.9495	0.9479	0.9615	**0.9616**
		F1-macro	0.7830	0.7598	0.8597	**0.8632**
		F1-micro	0.9108	0.9071	0.9335	**0.9340**
	0.6	auc	0.8700	0.8215	**0.9340**	0.9295
		F1-binary	0.9481	0.9470	0.9596	**0.9616**
		F1-macro	0.7719	0.7553	0.8519	**0.8634**
		F1-micro	0.9080	0.9054	0.9303	**0.9340**
WikiRfA	0.2	auc	0.6824	0.8393	0.8969	**0.8979**
		F1-binary	0.8748	0.9001	0.9101	**0.9136**
		F1-macro	0.4713	0.8295	0.7740	**0.7848**
		F1-micro	0.7793	0.8369	0.8560	**0.8619**
	0.4	auc	0.6801	0.8281	0.8901	**0.8948**
		F1-binary	0.8761	0.8970	0.9068	**0.9122**
		F1-macro	0.4716	0.7220	0.7633	**0.7814**
		F1-micro	0.7813	0.8321	0.8503	**0.8597**
	0.6	auc	0.6710	0.8069	0.8729	**0.8858**
		F1-binary	0.8750	0.8941	0.8995	**0.9082**
		F1-macro	0.4626	0.7090	0.7394	**0.7717**
		F1-micro	0.7791	0.8268	0.8378	**0.8533**

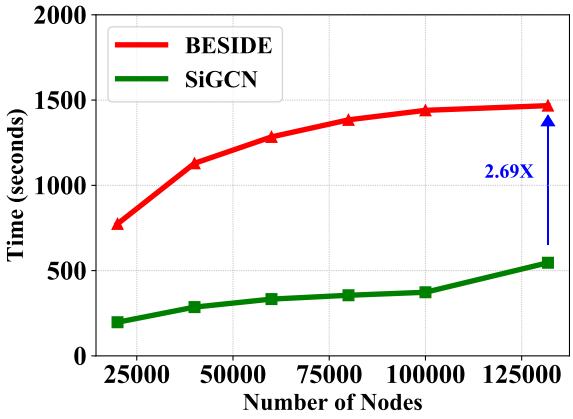
be generalized to large-scale network applications.



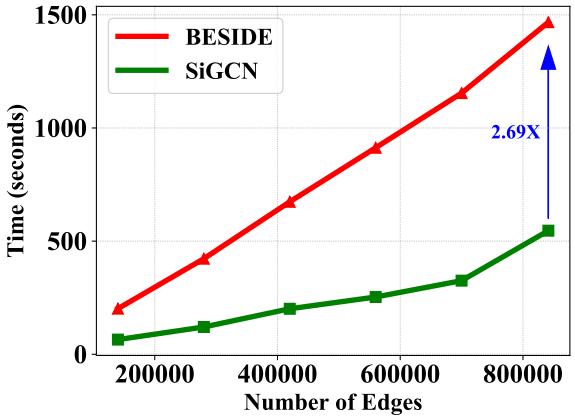
(a) Time vs. #of Nodes on WikiRfA



(b) Time vs. #of Edges on WikiRfA



(c) Time vs. #of Nodes on Epinions



(d) Time vs. #of Edges on Epinions

Figure 4.7: Scalability: SiGCN vs. BESIDE.

Table 4.6: Link Sign Prediction Accuracy Based on Status (%)

DATASET \ METHOD	PageRank	BESIDE-sta	BESIDE	SiGCN
Epinions	65.15	85.28	91.52	**92.40**
Slashdot	62.73	82.77	**86.01**	85.58
WikiRfA	66.37	80.67	82.39	**85.28**
WikiElec	72.83	79.65	81.72	**85.71**

Table 4.7: Global Ranking Results on WikiRfA: never elected (red), elected once (black), and elected twice (blue)

Rank \ Method	WikiRfA		
	PageRank	BESIDE	SiGCN
1	West.andrew.g	Can't sleep...	Nev1
2	Cobi	SarahStierch	Legoktm
3	ProtectionBot	Phaedriel	NCurse
4	Anomie	DerHexer	Dabomb87
5	Jason Quinn	Alex Bakharev	PeterSymonds
6	RedirectCleanupBot	Werdna	DerHexer
7	lustiger seth	HJ Mitchell	Can't sleep...
8	Dinoguy1000	Everyking	Phaedriel
9	Bellhalla	Dabomb87	HJ Mitchell
10	TommyBoy	PeterSymonds	SarahStierch

4.4.3 Link Sign Prediction Based on Status

Baselines. According to the theory of status, a positive edge indicates that the receiver has a higher status than the creator, which can be interpreted as "I trust people who have higher status than me" and vice versa. For example, if $\text{sign}_{e_{u\to v}}$ is $+1$, then $S_{\text{rev}}[v] - S_{\text{src}}[u]$ should be positive. To illustrate the effectiveness of the learned status score of each user, we select three baselines that can obtain the status scores of users. We do not include other methods, (e.g., such as Prestige [94], MPR [70], and Troll-Trust [83]), because BESIDE has shown its superiority over these methods.

PageRank [8] was a classical ranking algorithm for unsigned networks. For this implementation, we follow the same method in BESIDE to obtain the status score of each user, applying it to the positive subgraph (graph contains positive links only) to obtain the global values (status scores) for users.

BESIDE [16] uses both triangles and "bridge" edges to train the model and ob-

Table 4.8: Global Ranking Results on WikiElec: never elected (red), elected once (black), and elected twice (blue)

Rank \ Method	WikiElec		
	PageRank	BESIDE	SiGCN
1	ALKIVAR	SARAH_EWART	BD2412
2	CAMBRIDGEBAYWEATHER	KHOIKHOI	DERHEXER
3	GRAFT	AMIDANIEL	WJBSCRIBE
4	GUANACO2	ELONKA	ARJUN01
5	SCHISSEL	ALEX_BAKHAREV	DUJA
6	ANDYZ	Can't sleep...	NCURSE
7	SAVIDAN	NCURSE	Can't sleep...
8	SEBASTIANKESSEL	HALIBUTT	AMIDANIEL
9	PERUVIANLLAMA	PHAEDRIEL	NEWYORKBRAD
10	SLAMBO	DERHEXER	PHAEDRIEL

tain associated user status. **BESIDE-sta** is a component of BESIDE, which only uses "bridge" edges with status theory to learn the status of each user.

We use 80% edges for training and 20% edges for test. The result is obtained by comparing the status differential of two users and associated ground-truth (link sign) in the test set. Experiments are performed on four datasets, including Epinions, Slashdot, WikiRfA, and WikiElec. We use accuracy as the evaluation metric, as did in BESIDE. The results are shown in Table 4.6, in which we have the following observations:

In general, *SiGCN* outperforms all baselines over Epinions, WikiRfA, and WikiElec, while BESIDE performs slightly better on Slashdot. More precisely, *SiGCN* has significantly better results, a 2.89% improvement on WikiRfA and a 3.99% improvement on WikiElec, respectively, which verifies that these two datasets can be better characterized with status. Moreover, *SiGCN* improves the accuracy by up to 7.12% as compared with BESIDE-sta, indicating that our proposed status convolutional layers can capture the

status property very well. Among all methods, PageRank obtains poor performance, implying the importance of both positive links and negative links in signed network analysis.

4.4.4 Global Node Ranking Based on Status

To determine if the status scores are plausible for ranking on the global scale, we also compare the top 10 nodes ranked based on PageRank, BESIDE, and *SiGCN*, respectively. For PageRank, each node has a single ranking score which can be directly used to find the top 10 nodes. As for BESIDE and *SiGCN*, we use the combination equation as did in BESIDE [16] to obtain the ranking score for each node based on the learned status scores.

$$S_v = \sum_{u \in N_I(v)^+} \frac{S_u}{N_o(u)} - \sum_{u \in N_I(v)^-} \frac{S_u}{N_o(u)} \tag{4.18}$$

S_v is the status score of node v, $N_I(v)^+$ and $N_I(v)^-$ are two sets of source nodes pointing to the node v with positive or negative edges respectively, S_u is the source status score of node u and $N_o(u)$ is the out-degree of node u.

The experiments are conducted on WikiRfA and WikiElec since they have a clear indication of the global ranking. For WikiRfA, among 3949 candidates, 1885 and 18 users are elected once and twice respectively. For WikiElec, there are 2391 candidates, among which, 1223, 11, and 1 users are elected once, twice, and thrice respectively. More times of successful election indicate higher status. The results are shown in Table 4.7 and Table 4.8 with never-elected (in red), once-elected (in black), and twice-elected (in blue), respectively.

We can observe that *SiGCN* and BESIDE both have three twice-elected users on WikiRfA. In particular, the average ranks of these twice-elected users are higher in *SiGCN* than BESIDE. On WikiElec, both PageRank and BESIDE include never-elected candidates, while the top 10 users selected by *SiGCN* are all once-elected. The result indicates

SiGCN can capture better global ranking features.

- PageRank performs the worst overall: with one and two never-elected candidates for WikiRfA and WikiElec respectively, indicating that excluding negative links would not effectively rank the candidates.

- Excluding the overlapped candidates selected by BESIDE and *SiGCN* on WikiRfA, the average number of negative links on the rest four candidates is significantly less in *SiGCN*, 8.75 selected by *SiGCN* comparing to 123.25 by BESIDE. This result shows that *SiGCN* may give lower scores for candidates with a higher number of negative links.

- For the twice-elected candidates on WikiRfA, "Everyking" has the highest number of negative links—334—which might be the reason why this candidate is not included in the top-10 selected candidates by *SiGCN*.

- BESIDE has "halibutt" as one of its top-10 candidates. This candidate has 69 positive and 28 negative links, respectively. Allowing a high ratio of negative links may result in inaccurate ranking.

4.5 Related Work

In this section, we present and discuss some related works on network representation learning and recent advancements in applying graph convolutional neural networks.

4.5.1 Network Representation Learning

The goal of network representation learning is to learn low-dimensional representations for all nodes, which can be used for many different tasks of network analysis, such as link prediction [89], node classification, and community detection. An extensive amount of work has been developed in this area, including Node2Vec [28], Line [74], DeepWalk [66],

and GCN [41], all of which are proposed for unsigned networks. With the prevalence of social media that can be represented as signed networks, signed network embedding has emerged as a promising direction that leverages both positive and negative links to enhance network mining performance on signed networks [35, 39, 79, 87].

Two social-psychological theories, structural balance theory and status theory, have been widely used in mining signed networks. These two theories provide insights into ways in which users use linking mechanisms in social computing applications. Status theory is specialized in directional links, as it posits a status differential from the source node of a link to its target node. Balance theory was initially proposed for undirected networks, though it has been widely applied to directed networks. According to the social theories they built on, we can roughly divide existing works into two categories: structural balance theory-based, including SiNE [87], SIDE [39], SIGNet [35] and SGCN [19]; status theory-based. Most existing works mainly focused on representation learning with structural balance theory, while rare efforts were on signed networks analysis with status theory. BESIDE [16] was the state-of-the-art solution on signed network embedding, which proposed to incorporate both balance theory and status theory.

