2020

Stochastic Sampling and Machine Learning Techniques for Social Media State Production

Neda Hajiakhoond Bidoki
University of Central Florida

STOCHASTIC SAMPLING AND MACHINE LEARNING TECHNIQUES FOR SOCIAL MEDIA STATE PRODUCTION

by

NEDA HAJIAKHOOND BIDOKI
B.S. Sharif University of Technology, 2010.

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2020

Major Professor: Gita Sukthankar

ABSTRACT

The rise in the importance of social media platforms as communication tools has been both a blessing and a curse. For scientists, they offer an unparalleled opportunity to study human social networks. However, these platforms have also been used to propagate misinformation and hate speech with alarming velocity and frequency. The overarching aim of our research is to leverage the data from social media platforms to create and evaluate a high-fidelity, at-scale computational simulation of online social behavior which can provide a deep quantitative understanding of adversaries' use of the global information environment. Our hope is that this type of simulation can be used to predict and understand the spread of misinformation, false narratives, fraudulent financial pump and dump schemes, and cybersecurity threats. To do this, our research team has created an agent-based model that can handle a variety of prediction tasks.

This dissertation introduces a set of sampling and deep learning techniques that we developed to predict specific aspects of the evolution of online social networks that have proven to be challenging to accurately predict with the agent-based model. First, we compare different strategies for predicting network evolution with sampled historical data based on community features. We demonstrate that our community-based model outperforms the global one at predicting population, user, and content activity, along with network topology over different datasets.

Second, we introduce a deep learning model for burst prediction. Bursts may serve as a signal of topics that are of growing real-world interest. Since bursts can be caused by exogenous phenomena and are indicative of burgeoning popularity, leveraging cross-platform social media data is valuable for predicting bursts within a single social media platform. An LSTM model is proposed in order to capture the temporal dependencies and associations based upon activity information. These volume predictions can also serve as a valuable input for our agent-based model.

Finally, we conduct an exploration of Graph Convolutional Networks to investigate the value of *weak-ties* in classifying academic literature with the use of graph convolutional neural networks. Our experiments look at the results of treating *weak-ties* as if they were *strong-ties* to determine if that assumption improves performance. We also examine how node removal affects prediction accuracy by selecting nodes according to different centrality measures. These experiments provide insight for which nodes are most important for the performance of targeted graph convolutional networks. Graph Convolutional Networks are important in the social network context as the sociological and anthropological concept of 'homophily' allows for the method to use network associations in assisting the attribute predictions in a social network.

To whoever who has taught me something

ACKNOWLEDGMENTS

First, I would like to thank my parents for their unconditional care and support over the years and instilling me with a strong passion for learning and for doing everything possible to put me on the path to greatness. I would like to thank my advisors Dr. Gita Sukthankar and Dr. Alexander Mantzaris and my committee members, Dr. Damla Turgut and Dr. Cliff Zou, for their advice and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The proliferation of social media platforms and the increase in their usage has attracted much research attention. Activity on these platforms often mirrors what happens in the real world, making them a valuable tool for the scientific study of human behavior. Social networks can be extracted from the data on these platforms; they consist of two types of entities: users and content. In the traditional WWW, content is the main entity, but on social media platforms users are dominant first-class entities or are co-equals with content.

These platforms have become important information sharing networks. For instance, Reddit and Twitter boast millions of monthly users and are among the most visited websites in the US. Additionally specialized platforms such as GitHub have been created to support other collaborative activities such as software engineering. The environment can be used for discussion and sharing and is not limited to software development. As of May 2019, GitHub reports having more than 30 million users and over 50 million repositories; it is the largest host of source code in the world. Social networks extracted from these platforms are extremely dynamic, changing over time in response to external events. Predicting the network dynamics is valuable for a variety of tasks including advertising, comment moderation, and bot detection. Culivating an in-depth understanding of online social network structure and growth assists with both evaluating the current systems and designing future social media platforms.