4.5.2 Graph Convolutional Neural Networks

Graph convolutional neural networks (GCNs) have been proven to be powerful on representation learning on unsigned network graphs [31, 41], leading to new state-of-the-art results on benchmarks such as node classification and link prediction. These GCN-based representation learning on unsigned networks consistently outperformed techniques based upon random walks (e.g, node2vec [28], Line [74], and DeepWalk [66]). Therefore, signed graph convolutional network, SGCN [19] has been proposed to solve the representation learning on signed networks. SGCN introduced the definitions of balanced and unbalanced paths based on the structural balance theory. By providing a recursive definition for calculating the balanced and unbalanced sets, SGCN aggregated and propagated

information, extracted from both positive and negative links, across signed GCN layers.

Despite the compelling success achieved by previous work on signed network embedding, limited attention has been paid to signed directed networks with status theory, not to mention signed GCN based on status theory. Here we fill this gap by proposing a GCN-based framework with status theory, a specialized model for representation learning on signed directed networks.

4.6 Summary

In this chapter, we devised a new framework *SiGCN*, to learn signed network representations based on status theory via graph convolutional networks. The key of *SiGCN* is the newly proposed status convolutional layer, which can jointly capture graph structure and associated sign and direction information of the links. Extensive experiments on four benchmark datasets have demonstrated the rationality and effectiveness of our proposed *SiGCN*. In the meanwhile, it enjoys high efficiency, scalability, and robustness due to the notion of localized graph convolutions.

Chapter 5

Adversarial Attacks against GCN-Based Link Prediction

Link prediction is one of the fundamental problems for graph-structured data. However, a number of applications of link prediction, such as predicting commercial ties or memberships within a criminal organization, are adversarial, with another party aiming to minimize its effectiveness by manipulating observed information about the graph. In this chapter, we focus on the feasibility of mounting adversarial attacks against link prediction algorithms based on graph neural networks. We first propose a greedy heuristic that exploits incremental computation to find attacks against a state-of-the-art link prediction algorithm, called SEAL. We then design an efficient variant of this algorithm that incorporates the link formation mechanism and Υ-decaying heuristic theory to design more effective adversarial attacks. We used real-world datasets and performed an extensive array of experiments to show that the performance of SEAL is negatively affected by a significant margin. More importantly, our experimental results have shown that our adversarial attacks mounted based on SEAL can be readily transferred to several existing link prediction heuristics in the literature.

The remainder of this chapter is organized as follows. We first present some preliminary background in Sec. 5.1. Our attack model and problem formulation are introduced in Sec. 5.2. In Sec. 5.3, we present the details of our algorithm designed to craft ad-

versarial examples for link prediction effectively and efficiently. In Sec. 5.4, we present an extensive array of experimental results to evaluate the performance of our approach. Sec. 5.5 discusses related work and Sec. 5.6 concludes this chapter.

5.1 Preliminaries

Throughout this chapter, we consider link prediction task in a single large graph. Formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ is the set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of observed links/edges. Its observed global adjacency matrix is $\mathcal{A}$, where $\mathcal{A}_{i,j} = 1$ if $(i, j) \in \mathcal{E}$ and $\mathcal{A}_{i,j} = 0$ otherwise. For any nodes $x, y \in \mathcal{V}$, let $\Gamma(x)$ be the 1-hop neighbors of x, $\Gamma^d(x)$ be the set of nodes whose distance to x is shorter than or equal to d, $d = 1, 2, \cdots$ and $d(x, y)$ be the shortest path distance between x and y. Given two nodes $x, y \in \mathcal{V}$, the h-hop enclosing subgraph for (x, y) is the subgraph that induced from $\mathcal{G}$ by the set of nodes $\Gamma^h(x) \cup \Gamma^h(y)$.

Given a graph containing a set of observed links, the goal of link prediction is to learn a function $F : \mathcal{V} \times \mathcal{V} \to \mathcal{C}$ that maps the link existence between two given nodes $(x, y) \in \mathcal{V} \times \mathcal{V}$ to a class c in $\mathcal{C} = \{0, 1\}$, where $c = 0$ implies that the link does not exist (called a *negative link*), and $c = 1$ implies that the link exists (called a *positive link*). For clarity, the link to be predicted is called the *target link* throughout this chapter.

5.1.1 Heuristics for Link Prediction

A large category of link prediction algorithms is based on some heuristics that compute the proximity between nodes to predict whether they are likely to have a link. In this category, each heuristic is predefined and has a strong assumption on when two nodes are likely to have a link. Popular heuristics including common neighbors (CN), Jaccard [46], preference attachment (PA) [4], Adam-Adar (AA) [2], resource allocation (RA) [92], Katz index [38], PAGERANK [7], and SimRank [36]. Table 5.1 summarizes eight popular heuristics and associated heuristic formula, which will be used to analyze

the transferability of our mechanisms. Note that due to the large literature, we could not analyze to every heuristic, but to some popular ones.

Table 5.1: Popular Heuristics for Link Prediction

Algorithm	Heuristic Formula
common neighbors (CN)	$\|\Gamma(x) \cap \Gamma(y)\|$
Jaccard [46]	$\frac{\|\Gamma(x)\cap\Gamma(y)\|}{\|\Gamma(x)\cup\Gamma(y)\|}$
preference attachment (PA) [4]	$\|\Gamma(x)\| \times \|\Gamma(y)\|$
Adam-Adar (AA) [2]	$\sum_{z\in\Gamma(x)\cap\Gamma(y)} \frac{1}{\log\|\Gamma(z)\|}$
resource allocation (RA) [92]	$\sum_{z\in\{\Gamma(x)\cap\Gamma(y)\}} \frac{1}{\|\Gamma(z)\|}$
Katz index [38]	$\sum_{l=1}^{\infty} \beta^l \|path(x,y) = l\|$
PAGERANK [7]	$q_{xy} + q_{yx}$
SimRank [36]	$\gamma \frac{\sum_{a\in\Gamma(x)}\sum_{b\in\Gamma(y)} score(a,b)}{\|\Gamma(x)\|\times\|\Gamma(y)\|}$

Note: $\Gamma(x)$ and $\Gamma(y)$ denote the sets of x and y's one-hop neighboring nodes, respectively; β is a damping factor; $|path(x,y) = l|$ represents the number of length-l paths between nodes x and y; q_{xy} is the station probability distribution of y under the random walk from x.

5.1.2 Graph Neural Networks

Graph neural networks (GNNs) represent a new type of neural networks that are capable of learning from graphs. A graph neural network for graph classification typically consists of two main components: graph convolutional layers that extract local substructure features for individual nodes, and a graph aggregation layer that aggregates node-level features into a graph-level feature vector.

Deep graph neural networks (DGNN) are GNNs equipped with propagation-based graph convolution layers. They have been shown to achieve state-of-the-art graph classification performance on various benchmark datasets [90]. The aggregation layer in a DGNN is a SortPooling layer, which sorts the final node states to obtain an isomorphism invariant node ordering, and enables a traditional 1-D convolutional neural network on

the node sequence. Its last layer is a fully-connected layer followed by a log-softmax layer.

5.1.3 The SEAL Framework

In this chapter, we focus on the problem of crafting adversarial examples for link prediction based on graph neural networks. In particular, we consider a state-of-the-art link prediction framework, called SEAL [89], which learns heuristics from local enclosing subgraphs using a graph neural network. In essence, the SEAL framework is designed to automatically learn a function that maps local enclosing subgraph patterns to link existence. The foundation of this framework is a Υ-decaying heuristic theory, which shows that local enclosing subgraphs reserve rich information for link existence prediction.

Particularly in [89], Zhang *et al.* proposed a Υ-decaying heuristic theory that is able to unify a wide range of heuristics in a single framework, and proved that several existing heuristics, including Katz index [38], PAGERANK [7], and SimRank [36] can be well approximated from local enclosing subgraphs. The Υ-decaying heuristic for (x, y) has the following form:

$$\mathcal{H}(x, y) = \eta \sum_{l=1}^{\infty} \Upsilon^l f(x, y, l) \tag{5.1}$$

where $\Upsilon \in (0, 1)$ is a decaying factor, $\eta > 0$ is either a constant or a function of Υ bounded by a constant, f is a nonnegative function under the given network.

The Υ-decaying heuristic theory for link prediction: Given a Υ-decaying heuristic, if $f(x, y, l)$ satisfies the following two conditions:

- $f(x, y, l) \leq \lambda^l$ where $\lambda < \frac{1}{\Upsilon}$;
- $f(x, y, l)$ can be achieved from the h-hop subgraph of (x, y) for $l = 1, 2, 3, \ldots, g(h)$, where $g(h) = ah + b$, $a, b \in \mathcal{N}$, $a > 0$.

Then the Υ-decaying heuristic for (x, y) can be approximated from the h-hop enclosing subgraph of (x, y) and the approximation error decreases at least exponentially with h.

Following this theory, as illustrated in [89], several existing heuristics inherently share the same Υ-decaying heuristic form, which implies that from the small enclosing subgraphs extracted around links, it is able to approximate a wide range of heuristics with small errors.

In this regard, the SEAL framework is designed to automatically learn a 'heuristic' function that maps local enclosing subgraph patterns to link existence instead of using predefined ones. It contains three stages: extracting an h-hop local subgraph, either for a training or a testing link; constructing the node information matrix (X) for each link, and then learning with a graph neural network. The input of the graph neural network consists of (A, X) tuples, where A represents the adjacency matrix of the subgraph, and the output of the graph neural network consists of link labels c. The optimal model parameters W are learned by minimizing the cross-entropy on the output of the training links. At test time, the link existence of two nodes then can be predicted by applying the trained SEAL model.

For clarity, let's see a running example as illustrated in Fig. 5.1. In this example, the SEAL framework learns the 'heuristic' function using 1-hop enclosing subgraphs. The learned heuristic may contain information about its graph structure features, such as the number of common neighbours, Jaccard, and Katz index, etc. (A, B) and (C, D) are the links with labels and regarded as training links.