The goal of this research is to contribute to the Computational Simulation of Online Social Behavior (SocialSim) project. SocialSim is a challenge started by the Defense Advanced Research Projects Agency (DARPA) with the goal of engendering a deep quantitative understanding of the global information environment. Our team has created a large agent-based simulation for modeling the spread and evolution of information across multiple social media environments at a global

scale. Leidos has provided several datasets from multiple social media platforms to be used in this modeling effort. This dissertation introduces techniques in sampling, time series modeling, and deep learning that were created to supplement the agent-based simulation.

The work elaborated in this thesis can be divided into three parts. The first part introduces sampling approaches to simulate a social network; these approaches have lower computational complexity than other techniques for network modeling. The second part aims for an effective prediction approach for tracking social network dynamics. Inspired by the recent success of deep neural networks in a wide range of computing applications combined with the availability of data from multiple social networks, we propose a sequence learning model to predict the network dynamics. Next, we investigate the usage of Graph Convolution Networks for predicting unknown node attributes in social networks.

This document is organized into the following parts: (a) discussions of background, data and related work (b) measurements and metrics; (c) a description of our stochastic sampling technique for modeling social network dynamics; (d) our LSTM model for predicting cross-platform bursts of social media activity and (e) an exploration of weak-ties using Simplified Graph Convolutional Neural Network. In the following, we describe each part in detail.

Background, Data and Related Work

This thesis begins with a brief description of the online social platforms that have been investigated in this work and a discussion of research related to this thesis. Chapter 2 presents our motivation for this work as well as the background on online social platforms and their networks. We also describe the data used in this dissertation. In Chapter 3 we present a review of related work from graph theory, sociology, agent-based modeling and deep learning. We detail previous work on

studying and modeling online social networks and the approaches for predicting social network dynamics and evolution.

Measurements and Metrics

To be able to compare our models' performance, we first describe our measurements as well as our metrics. In contrast to most previous studies, we examine multiple social media platforms. Using multiple platforms allows us to identify how activities on one platform affect the others. To capture this correlation, we use various measurements to measure the network characteristics and employ different metrics to compare the results with ground truth and other benchmarks. A measurement is a value corresponding to a data point at a single point in time; metrics reflect the changes. Several measurements and metrics have been employed to evaluate the models. In Chapter 4 we detail our measurements and metrics. These are divided into baseline and network structure measurements. Measurements can be in the form of scalar values, discrete distributions, continuous distributions, time series and ranked lists.

Sampling From Historical Data

Chapter 5 focuses on how historical data from a social platform can be used to simulate its future. Communities are components of great importance within a social network. They are significant for a variety of reasons. Users usually belong to at least one community; they normally tend to interact and share their information or interests with other members of their own community. Therefore, communities are a vital topic of study for understanding information dissemination and acquisition. Our work uses communities to sample data to model the future of the network. We tested multiple sampling approaches which we describe in detail in Chapter 5. Our tests have revealed the truth

of the saying “the world is its own best model” and thus our simulation directly employs sampled historical data to create the simulation at scale.

Burst Prediction

Chapter 6 presents our model for predicting if a burst is going to occur in one of the networks in the future. For this application, our model exploits parallel streams from different social platforms and maintains the correlation between them to address the problem of predicting irregular burst occurrence.

Burst analysis and prediction are fundamental problems in social network analysis. This is due to the fact that user activities have been shown to have an intrinsically bursty nature; yet bursts may also be an signal of topics that are of growing real-world interest. Since bursts can be caused by exogenous phenomena and are indicative of burgeoning popularity, we hypothesize that leveraging cross platform social media data may be valuable for predicting bursts within a single social media platform.

Exploring weak-ties with Simplified Graph Convolutional Neural Networks

The Graph Convolutional Network (GCN) methodological framework can be used to predict user attributes in a social network by considering the features of neighboring nodes. The sociological and anthropological concept of ’homophily’ allows for the method to use network associations in assisting the attribute predictions. In Chapter 7 we explore the value of weak-ties in classifying academic literature with the use of graph convolutional neural networks. We also examine how node removal affects prediction accuracy by selecting nodes according to different centrality measures.

The final chapter concludes the dissertation with a summary of contributions and a description of future research directions.

CHAPTER 2: BACKGROUND AND DATA

In this section, we first give a brief review of the SocialSim challenge, and next we detail the data that was used for our investigation. An external provider (Leidos) harvested the cross platform data used in the SocialSim challenge. Our research was conducted on GitHub, Reddit and Twitter data collected from three different domains described in Section 2.

SocialSim

The Computational Simulation of Online Social Behavior (SocialSim) aims to develop innovative technologies to support a high-fidelity computational simulation of online social behavior with a focus on information spread and evolution. The percentage of the world's population connected via the global information environment is increasing rapidly and as a consequence information spread speed has been significantly boosted. DARPA launched the Socialsim challenge to address these challenges. Our team at University of Central Florida (UCF) is developing an agent-based simulation of information spread and evolution in online social networks. Our comprehensive model employs a combination of many sub models alongside machine learning approaches and data analytics. It uses massive parallel computing to explore, test and validate many models against a large set of target behaviors or phenomena happening in the world. Each sub model relies on one or more psychological theories drawn from previous research. This multi layer development provides a quantitative understanding of global information spread.

GitHub

GitHub is one of the most commonly used services for asynchronous team-based software development. It provides a space for developers to store source code and interact with collaborators to complete software engineering projects. GitHub has more than 14 event types to track the developers' activities. Developers create their repositories and later clone, change, collaborate, receive changes, make code public, and monitor other repositories. GitHub networks are composed of users (e.g. developers) with repositories as nodes and events as links. Following are the most commonly created events (also summarized in Table 2.1).

- Create: represents a created repo, branch, or tag.
- Public: is triggered when a private repository is open sourced.
- Watch: is related to starring a repository.
- Push: triggered on pushes on the files to a repository branch.
- Pull request: lets you inform others about changes you've pushed to a branch in a repository on GitHub.
- Issues: triggered when an issue is assigned, unassigned, labeled, unlabeled, opened, edited, milestoned, demilestoned, closed, or reopened
- Commit Comment: this call is triggered when a commit comment is created.
- Fork: creates a copy of a repository
- Issue Comment: triggered when an issue comment is created, edited, or deleted.
- Release: triggered when a release is published.
- Member: triggered when a user is added or removed as a collaborator to a repository, or has their permissions changed.
- Delete: triggered when when a branch or tag was deleted.
- Gollum: triggered when a Wiki page is created or updated.

Table 2.1: GitHub event types

Event type	For
Create	representing a created repo, branch, or tag
Watch	starring a repository
Push	propagating changes on the files to a repository branch
Pull request	alerting others about changes pushed to a branch
Issues	triggered when an issue is assigned, unassigned, labeled, unlabeled, opened, edited, milestoned, demilestoned, closed, or reopened
Commit Comment	commiting comment
Fork	creating a copy of a repository
Issue Comment	creating, editing and deleting an issue comment
Release	publishing a release
Member	triggered when a user is added or removed as a collaborator to a repository, or has their permissions changed
Delete	deleting a branch or tag
Gollum	creating or updating Wiki page

Reddit

Reddit is a social news propagation and discussion website. Each registered user can submit content including text, image, and links; others can vote content up or down. Reddit users can post content; they can also post on other users' content through subreddits. The Reddit network is composed of users and subreddits as nodes and their events as network links. The following terms are used when discussing Reddit:

- Comment: Users trigger this event when they comment on existing content.
- Post: Users trigger this event when they upload new content or when they create a new content node.

Twitter

Twitter is another commonly used social networking service. People use messages known as "tweets" to interact with each other. Tweets were originally limited in terms of characters. Formerly, they were restricted to 140 characters and later in 2017 this limit was extended to 280 for most languages [4].

Data Domains

At the time GitHub has been reported to have over 37 million users and more than 100 million repositories. It offers plans for free, professional, and enterprise accounts, and its users work on various topics. Out of the wide range of topics, our simulation focuses on three main topics as described below:

- **Software Vulnerability** The Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known information-security vulnerabilities and exposures. Data in this domain was collected by pattern matching against activities related to 2600 CVEs.

- **Cryptocurrency** Data in this domain was collected by all activities related to Cryptocurrency coins such as Bitcoin.

- **Cybersecurity** This data set includes activities related to cyber security, starting with cybersecurity seeds such cybersecurity and Netsec. All of the other cybersecurity activities were identified using Jaccard similarity.