Essentially, the node information matrix X contains information about each node, including the structural node labels, embeddings, or node attributes. As the structural node label is used to mark the different roles (topological structure) of nodes in an enclosing graph, it is a kind of graph structural feature. By incorporating the node information matrix, SEAL can learn the mapping function from its graph structure and node attributes. Our work focuses on mounting adversarial attacks on link prediction

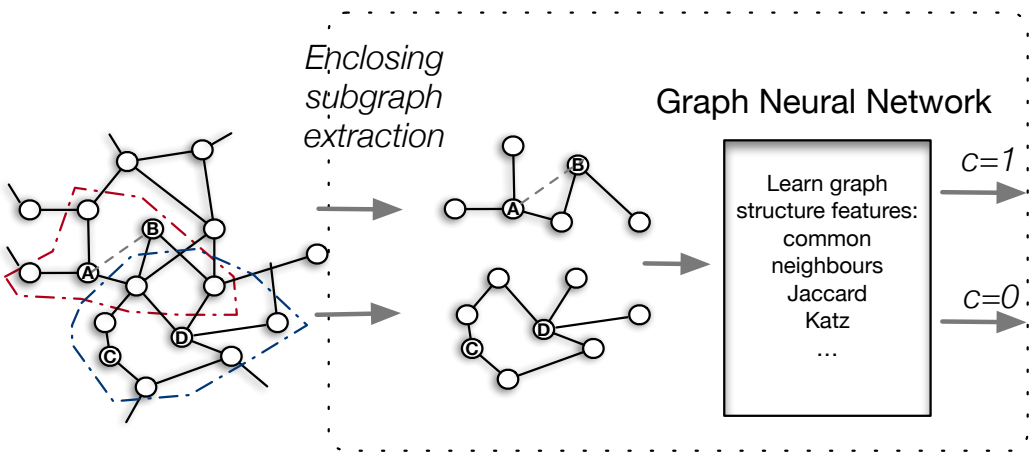


Figure 5.1: The SEAL framework: learning graph structure features from 1-hop local enclosing subgraphs: (A, B) and (C, D) are links with labels and regarded as training links.

algorithms based on graph neural network, particularly targeting the link prediction algorithm in SEAL. We assume that the link prediction model is trained with graph data that is clean and attack-free.

5.1.4 Attack Transferability

Existing work in the literature on adversarial machine learning demonstrated that adversarial examples produced to mislead a specific model are highly likely to mislead other models; such property is referred to as *transferability*. A practical impact of this property is that it leads to oracle-based black-box attacks. More specifically, the adversary is able to use the target model as an oracle to label a synthetic training set for the surrogate, so the adversary need not even observe the full data to mount the attack [53, 64].

The transferability of adversarial machine learning has been extensively studied in the literature [64, 65, 75]. In this dissertation, the definition of *transferability* is more general and not only limited among machine learning models. Note that, we aim to offer a comprehensive algorithmic investigation of the problem of attacking link prediction algorithms based on graph neural networks. For this purpose, we focus on evasion attacks against a GNN-based framework, called SEAL, which is proposed based on a Υ-decaying

heuristic theory. Regarding the Υ-decaying heuristic framework, we can envision that the mounted attacks may be transferred to the heuristics.

To illustrate the effectiveness of our attacks, we will empirically analyze their transferability to existing heuristics in Sec. 5.4.

5.2 Problem Setup

In this section, we will describe our threat model and explain our attacks as modifications to a graph. In practice, the adversary changes the graph based on the underlying data that are explored for the prediction of missing (or unobserved) links. For example, in criminal networks, the adversary aims to hide connections between the entities to avoid being detected in criminal investigations.

5.2.1 Notations

An undirected graph $\mathcal{G}$ is defined by the sets of nodes $\mathcal{V}$ and edges $\mathcal{E}$. $\mathcal{G}$ is such a graph that represents the underlying data — a defender analyses the missing links based on its observed information (e.g., observed graph structure, etc).

Often when applied, a defender learns a prediction model — described as F in Sec. 5.1 — according to its observed graph and seeks to predict the link existence $e(x, y)$ given any two target nodes $x, y \in \mathcal{V}$. In this dissertation, an adversary controls an attacker subset $V_s \in \mathcal{V}$. The adversary is capable of accessing and performing perturbations on this subset within a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, leading to the graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, such that the link prediction model learned is fooled.

5.2.2 Threat Model

Before describing the adversary's knowledge, we first discuss the knowledge that is available to the adversary. We assume that the adversary has an active infection set to manipulate, or $V_s \in \mathcal{V}$. This reflects the real-world situation that some of the links are

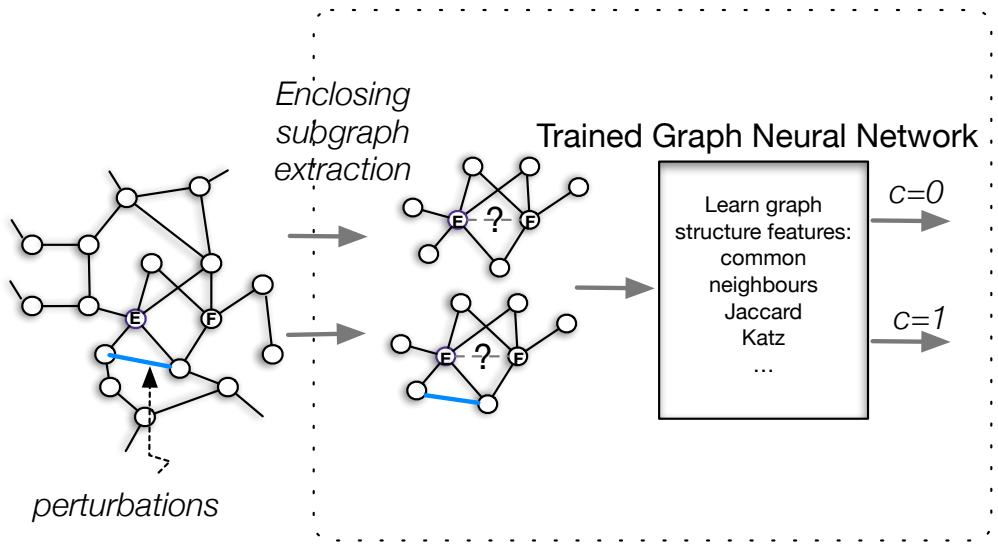


Figure 5.2: Small perturbations on the graph lead to wrong prediction of the target link by SEAL: ((E, F) is a testing link without knowing its label).

intensively monitored and can be easily detected if an attack occurs.

On the other hand, the adversary has full knowledge of the link prediction algorithms, including the generation of the node information matrix, as well as its learning algorithm. In other words, we assume in this dissertation that the adversary has complete access to the graph neural networks, including the architecture and model parameters, and can use them in a *white-box* manner. This is a conservative and realistic assumption: due to the *transferability* property as illustrated in Sec. 5.1.4, it is possible to train a surrogate model given black-box access to a target model, and by attacking the surrogate model, the adversary can transfer the attacks to the target [53, 64].

As in the literature [95], we also assume that the adversary has perfect knowledge of the graph $\mathcal{G}$, obtained from the defender. Given the full dataset and the knowledge of the modeling process, the adversary can completely reconstruct the link prediction results as the defender does to evaluate the effectiveness of their attacks. Ideally, this data would be well guarded, making this level of knowledge only realistic for the most sophisticated adversaries. Nevertheless, considering the damage that could be done by a perfectly knowledgeable adversary is important as a security evaluation, since it allows

us to find potential weaknesses in link prediction models.

Our attack architecture is shown as Fig. 5.2. During the testing phase, a testing link (E, F) is predicted as a negative link, while with a perturbation (adding an edge denoted as a blue link in Fig. 5.2), it is predicted as a positive link. Since X contains information about a node, including node structural labels, manipulating perturbations on the graph structure (adding or deleting edges) would lead to changes in both A and X (coupled variables while performing perturbations). Without loss of generality, attacks induced by adding or deleting edges are referred to as *graph structure attacks*. Directly manipulating the target link is easy to be detected; thus we also assume that the adversary would not add or delete an edge between the target nodes. To summarize, as the inputs of the prediction model are (A, X) tuples representing the enclosing subgraph of two given target nodes x and y, perturbations causing changes on A and X may lead to an incorrect prediction of the target link $e(x, y)$.

5.2.3 Unnoticeability Constraint

In typical application domains, a successful adversarial example is crafted under some simple constraints to ensure its unnoticeability. For example, in the image recognition domain, the perturbation constraint is measured by the distance (l_0, l_1, l_2, l_∞-norm, etc.) between the adversarial example and its normal example. Its effectiveness can be easily verified by human vision [10, 14]. However, in a complex network graph, manipulating the input data to fool its learning model is much harder.

To quantitatively evaluate unnoticeability, we use the perturbation constraint measured by l_1-norm distance of the graph adjacency matrices before and after perturbations. It can be formulated as:

$$|A - A'| \leq \Delta \tag{5.2}$$

where A, A' are the adjacency matrices of the subgraphs before and after perturbations. It sets the maximum bound that the adversary can change the graph, and with this

constraint it is more likely to satisfy the unnoticeability constraint.

Instead of verifying by human vision, we employ the graph property preservation technique to ensure its unnoticeable perturbations, which has been discussed by Zugner *et al.* [95]. Precisely, we use degree distribution preservation to ensure unnoticeable perturbations in the graph — likelihood ratio test for the power-law degree distribution of the two graphs [95]. The intuition is that two highly similar graphs would follow similar power-law behaviour regarding their degree distributions. According to [95], the graph structure perturbations $\mathcal{G}' = (\mathcal{V}', \mathcal{E}, \mathcal{X}')$ can be accepted only when the degree distribution satisfies:

$$\Lambda(\mathcal{G}^{(0)}, \mathcal{G}') < \tau \approx 0.004 \tag{5.3}$$

where Λ denotes the log-likelihood ratio test statistic according to the graphs' power-law degree distributions; it follows a χ^2 distribution with one degree of freedom. τ is approximated using the critical p-value setting in the χ^2 distribution.

5.2.4 Attacks as an Optimization Problem

As is commonly done on evasion attacks in other application domains, such as image [10] and audio [11], we formulate the problem of generating adversarial perturbations for the link prediction task as follows: given any two target nodes x, y, we solve the following problem:

$$\begin{aligned} \underset{(A',X')}{\text{maximize}} \quad & F(A', X')_{c'} - F(A', X')_c \\ \text{subject to} \quad & |A - A'| \leq \Delta \\ & \Lambda(\mathcal{G}^{(0)}, \mathcal{G}') < \tau \approx 0.004 \end{aligned} \tag{5.4}$$

where F is the pre-trained model that the adversary aims to fool, $F(A', X')$ is the output of the *log-softmax* layer of the graph neural network, $\mathcal{G}^{(0)}$ represents the initial graph, (A, X) and (A', X') denote the enclosing subgraphs of the target nodes (x, y) extracted from $\mathcal{G}^{(0)}$ and $\mathcal{G}'$, respectively. c and c' indicate the predicted labels of $e(x, y)$

before and after perturbations. For simplicity, we use $f(A', X')$ to represent the loss function $\{F(A', X')_{c'} - F(A', X')_c\}$, which is designed to measure how close (A', X') is to a successful attack.

5.3 Generating Adversarial Attacks

Solving the optimization problem as illustrated in Sec. 5.2.4 is non-trivial. While the optimization-based problem for adversarial attacks has been addressed in the literature using gradient-based computations [10, 11, 27], these existing solutions are not applicable in our case.

Except for its discreteness property and non i.i.d of the graph data, the perturbation variables used to optimize the objective function are not independent. Precisely, as discussed in Sec. 5.1.3, the node information matrix contains the node structural label, which is one kind of graph structural features. With a graph structure attack, not only the adjacency matrix but also the node information matrix would be changed, which implies that perturbations on A and X are coupled variables in our problem.

In this section, we will describe the algorithms that we use to overcome the challenges of crafting adversarial examples to fool the link prediction model. In particular, the goal of generating adversarial examples in the complex network graph is to mislead the SEAL framework, causing the link predicted results to be incorrect.

5.3.1 Greedy Graph Structure Perturbation

To cope with data discreteness and variable dependencies, we adopt a locally optimal strategy that perturbs the graph one at a time using an optimal way to manipulate the graph structure (by adding or deleting an edge). To be unnoticeable, the total perturbation magnitude is limited by Δ as shown in Eq. (5.2).

For each particular perturbation, we select the one that can achieve the maximum loss function as Eq. (5.4), from its current feasible graph structure perturbation space

— S_{struct} (constructed based on **procedure GGSP** or **procedure OGSP** which will be explained later). After each effective perturbation, the enclosing subgraph of the target nodes is changed and requires to be re-extracted from the perturbed graph. With this renewed subgraph, a new pair of $(A_{xy}^{(t)}, X_{xy}^{(t)})$ is generated as the current state of the subgraph and would be the input of the next perturbation. The graph structure perturbation would terminate either due to a successful attack ($f(A', X') > 0$) or the maximum perturbation constraint Δ.

As we mentioned above, the variables (A', X') that are used to optimize the loss function, regarding the structure attack, are dependent. Thus, the typical approaches of using gradient-based search for each perturbation are not applicable in our case. To solve Eq. (5.4) with dependent variables, the most intuitive way to construct S_{struct} is to employ a heuristic search under its unnoticeability constraint (see line 6 and 9 in **procedure GGSP**).

1: **procedure (GGSP)**: Greedy Graph Structure Perturbation
2: **Input:** Graph $\mathcal{G}^{(t)}(\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$, h-hop subgraph $G_{xy}^{(t)}$.
3: **Output:** S_{struct}
4: $S_{struct} \leftarrow \emptyset$
5: **for all** $u \in \Gamma^{h-1}(x) \cup \Gamma^{h-1}(y)$ **do**
6: **for all** $v \in V_s$ **do**
7: **if** $e(u, v) = 1$ and $\Lambda(\mathcal{G}^{(t)}, \mathcal{G}^{(t)} - e(u, v)) < 0.004$ **then**
8: $\mathcal{G}' \leftarrow \mathcal{G}^{(t)} - e(u, v)$
9: $S_{struct} \leftarrow S_{struct} \cup \{\mathcal{G}'\}$
10: **if** $e(u, v) = 0$ and $\Lambda(\mathcal{G}^{(t)}, \mathcal{G}^{(t)} + e(u, v)) < 0.004$ **then**
11: $\mathcal{G}' \leftarrow \mathcal{G}^{(t)} + e(u, v)$
12: $S_{struct} \leftarrow S_{struct} \cup \{\mathcal{G}'\}$

According to the Υ-decaying heuristic theory [89], given two target nodes, their h-hop enclosing subgraphs are very informative for link prediction, which means the perturbations are likely feasible when they lead to changes on its subgraph; $h = 1$ or 2 is typically sufficient for accurate link prediction. Hence, we only have to inspect the optimal perturbations that can make changes to the h-hop subgraph. In other words, at least one end of the edge added/deleted should be included in the enclosing subgraph to be a possible

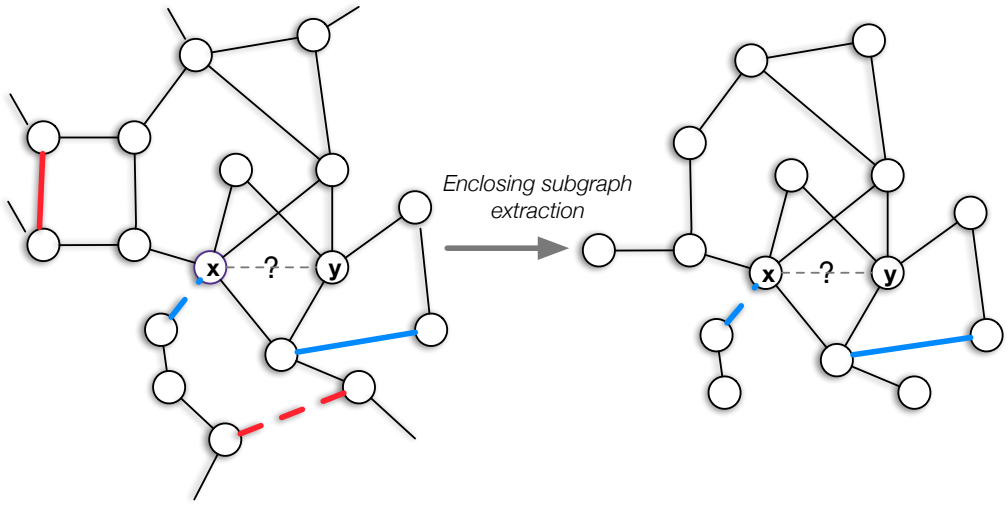


Figure 5.3: An illustration of effective edge perturbations: add perturbations in its global graph (left) and corresponding effects in its subgraph (right), where the red edges denote the ineffective perturbations, the blue edges indicate the effective perturbations, bold dash lines represent edge deletion, and bold solid lines denote addition.

feasible perturbation. Precisely, one end of the perturbed edge should be included in its set of $(h-1)$-hop nodes, denoted as $\{\Gamma^{h-1}(x) \cup \Gamma^{h-1}(y)\}$ (see **procedure GGSP**).

Let's see an example with $h = 2$ as shown in Fig. 5.3. x, y are the target nodes and the link in between is requested for link existence prediction. As shown in Fig. 5.3, only when one end of edge added/ deleted (the blue lines) is included in their 1-hop node set (1-hop neighbours of x/ y), the perturbations can lead to changes to the subgraph. However, the ends of the perturbed edges (the red lines) are not in their 1-hop node set (not 1-hop neighbours of x/ y), could not change the subgraph.

The search time complexity in **procedure GGSP** would be $\mathcal{O}(|V_s| \times (|\Gamma^{h-1}(x)| + |\Gamma^{h-1}(y)|))$, where $|V_s|$ is the number of nodes that the adversary is able to manipulate and it can reach N as the capability of the adversary increases. With rapidly growing volumes of data, the size of the graph (N) is typically very large; for example, there were 300 millions of Amazon customer accounts in 2018 as reported [1]. Even we only consider the search space from its h-hop subgraph, the perturbation search time cost is still very high. Can we further improve our attack efficiency? The answer is affirmative.

[1] https://expandedramblings.com/index.php/amazon-statistics/

1: **procedure (OGSP)**: Optimized Graph Structure Perturbation
2: **Input:** Graph $\mathcal{G}^{(t)}(\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$, h-hop subgraph $G_{xy}^{(t)}$.
3: **Output:** S_{struct}
4: $S_{struct} \leftarrow \emptyset$
▷ decrease common neighbours
5: **if** $F(A_{xy}^{(0)}, X_{xy}^{(0)}) = 1$ **then**
6: **for** $\{\forall u, v \in V_s \cap \{\Gamma^h(x) \cap \Gamma^h(y)\}\} \wedge \{e(u,v) = 1\}$ **do**
7: **if** $\Lambda(\mathcal{G}^{(t)}, \mathcal{G}^{(t)} - e(u,v)) < 0.004$ **then**
8: $\mathcal{G}' \leftarrow \mathcal{G}^{(t)} - e(u,v)$
9: $S_{struct} \leftarrow S_{struct} \cup \{\mathcal{G}'\}$
▷ increase common neighbours
10: **if** $F(A_{xy}^{(0)}, X_{xy}^{(0)}) = 0$ **then**
11: **for all** $u \in V_s \cap \{\Gamma^h(x)/\Gamma^h(y)\} \wedge v \in \{\Gamma^{h-1}(y)/\Gamma^h(x)\}$ **do**
12: **if** $\{\Lambda(\mathcal{G}^{(t)}, \mathcal{G}^{(t)} + e(u,v)) < 0.004\} \wedge (u,v) \neq (x,y)$ **then**
13: $\mathcal{G}' \leftarrow \mathcal{G}^{(t)} + e(u,v)$
14: $S_{struct} \leftarrow S_{struct} \cup \{\mathcal{G}'\}$
15: **for all** $u \in V_s \cap \{\Gamma^h(y)/\Gamma^h(x)\} \wedge v \in \{\Gamma^{h-1}(x)/\Gamma^h(y)\}$ **do**
16: **if** $\{\Lambda(\mathcal{G}^{(t)}, \mathcal{G}^{(t)} + e(u,v)) < 0.004\} \wedge (u,v) \neq (x,y)$ **then**
17: $\mathcal{G}' \leftarrow \mathcal{G}^{(t)} + e(u,v)$
18: $S_{struct} \leftarrow S_{struct} \cup \{\mathcal{G}'\}$
19: **for all** $v \in V_s / \{\Gamma^h(x) \cup \Gamma^h(y)\}$, $\forall i \in \Gamma^{h-1}(x)$, and $\forall j \in \Gamma^{h-1}(y)$ **do**
20: **if** $\Lambda(\mathcal{G}^{(t)}, \mathcal{G}^{(t)} + e(i,v) + e(v,j) < 0.004$ **then**
21: $\mathcal{G}' \leftarrow \mathcal{G}^{(t)} + e(i,v) + e(v,j)$
22: $S_{struct} \leftarrow S_{struct} \cup \{\mathcal{G}'\}$

5.3.2 Optimized Graph Structure Perturbation

Inspired by the intuition of link formation mechanism — the more common neighbours two target nodes have, the more likely they are connected — we construct S_{struct} based on the common neighbours that the target nodes share. In this context, a *common neighbour* is defined as the intersectional neighbours of the two target nodes within their h-hop subgraphs.

We consider two different kinds of attacks when constructing S_{struct}. On the one hand, to force a positive link to be a negative link (link hidden), we delete edges to reduce the number of common neighbours in the subgraph (see line $4-8$ in **procedure OGSP**). On the other hand, to encourage a negative link to become a positive link, we add edges to force more nodes to become the common neighbours of the target nodes. Precisely,

we first consider the nodes in the h-hop subgraph but are not common neighbours. For these nodes, we add one edge for each perturbation under the unnoticeability constraint (see line $10-17$ in **procedure OGSP**). Besides, we consider the nodes that are not included in the h-hop subgraph. In this case, we add two edges simultaneously and force the nodes, outside of the h-hop subgraph, to become the common neighbours under the unnoticeability constraint (see line $18-21$ in **procedure OGSP**).