CHAPTER 3: LITERATURE REVIEW

In this chapter, we first review the history of social networks, what characteristics they normally have and why they grow quickly. Then, we provide a review of various approaches that have been presented to model and simulate them. Next, we describe applications of online social networks and prediction tasks in this realm, motivating why providing an effective approach to forecast the future of a social network is a necessary step to building models. Finally, we provide background on neural networks and their applications to prediction tasks.

History of Social Networks

A social network is a network created with the use of Internet-based social media platforms by users with the purpose of staying connected with family, friends, colleagues, professionals, etc. It allows users to communicate with people who are a part of their extended network or upload content to share with their network members [5]. Social networks emerged in 1995 with the creation of Classmates.com that was the first web site allowing its users to communicate with each other. Classmates.com [6] users were limited to link to only those who attended the same schools. Therefore, former classmates used it to stay connected to each other. Later in 1997, a recognizable social media site, Six Degrees [7] enabled its users to upload a profile and to connect to other users regardless of their profile. From then, social networks started to grow quickly. In the early 2000s, several social sites were spawned. They were mainly designed for making new friendships. Match.com, Ryze [8], CyWorld [9] and LinkedIn [10] were among the best known ones. In 2003, MySpace emerged and rapidly became the most famous social network. Later social sites began to expand the networking features and become larger and larger. LiveJournal [11], Zoomr [12], BlogSpot [13], LinkedIn, Ryze, Digg [14] and Reddit were among pioneering sites that added fea-

tures such as multimedia content sharing, blogging features, news aggregation, and professional networking.

In addition to social networks which were mainly used for identifying people with common interests, social coding platforms emerged to change the developer community, ushering in a new era of collaborative coding. GitHub is the largest social coding platform; it was developed by Chris Wanstrath, P. J. Hyett, Tom Preston-Werner and Scott Chacon in February 2008. GitHub provides distributed version control and source code management as well as access control and several collaboration features including bug tracking, feature requests, task management, and wikis [15]. We refer the reader to these papers on the history of social networks [16, 5, 17, 18, 19, 20] for more information.

Social Network Analysis (SNA)

Social Network Analysis (SNA) refers to the process of carrying out a systematic or formal inquiry to discover and examine social structures through the use of networks and graph theory. It characterizes networked structures in terms of nodes and their connections. The more dynamicity a social network exhibits the more difficult it is to understand how it will evolve, change, and adapt [21]. SNA is a powerful tool for capturing a representation of social reality. Many formal models including graph theory and algebraic topology [22, 23, 24] have been employed for social network analysis.

Social Network Models

Existing work on simulating online social networks can be categorized into three main approaches: statistical analysis/machine learning, physics models, and agent-based simulation [25, 26, 27, 28,

29, 30]. Works which use the first method, mainly focus on a particular phenomenon and then fit a statistical model to data corresponding to its information evolution [31, 32]. Within machine learning, recommender systems are commonly employed to predict future network links [33, 34, 35]. Assuming data availability, statistical analysis works well for studying a single phenomenon [36], but can encounter problems at simulating complex interactions. A popular statistical physics approach is based on simulating the behavior of Brownian agents [37]. The Brownian agents are embedded into a specific framework and are restricted by its stochastic physics. As such they can only model the limited range of behavior options encoded into the framework. The third option is to use a multi-agent system to simulate a theoretical social model [38, 39]. This approach is very powerful and can simulate multiple phenomena simultaneously but requires careful parameter tuning.

Prediction in Social Networks

Social media forecasting has been employed for a broad range of purposes. Prediction market models, survey models, and statistical models are among the most popular models that have been applied to election outcomes [40], stock market prediction [41], disease outbreak prediction [42], and to forecast box-office revenues for movies [43]. Social media has been used widely for predicting future outcomes in this context and has been shown as a reliable source of data for prediction tasks [44]. [45, 46] applied prediction models on blogs; web search scope has been investigated in [47, 48, 49]; message boards were the target of [50, 51]. However, to best of our knowledge there are a few works examining cross platform social media prediction.