Regarding the search time complexity, the best-case complexity is $\max(\mathcal{O}(|\Gamma^h(x)|), \mathcal{O}(|\Gamma^h(y)|))$, which can be achieved when only considering link-hidden attack. The worst-case complexity is $\mathcal{O}(|V_s| \times (|\Gamma^{h-1}(x)| + |\Gamma^{h-1}(y)|))$, which is equal to the average complexity of **procedure GGSP**. The benefit of **procedure OGSP** is more significant in applications where the adversary is more focusing on hiding links.

5.4 Experimental Evaluation

We have conducted an extensive array of experiments to evaluate our proposed methods, including our greedy algorithm (GGSP) and its efficient variation (OGSP). Our results show that both of them are able to reduce the availability of the SEAL framework significantly, achieving strong performance on various datasets. More importantly, our experimental results have also shown that our adversarial attacks mounted based on SEAL can be readily transferred to several existing heuristics in the literature. We run the experiments 5 times and then use the average attack success rate (ASR) and the average AUC as our evaluation metrics. To make direct comparisons, we use the same model architectures as SEAL shown in Table 5.3, where k is set to ensure that 60% of the subgraph nodes are larger than k [89, 90].

Datasets. We have selected four datasets as the benchmarks to evaluate our methods. The datasets statistics are given in Table 5.2. USAir is a network of US Airlines [5], which average node degree is 12.81. NS is a collaboration network of researchers in network science [62], which average node degree is 3.45. Celegans [81] is a neural network of

Table 5.2: Dataset Statistics and AUC using SEAL

Network	#nodes/ #edges	#training/ #testing	AUC
USAir	332/2,126	3,400/424	0.959
NS	1,589/2,742	4,386/548	0.959
Celegans	297/2,148	3,436/428	0.885
PB	1,222/16,714	26,742/3,342	0.940

C.elegans, which average node degree is 14.46 and PB is a network of US political blogs [1], which average node degree is 27.36.

Similar to SEAL, we split the existent links randomly into a positive training set (80%) and testing set (10%). As for negative sets, we randomly sample an equal number of non-existent links as the negative training set and testing set, respectively. We retrain SEAL for 50 epochs for each dataset, and select the model with the smallest loss on 10% validation data; these pre-trained models are used as our target models to mount attacks.

Note that, we remove the edges between the two target nodes in the enclosing subgraphs while we train graph neural network, as did in [89]. This is because these edges would contain the link existence information, while is not available in the enclosing subgraphs of testing links. As observed, we report the model AUC on clean data in Table 5.2.

We report success if the attack produces an adversarial example with the incorrect prediction within the perturbation bound Δ, and the associated perturbed graph still satisfies the unnoticeability constraint. In our experiments, we set Δ as the target link degree, which is the sum of degrees of two nodes. This is inspired by the observation that high-degree links are harder to attack than the low-degree ones.

Within the testing set, we select 10 links with the highest prediction margin, including 5 positive links and 5 negative links, i.e., they clearly are correct predictions (*best-set*). We also select 10 links with the lowest prediction margin (but still correctly predicted), including 5 positive links and 5 negative links (*worst-set*). Finally, we select 20 links

randomly sampled from the links that are correctly predicted, including 10 positive links and 10 negative links, respectively (*random-set*). These will serve as the target links for our attack. By default, the average ASR and AUC are reported according to the prediction results of the links randomly selected.

Table 5.3: Model Architecture of SEAL

Layer Type	Parameter
4 Graph Convolutions + Tanh	32, 32, 32, 1 channels
Max-K SortPooling	k
1-D Convolution + ReLU	16 output channels, filter size 2, step size 2
1-D Convolution + ReLU	32 output channels, filter size 5, step size 1
Dense Layers	128 units
Log-Softmax	2 channels

5.4.1 Attacks on SEAL

We start by analyzing both of our two algorithms, GGSP and OGSP, by inspecting their influences on link prediction performance of SEAL (with full knowledge of the network graph). In Table 5.4, we report the average ASR and average AUC over 5 runs when performing attacks on SEAL. For each run, we use the *random-set* as our target links. We can see GGSP achieves very high ASR on NS — 100%, and its AUC degrades to 0.000. Even OGSP has an attack performance degradation on this dataset, it still can decline its model AUC to 0.038.

In Fig. 5.4, we report how our methods affect different testing sets (*best*, *worst* and *random*). We define the *prediction margin* as the difference between the ground truth label probability with SEAL and the target label probability. The smaller prediction

margin indicates better attack performance (margin less than 0 indicates a successful attack). As observed in Fig. 5.4, most of the average margins are under 0, represented as 'star'. *best-set* is harder to attack as its initial prediction margin are large (typically close to 1). Overall, it still can achieve reasonably good attack performance with our algorithms.

Table 5.4: Average Attack Success Rate/Average AUC

Data	GGSP	OGSP
USAir	96.3%/0.050	98%/0.000
NS	**100%/0.000**	79%/0.038
Celegans	87.5%/0.133	82%/0.166
PB	82.5%/0.207	78%/0.294

Furthermore, to inspect and compare the time cost of our two attack algorithms, we also report the average attack time per link across different datasets. As Fig. 5.5 shows, OGSP is way more efficient than GGSP; it can even be approximately 10× faster on Celegans. Note that our time cost is averaged over the target links, containing 50% positive links and 50% negative links. We suspect that OGSP can achieve even better performance at runtime while the adversary is more focused on hiding links.

To analyze the performance of our attack with respect to the adversary capability, we run four sets of experiments when $|V_s|$ are in different settings for each dataset. We set the number of nodes that the adversary is capable of manipulating as 25%, 50%, 75% and 100% of the entire node set, respectively. For each run, the node set V_s is selected randomly. As shown in Fig. 5.5, as $|V_s|$ becomes larger, GGSP produces better results. As observed, even with only 25% perturbable nodes, our algorithm can still achieve a high ASR.

We further report how the target link surrounding structure affects the attack perfor-

Table 5.5: Average ASR (GGSP)

$\lvert V_s\rvert/N$	USAir	NS	Celegans	PB
0.25	66.3%	97.5%	72.5%	69.9%
0.50	91.3%	**100**%	81.3%	79.3%
0.75	92.5%	**100**%	84.3%	81.2%
1.00	96.3%	**100**%	87.5%	82.5%

mance. We run three sets of experiments in USAir when the adversary can perturb 25% of the entire node set. The target links are categorized according to their link degrees. As seen in Table 5.6, the target links with higher degrees achieve lower ASRs, indicating that higher-degree links are harder to attack. The ASR of the target links in the range $[1:30)$ can achieve as high as 90.96%.

Table 5.6: ASR: Target Link Surrounding Structure Complexity

Degree range	$[1:30)$	$[31:90)$	$[90:\infty)$
#target links	188	93	78
Attack success rate	90.96%	64.52%	43.58%

Inspecting an adversarial example. Fig. 5.6 illustrates a real adversarial example mounted for NS. We first randomly select a pair of nodes as the target nodes (grey); the link (dash line) is to be predicted using the pre-trained SEAL. Fig. 5.6a shows its 1-hop enclosing subgraph; the link is initially predicted as $c = 0$ by SEAL, indicating the link is negative. Fig. 5.6a shows its 1-hop enclosing subgraph with our attack method. The edge in blue is suggested to be added by our attack. Even just with this edge addition, the link is predicted as positive.

Transferability of attacks. Note that, our overall objective is to offer a comprehensive

Table 5.7: Transferability (AUC)

Data		**SEAL**	CN	Jaccard	PA	AA	RA	Katz	PR	SimRank	WLK
USAir	Clean	0.967	0.926	0.902	0.843	0.943	0.947	0.910	0.920	0.790	0.960
	Attacked	**0.002**	0.139	0.346	0.827	0.591	0.246	0.780	0.187	0.083	0.517
NS	Clean	0.987	0.934	0.934	0.631	0.934	0.934	0.934	0.934	0.935	0.982
	Attacked	**0.000**	**0.000**	**0.000**	0.178	**0.000**	**0.000**	**0.000**	0.100	0.170	0.467
Celegans	Clean	0.872	0.834	0.782	0.745	0.849	0.853	0.849	0.881	0.751	0.880
	Attacked	**0.233**	0.263	0.094	0.703	0.187	0.152	0.300	0.367	0.190	0.508
PB	Clean	0.940	0.908	0.859	0.899	0.913	0.916	0.919	0.934	0.766	0.929
	Attacked	**0.277**	0.564	0.151	0.881	0.510	0.400	0.590	0.603	0.082	0.500

study on the ability of an "adversary" to manipulate link prediction via adversarial machine learning. For this purpose, we analyze the transferability of the adversarial attack, generated based on SEAL, to existing heuristics, including common neighbors (CN), Jaccard [46], preference attachment (PA) [4], Adam-Adar (AA) [2], resource allocation (RA) [92], Katz index [38], PAGERANK (PR) [7], SimRank [36] and WLK [72]. We report the associated average model AUC over 5 runs, including the model AUC tested on clean graph data and model AUC tested on attacked graph data.

Note that, to be comparable, we trained SEAL purely using graph structure features (the node information matrix contains structural node labels only as did in [89]). Then we mount the adversarial attacks using our efficient design — OGSP. As shown in Table 5.7, the performance of most existing heuristics is even worse than random guessing (the model AUC is less than 0.5), indicating that our attacks can be readily transferred to several existing heuristics in the literature. Noticeably, among all the datasets, the mounted attacks perform extremely well for NS. We can observe that the mounted attacks failed transfer to preference attachment. We suspect that is because the heuristic used

by preference attachment only considers individual node degree of the target node and does not contain any neighbour information, which is not consistent with the typical link formation mechanism we considered.

Defense against attacks. The mounted attacks violate the fundamental assumptions used for link predictions, i.e., the Υ-decaying theory and link formation mechanism, it is very hard to completely eliminate the problem. As often applied in the image domain, adversarial training would force the model to assign both clean and adversarial examples to the same output labels [37]. This raises the idea of adding adversarial examples into the training set while we are training the model, which we leave for our future work.

5.5 Related Work

In line with the focus of this work, we briefly describe deep neural networks (DNNs) for complex network graphs aiming to solve the link prediction tasks.

Link prediction algorithms: A large number of link prediction algorithms have been introduced in the literature. Existing approaches can be categorized into two classes. The first is heuristic methods which use predefined similarity functions to measure the likelihood of links [2, 36, 38, 54]. Although they worked well in practice, these heuristics make strong assumptions on when links may exist, and none of them performs consistently well across all complex networks [54].

The second is learning-based methods, which automatically learn a mapping function from the network [3, 78, 88, 89]. Weisfeiler-Lehman Neural Machine (WLNM) proposed by Zhang *et al.* is the first attempt to employ DNNs for link-prediction tasks [88]. In particular, it learns general graph structure features by encoding enclosing subgraphs of the training links into fixed-size adjacency matrices and trains a fully-connected neural network on the adjacency matrices. As the typical neural networks only accept fixed-size tensor, WLNM can only learn the link local patterns from the subgraphs with fixed-size.

Following this work, Zhang *et al.* proposed the SEAL framework for link prediction

using graph neural network, called SEAL [89]. It has been shown its state-of-the-art prediction performance empirically and theoretically. In particular, a Υ-decaying theory has been developed to unify a wide range of existing heuristics for this task, which proved that local subgraphs are informative for link prediction.

Adversarial attacks on link prediction: Zhou *et al.* investigated the problem of mounting attacks on several heuristics for link prediction. In this work, they further categorized the heuristics based on the maximum hop of neighbours needed to calculate the similarity score. For each category, they proposed associated attack approaches via deleting edges only. Chen *et al.* proposed an iterative gradient attack against graph auto-encoder (GAE)-based link prediction [13]. In contrast, our attacks are mounted based on SEAL under "unnoticeability" constraint and are more general, as they perform well in several link prediction heuristics.

Adversarial attacks on machine learning: Huang *et al.* categorized attacks in adversarial machine learning as either *causative* or *evasion*, with the former poisoning the training dataset and the latter evading the pre-trained model by crafting adversarial examples [34]. Following the terminology of Huang *et at.* , our work focuses on the *evasion* attacks that aim to create adversarial examples deliberately to mislead the state-of-the-art link prediction framework — SEAL, yet still preserving its unnoticeability.

Adversarial attacks have been shown their capability of significantly degrading the performance of deep learning models in various application domains, e.g., image [10], audio [11] and malware detection [27]. These applications consider the data instances to be independent and classify an instance based on features extracted from only that instance. This directly enables evasion techniques such as gradient descent/ascent directly in the feature space. For complex network graph as we considered in this dissertation, data instances (e.g., links) are interrelated and treated as non i.i.d data.

Adversarial attacks against graph data. Works on adversarial attacks against complex network graph for GNN-based graph learning tasks are exceedingly rare, in contrast

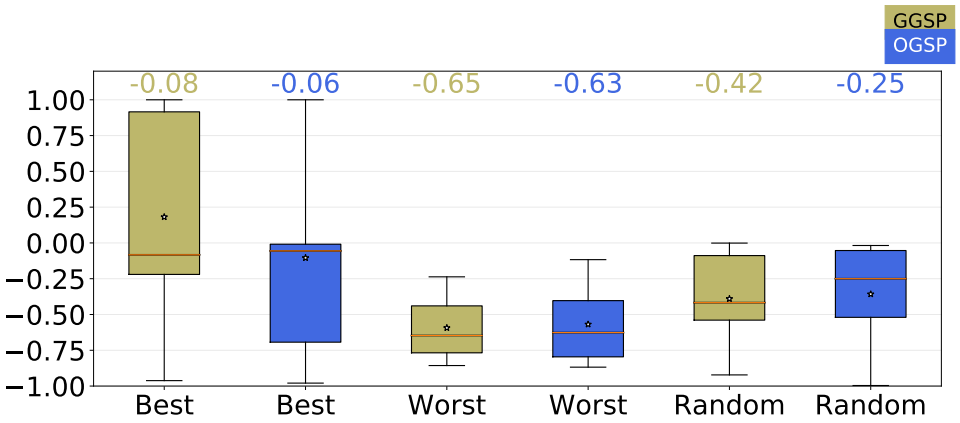
to other application domains, not to mention adversarial attacks particularly for link prediction. Very recently, Zugner *et al.* first proposed the work on adversarial attacks against network graph particularly for node classification tasks [95]. As for the classification model, they focused on a transductive learning setting. Zugner *et al.* 's work is inherently related to the causative/poisoning attacks [34]. Following this work, Zugner *et al.* investigated the problem of training-time attacks on the overall performance of node classification via the principle of meta learning [96].

Along with the work of Zugner *et al.* , Dai *et al.* proposed to employ reinforcement learning approach to attack in both the graph-level learning tasks and the node-level classification tasks [18]. Unlike Zugner *et al.* 's causative/poisoning attacks, Dai *et al.* 's work focused on the evasion attacks particularly against node/graph classification tasks. Except for Dai *et al.* 's evasion attacks, Wang *et al.* studied evasion attacks against collective classification methods via manipulating the graph structure [77]. In contrast, our work focuses on adversarial GNN-based link prediction, which deals with coupled perturbation variables and non i.i.d graph data. We also evaluate the different capabilities of the adversary with various perturbable sets, and analyze its transferability to existing link prediction heuristics.

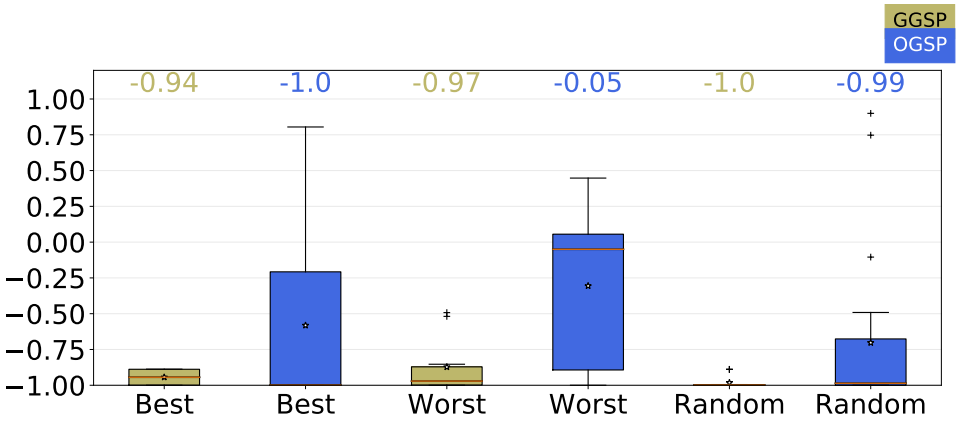
5.6 Summary

We have shown that adversarial attacks on graphs can break the SEAL framework that uses GNNs for link prediction. These attacks can often achieve significantly higher attack success rates, and can degrade the model performance by a substantial margin, making the prediction error even worse than random guessing. By incorporating the Υ-decaying theory and the link formation mechanism, our attacks can be generated efficiently and effectively. The prediction performance is consistently exacerbated, even when the adversary's capability is restricted to only parts of the network graph. Based on our extensive experiments, we show that the attacks mounted based on GNN-based link prediction can

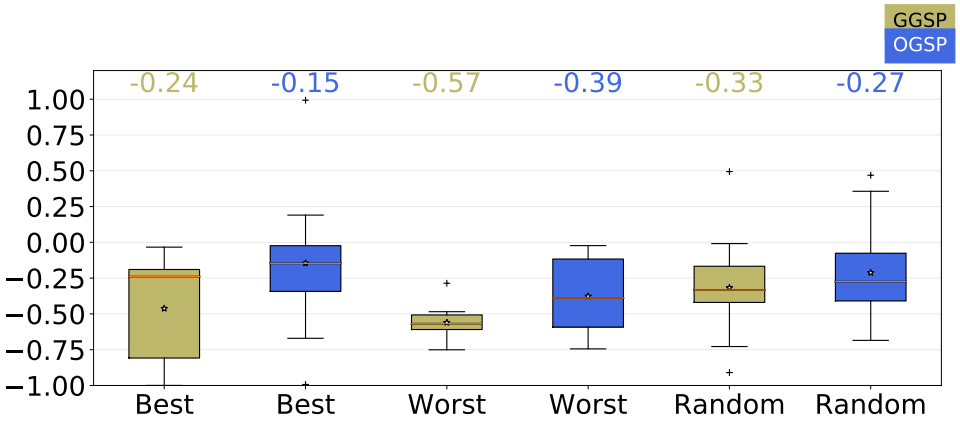
be transferred to several existing heuristics with our design.



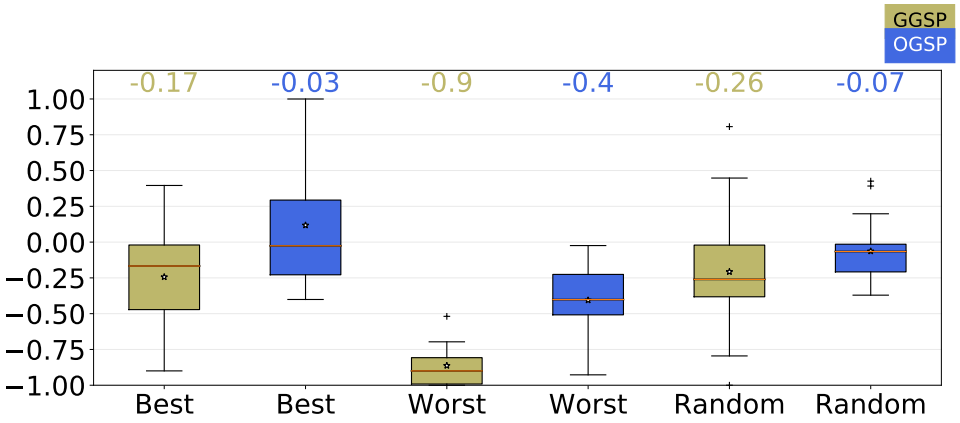
(a) USAir.



(b) NS.



(c) Celegans.



(d) PB.

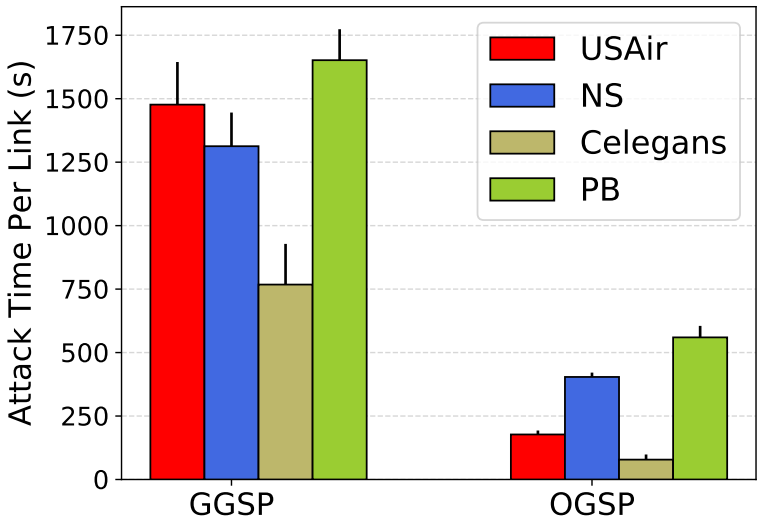
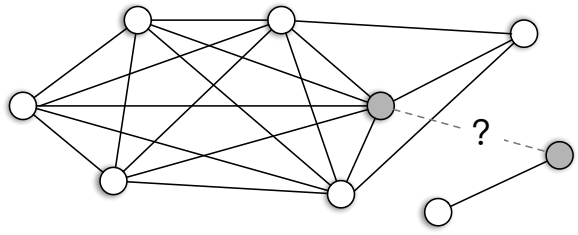
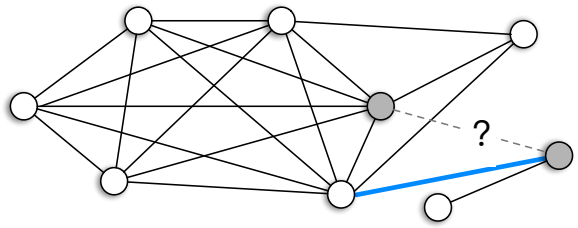


Figure 5.5: Average attack time per link: GGSP vs. OGSP.



(a) Clean graph: $c = 0$



(b) Attacked graph: $c = 1$

Figure 5.6: Adversarial example from NS: given two target nodes (in grey) randomly selected from NS, the link in between is un-observed (a) its initial 1-hop subgraph was predicted as 'non-existed' by SEAL; (b) its 1-hop subgraph after perturbation (just one edge perturbation — blue line — in this case) was predicted as 'existed' by SEAL.

Chapter 6

Concluding Remarks

6.1 Conclusion

In this dissertation, we have investigated the following important research problems centering on the theme of social relationship study, particularly social trust in online social networks, involving social trust evaluation with graph convolutional networks, representation learning with graph convolutional networks when the social relationships are modeled as signed networks and the robustness of used techniques, graph convolutional networks.

We first study the social trust evaluation problem with graph convolutional networks, with the objective of accurately and efficiently estimating the value of trustworthiness between any two users in online social networks. Though graph convolutional neural networks (GCNs) have shown to be powerful in learning on graph data, which provide great potential to the study of social trust, it is challenging to directly apply GCNs to evaluate social trust in online social networks due to its inherent complexity nature. Specifically, online social networks not only contain the social graph structure (social connections between users) but also include pairwise social trust relationships.

To address the challenges, we first study the inherent properties of social trust relationship — the propagative nature, composable nature, and asymmetric nature. We then proposed *Guardian*, to address the challenges in social trust evaluation based on

graph convolutional neural networks. *Guardian* is an end-to-end framework that stacks multiple trust convolutional layers, which is designed to discover hidden and predictive latent factors of trust in online social networks. We demonstrated the effectiveness and efficiency of our proposed framework using two online social networks from different domains — Advogato and Pretty Good Privacy. Our extensive array of experiments on benchmarking datasets demonstrated that *Guardian* can speedup trust evaluation by up to $2,827\times$ with comparable accuracy as compared to NeuralWalk [51], and increase accuracy by up to 18.8% and 19.8% compared with Matri [85] and OpinionWalk [50], respectively.

Then we focus on the problem of representation learning when the social relationships are modeled as signed networks. Yet, representation learning in signed networks using graph convolutional neural networks is challenging. GCNs were originally designed to capture the homophily nature of unsigned networks, which is not applicable when both positive and negative links are jointly considered. To address the challenges, we proposed *SiGCN*, a new framework that learns representations of users for signed directed networks with graph convolutional neural networks. *SiGCN* is designed based on *status theory*, which provides an organizing principle of signed links for signed directed networks [44].

Then we demonstrated the effectiveness and efficiency of *SiGCN* using four signed directed networks from different domains. Our extensive array of experiments on benchmarking datasets demonstrated that *SiGCN* can speedup representation learning for link sign prediction by up to $6.5\times$ as compared with the baselines. More specifically, *SiGCN* speeds up to $4\times$ faster and achieves comparable accuracy as compared to BESIDE [16], increases accuracy by up to 18.8% compared with SIDE [39], and achieves the state-of-the-art robustness and scalability as reported in the literature. We also show that *SiGCN* can learn effective status scores of each user, which can be used for link sign prediction and node ranking and yield state-of-the-art performance.

We further study the robustness of used methodologies, graph convolutional networks,

by studying the ability of an "adversary" to manipulate link prediction based on graph convolutional networks. We formulate the problem of crafting adversarial examples to deceive graph convolutional neural networks-based link prediction models as an optimization problem. In particular, we focus on evasion attacks against a state-of-the-art link prediction algorithm, called SEAL [89], which learns missing/unobserved links from local enclosing subgraphs. We first proposed a greedy heuristic that perturbs the network graph incrementally by manipulating the graph structure. We then proposed an efficient variant that utilizes the link formation mechanism and the Υ-decaying heuristic theory. To validate the effectiveness of our crafted attacks, we use real-world datasets to perform an extensive array of experiments. Our results have shown convincing evidence that the performance of link prediction in SEAL has been negatively affected by a significant margin using our adversarial attack, even with very limited knowledge of complex network graphs. More importantly, our experimental results have also shown that our adversarial attack can be readily transferred to several link prediction heuristics in the literature.

6.2 Future Directions

Our work on social trust study in online social networks and the robustness of graph convolutional network-based methodologies are by no means complete.

Explaining the predictions of GCNs. Despite their capabilities for machine learning on graphs, GCNs still lack transparency as they do not easily allow for human-intelligible explanations of their predictions. Understanding the reasons behind the outcome of graph convolutional networks is, however, quite important and useful for several reasons. Firstly, it can be used to assess the "trustworthiness" of the model. Determining trust in individual predictions is a fundamental problem, especially when the model is used in decision-critical applications pertaining to fairness, privacy and other safety challenges. For example, when using GCNs for medical diagnosis or terrorism detection, predictions cannot be acted upon on blind faith, as the consequences may be catastrophic.

Secondly, the ability to understand GCNs' predictions allows users to get an insight into the network characteristics, identify and correct systematic mistakes made by models before deploying them "in the wild." However, prediction explanations, particularly for graph convolutional network-based learning models, are far from maturity and require further analytical investigations, engineering innovations, and implementation efforts.

Exploring the robustness by exploring the reliability of model explanation. Including our adversarial attacks on link predictions, a growing body of work is exploring the robustness question from the security perspective by proposing attacks — methods for crafting adversarial examples and by proposing defenses, methods for making GCNs robust to adversarial attacks on graph-structural data. However, it is still unclear the reliability of the explanations in highlighting the true cause of the predictions, under these carefully constructed adversarial attacks. We are interested in adversarial attack algorithms that not only fool the machine learning models but also fool the network interpretation.

Bibliography

[1] Robert Ackland et al. Mapping the US Political Blogosphere: Are Conservative Bloggers More Prominent? In *BlogTalk Downunder 2005 Conference, Sydney*. BlogTalk Downunder 2005 Conference, Sydney, 2005.

[2] Lada A Adamic and Eytan Adar. Friends and Neighbors on the Web. *Social Networks*, 25(3):211–230, 2003.

[3] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link Prediction using Supervised Learning. In *SDM06: Workshop on Link Analysis, Counter-Terrorism and Security*, 2006.

[4] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[5] Vladimir Batagelj and Andrej Mrvar, 2006.

[6] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam DI Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. A 61-Million-Person Experiment in Social Influence and Political Mobilization. *Nature*, 489(7415):295, 2012.

[7] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[8] Sergey Brin and Lawrence Page. Reprint of: The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer networks*, 56(18):3825–3833, 2012.

[9] Emrah Budur, Seungmin Lee, and Vein S Kong. Structural Analysis of Criminal Network and Predicting Hidden Links using Machine Learning. *arXiv preprint arXiv:1507.05739*, 2015.

[10] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proc. IEEE Symposium on Security and Privacy (S&P 2017)*, 2017.

[11] Nicholas Carlini and David Wagner. Audio Adversarial Examples: Targeted Attacks on Speech-to-Text. *arXiv preprint arXiv:1801.01944*, 2018.

[12] Dorwin Cartwright and Frank Harary. Structural Balance: a Generalization of Heider's Theory. *Psychological review*, 63(5):277, 1956.

[13] Jinjin Chen, Ziqiang Shi, Yangyang Wu, Xuanheng Xu, and Haibin Zheng. Link Prediction Adversarial Attack. *arXiv preprint arXiv:1803.06373*, 2018.

[14] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In *Proc. AAAI Conference on Artificial Intelligence (AAAI 2018)*, 2018.

[15] Yizheng Chen, Yacin Nadji, Athanasios Kountouras, Fabian Monrose, Roberto Perdisci, Manos Antonakakis, and Nikolaos Vasiloglou. Practical Attacks against Graph-Based Clustering. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*, 2017.

[16] Yiqi Chen, Tieyun Qian, Huan Liu, and Ke Sun. Bridge Enhanced Signed Directed Network Embedding. In *Proc. CIKM*. ACM, 2018.

[17] Karen S Cook, Toshio Yamagishi, Coye Cheshire, Robin Cooper, Masafumi Matsuda, and Rie Mashima. Trust Building via Risk Taking: A Cross-Societal Experiment. *Social Psychology Quarterly*, 68(2):121–142, 2005.

[18] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial Attack on Graph Structured Data. In *Proc. International Conference on Machine Learning (ICML 2018)*, 2018.

[19] Tyler Derr, Yao Ma, and Jiliang Tang. Signed Graph Convolutional Networks. In *Proc. ICDM*. IEEE, 2018.

[20] Yuxiao Dong, Jie Tang, Sen Wu, Jilei Tian, Nitesh V Chawla, Jinghai Rao, and Huanhuan Cao. Link Prediction and Recommendation across Heterogeneous Social Networks. In *Proc. IEEE International Conference on Data Mining (ICDM 2012)*, pages 181–190, 2012.

[21] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph Neural Networks for Social Recommendation. *arXiv preprint arXiv:1902.07243*, 2019.

[22] Peixin Gao, Hui Miao, John S Baras, and Jennifer Golbeck. Star: Semiring Trust Inference for Trust-Aware Social Recommenders. In *Proc. International Conference on Recommender Systems*. ACM, 2016.

[23] Xavier Glorot and Yoshua Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proc. International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[24] Jennifer Golbeck. Combining Provenance with Trust in Social Networks for Semantic Web Content Filtering. In *International Provenance and Annotation Workshop*. Springer, 2006.

[25] Jennifer Golbeck, James Hendler, et al. FilmTrust: Movie Recommendations using Trust in Web-Based Social Networks. In *Proc. International Conference on Consumer Communications and Networking*. IEEE, 2006.

[26] William Josiah Goode. *The Celebration of Heroes: Prestige as a Control System*. University of California Press Berkeley, 1978.

[27] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial Examples for Malware Detection. In *Proc. European Symposium on Research in Computer Security (ESORICS 2017)*. Springer, 2017.

[28] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proc. SIGKDD*. ACM, 2016.

[29] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of Trust and Distrust. In *Proc. WWW*. ACM, 2004.

[30] Guibing Guo, Enneng Yang, Li Shen, Xiaochun Yang, and Xiaodong He. Discrete Trust-Aware Matrix Factorization for Fast Recommendation. In *Proc. International Joint Conference on Artificial Intelligence*. AAAI Press, 2019.

[31] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Proc. International Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[32] Fritz Heider. Attitudes and Cognitive Organization. *The Journal of Psychology*, 21(1):107–112, 1946.

[33] Fali Huang. Building Social Trust: A Human-Capital Approach. *Journal of Institutional and Theoretical Economics (JITE)/Zeitschrift für Die Gesamte Staatswissenschaft*, pages 552–573, 2007.

[34] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar. Adversarial Machine Learning. In *Proc. ACM Workshop on Security and Artificial Intelligence*, 2011.

[35] Mohammad Raihanul Islam, B Aditya Prakash, and Naren Ramakrishnan. SIGNet: Scalable Embeddings for Signed Networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018.

[36] Glen Jeh and Jennifer Widom. SimRank: A Measure of Structural-Context Similarity. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, pages 538–543, 2002.

[37] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial Logit Pairing. *arXiv preprint arXiv:1803.06373*, 2018.

[38] Leo Katz. A New Status Index Derived from Sociometric Analysis. *Psychometrika*, 18(1):39–43, 1953.

[39] Junghwan Kim, Haekyu Park, Ji-Eun Lee, and U Kang. SIDE: Representation Learning in Signed Directed Networks. In *Proc. WWW*, 2018.

[40] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[41] Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. International Conference on Learning Representation (ICLR 2017)*, 2017.

[42] Jérôme Kunegis, Stephan Schmidt, Andreas Lommatzsch, Jürgen Lerner, Ernesto W De Luca, and Sahin Albayrak. Spectral Analysis of Signed Graphs for Clustering, Prediction and Visualization. In *Proc. SIAM International Conference on Data Mining*. SIAM, 2010.

[43] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting Positive and Negative Links in Online Social Networks. In *Proc. WWW*. ACM, 2010.

[44] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed Networks in Social Media. In *Proc. SIGCHI conference on Human Factors in Computing Systems*. ACM, 2010.

[45] Xiaoming Li, Qing Yang, Xiaodong Lin, Shaoen Wu, and Mike Wittie. ITrust: Interpersonal Trust Measurements from Social Interactions. *IEEE Network*, 30(4):54–58, 2016.

[46] David Liben-Nowell and Jon Kleinberg. The Link-Prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[47] Wanyu Lin, Zhaolin Gao, and Baochun Li. *Guardian*: Evaluating Trust in Online Social Networks with Graph Convolutional Networks. In *Proc. IEEE INFOCOM*, 2020.

[48] Wanyu Lin, Shengxiang Ji, and Baochun Li. Adversarial Attacks on Link Prediction Algorithms Based on Graph Neural Networks. In *Proc. ACM ASIA Conference on Computer and Communications Security (ASIACCS 2020)*, 2020.

[49] Wanyu Lin and Baochun Li. *SiGCN*: Signed Network Embedding Based on Status Theory with Graph Convolutional Networks. In *Under review*, 2020.

[50] Guangchi Liu, Qi Chen, Qing Yang, Binhai Zhu, Honggang Wang, and Wei Wang. OpinionWalk: An Efficient Solution to Massive Trust Assessment in Online Social Networks. In *Proc. INFOCOM*. IEEE, 2017.

[51] Guangchi Liu, Chenyu Li, and Qing Yang. NeuralWalk: Trust Assessment in Online Social Networks with Neural Networks. In *Proc. INFOCOM*. IEEE, 2019.

[52] Guangchi Liu, Qing Yang, Honggang Wang, Xiaodong Lin, and Mike P Wittie. Assessment of Multi-hop Interpersonal Trust in Social Networks by Three-Valued Subjective Logic. In *Proc. INFOCOM*. IEEE, 2014.

[53] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into Transferable Adversarial Examples and Black-Box Attacks. In *Proc. International Conference on Learning Representation (ICLR 2017)*, 2017.

[54] Linyuan Lü and Tao Zhou. Link Prediction in Complex Networks: A Survey. *Physica A: Statistical Mechanics and Its Applications*, 390(6):1150–1170, 2011.

[55] Peter V Marsden and Noah E Friedkin. Network Studies of Social Influence. *Sociological Methods & Research*, 22(1):127–151, 1993.

[56] Paolo Massa and Paolo Avesani. Controversial Users Demand Local Trust Metrics: An Experimental Study on epinions. com Community. In *Proc. AAAI*, 2005.

[57] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of A Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27(1):415–444, 2001.

[58] Guido Mollering. 20 Trust, Institutions, Agency: Towards a Neoinstitutional Theory of Trust. *Handbook of Trust Research*, 355, 2006.

[59] Lik Mui. *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[60] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Connected Colors: Unveiling the Structure of Criminal Networks. In *Proc. International Workshop on Recent Advances in Intrusion Detection*. Springer, 2013.

[61] Surya Nepal, Wanita Sherchan, and Cecile Paris. STrust: A Trust Model for Social Networks. In *Proc. International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2011.

[62] Mark EJ Newman. Finding Community Structure in Networks using the Eigenvectors of Matrices. *Physical Review E*, 74(3):036104, 2006.

[63] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and Ranking Algorithms for Event-Based Network Data. *ACM SIGKDD Explorations Newsletter*, 7(2):23–30, 2005.

[64] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv preprint arXiv:1605.07277*, 2016.

[65] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks against Machine Learning. In *Proc. ACM on Asia Conference on Computer and Communications Security (AsiaCCS 2017)*, pages 506–519. ACM, 2017.

[66] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online Learning of Social Representations. In *Proc. SIGKDD*. ACM, 2014.

[67] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Social Influence Prediction with Deep Learning. In *Proc. SIGKDD*. ACM, 2018.

[68] Sini Ruohomaa, Lea Kutvonen, and Eleni Koutrouli. Reputation Management Survey. In *The Second International Conference on Availability, Reliability and Security (ARES'07)*, pages 103–111. IEEE, 2007.

[69] Michael Sauder, Freda Lynn, and Joel M Podolny. Status: Insights from Organizational Sociology. *Annual Review of Sociology*, 38:267–283, 2012.

[70] Moshen Shahriari and Mahdi Jalili. Ranking Nodes in Signed Social Networks. *Social Network Analysis and Mining*, 4(1):172, 2014.

[71] Wanita Sherchan, Surya Nepal, and Cecile Paris. A Survey of Trust in Social Networks. *ACM Computing Surveys (CSUR)*, 45(4):47, 2013.

[72] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[73] Jiliang Tang, Yi Chang, Charu Aggarwal, and Huan Liu. A Survey of Signed Network Mining in Social Media. *ACM Computing Surveys (CSUR)*, 49(3):42, 2016.

[74] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-Scale Information Network Embedding. In *Proc. WWW*. ACM, 2015.

[75] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The Space of Transferable Adversarial Examples. *arXiv preprint arXiv:1704.03453*, 2017.

[76] Raquel Urena, Gang Kou, Yucheng Dong, Francisco Chiclana, and Enrique Herrera-Viedma. A Review on Trust Propagation and Opinion Dynamics in Social Networks and Group Decision Making Frameworks. *Information Sciences*, 478:461–475, 2019.

[77] Binghui Wang and Neil Zhenqiang Gong. Attacking graph-based classification via manipulating the graph structure. In *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS 2019)*, 2019.

[78] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Relational Deep Learning: A Deep Latent Variable Model for Link Prediction. In *Proc. AAAI Conference on Artificial Intelligence*, 2017.

[79] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. Signed Network Embedding in Social Media. In *Proc. SIAM International Conference on Data Mining*. SIAM, 2017.

[80] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *Proc. The Web Conference 2020*, 2020.

[81] Duncan J Watts and Steven H Strogatz. Collective Dynamics of "Small-World" Networks. *Nature*, 393(6684):440, 1998.

[82] Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. Exploiting Social Network Structure for Person-to-Person Sentiment Analysis. *Transactions of the Association for Computational Linguistics*, 2:297–310, 2014.

[83] Zhaoming Wu, Charu C Aggarwal, and Jimeng Sun. The Troll-Trust Model for Ranking in Signed Networks. In *Proc. International Conference on Web Search and Data Mining*. ACM, 2016.

[84] De-Nian Yang, Hui-Ju Hung, Wang-Chien Lee, and Wei Chen. Maximizing Acceptance Probability for Active Friending in Online Social Networks. In *Proc. SIGKDD*. ACM, 2013.

[85] Yuan Yao, Hanghang Tong, Xifeng Yan, Feng Xu, and Jian Lu. Matri: a Multi-Aspect and Transitive Trust Inference Model. In *Proc. WWW*. ACM, 2013.

[86] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proc. SIGKDD*. ACM, 2018.

[87] Shuhan Yuan, Xintao Wu, and Yang Xiang. SNE: Signed Network Embedding. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2017.

[88] Muhan Zhang and Yixin Chen. Weisfeiler-Lehman Neural Machine for Link Prediction. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2017)*, 2017.

[89] Muhan Zhang and Yixin Chen. Link Prediction Based on Graph Neural Networks. In *Proc. International Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

[90] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An End-to-End Deep Learning Architecture for Graph Classification. In *Proc. AAAI Conference on Artificial Intelligence (AAAI 2018)*, 2018.

[91] Xiaoming Zheng, Yan Wang, Mehmet A Orgun, Youliang Zhong, and Guanfeng Liu. Trust Prediction with Propagation and Similarity Regularization. In *Proc. AAAI*, 2014.

[92] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting Missing Links via Local Information. *The European Physical Journal B*, 71(4):623–630, 2009.

[93] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling Polypharmacy Side Effects with Graph Convolutional Networks. *Bioinformatics*, 34(13):i457–i466, 2018.

[94] Kiyana Zolfaghar and Abdollah Aghaie. Mining Trust and Distrust Relationships in Social Web Applications. In *Proc. International Conference on Intelligent Computer Communication and Processing*. IEEE, 2010.

[95] Daniel Zugner, Amir Akbarnejad, and Stephan Gunnemann. Adversarial Attacks on Neural Networks for Graph Data. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2018)*, 2018.

[96] Daniel Zugner and Stephan Gunnemann. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *Proc. International Conference on Learning Representation (ICLR 2019)*, 2019.