GitHub Models

One locus of interest is understanding social behavior and teamwork in GitHub communities, using approaches such as regression modeling to investigate key drivers and behaviors in projects and teams [52]. Ecosystems in social coding platforms, emerge from commonalities in programming language and topic, along with code dependencies; it is possible to study their evolution over time using networks extracted from the GitHub event data [53, 54].

In addition, there has been research investigating the impact of utilizing social coding platforms on the software development process [55, 56, 57, 58]. These studies highlight the benefits and challenges of completing complex software development projects in this space. Much of the work in this area utilizes data-driven approaches that leverage the available data to investigate behaviors such as on-boarding, pull-requests, and documentation evolution [59, 60, 61]. While many of the studies on GitHub utilize data-driven techniques to investigate these phenomena, there are also several examples of survey and interview studies that aim to develop a more nuanced understanding of these events [62, 63, 64]. Prior work on GitHub popularity prediction demonstrated that the fraction of fork events a repository has received in the past is an effective heuristic for predicting the relative distribution of fork events across repositories in the future [65]. However this popularity-based model of network evolution was only used to predict the general structure of the repo-user network rather than future event sequences. There has also been research on modifying the recurrent neural network architecture to improve prediction performance. Wu et al. recently introduced a new network architecture, Deep Temporal Context Networks, for predicting social media popularity [66]. Rather than using a single time representation, DTCN uses multiple temporal contexts, combined with a temporal attention mechanism, to improve performance over a standard LSTM at ranking the popularity of photos on Flickr. Other types of prediction techniques, such as point process models, have been used to predict tweet popularity, measured by retweeting [67]. The key

contribution of our work is illustrating the value of cross-repository information, in any prediction model employed.

Burst Prediction

Burst analysis has been applied to many diverse data sources including neural firing activity [68] and social network analysis [69, 70, 68]. Myers and Leskovec found that the dynamics of network structure can be characterized by a stable pattern of changes punctuated by sudden bursts [70]. Since the nature of the Facebook platform facilitates information transmission to many friends at once, thousands of rumors are constantly appearing there. [69] tracks the propagation of these rumors and examines the rate of bursts created by rumors via uploading and re-sharing.

Information sharing cascades tend to be bursty and many approaches have focused on detecting the basic rising-and-falling pattern that characterizes the initial onset of a cascade [71, 72, 73, 74]. [75] used characteristics of a cascade's initial burst to predict whether it will recur in the future. [76] notes that rumors exhibit bursty temporal fluctuations and that this temporal feature has a high predictive power. We employ our recurrent neural networks based model for predicting bursts from cross-platform data.

Graph Convolutional Neural Networks (GCNs)

Recently, Graph Convolutional Networks (GCNs) have attracted the attention of researchers who analyze social networks. They are able to leverage graph structures and use them alongside node feature information to model information diffusion and have shown promising results in this context [77]. The work of [78] introduces how graph based methods can be used with convolutional neural networks (CNNs). These graph based methods are spectrally defined and their use within a

spatial application utilizes recursive polynomials on the graph Laplacian. This enables spectrally motivated approaches to handle heterogeneous graphs. Convolutional neural networks [79] are employed as they provide a good architecture for extracting meaningful statistical patterns in large, high-dimensional datasets and have been successfully applied to many big data applications [80]. The application of CNNs to learn local stationary structures and apply them to hierarchical pattern searches has driven many advancements in ML tasks [81]. A key contribution of [78] is using localized graph filters instead of the CNN localized convolution filter (or kernel) to extend the model to graphs. They present a spectral graph formulation and demonstrate how filters can be defined with respect to individual nodes in the graph with a certain number of 'hops' distance. An introduction to the field can be found in [82], where the reader can find a motivation for the fundamental analysis operations of signals from regular grids (lattice structures) to more general graphs. The authors of [83] show that a shift-invariant convolution filter can be formulated as a polynomial of adjacency matrices. These filters are defined as polynomials of functions of the graph adjacency matrix, which describes an intuitive spatial formulation of the graph convolutional neural network. Our work also utilizes a similar adjacency matrix and builds upon the work of [84] which relies on the adjacency matrix for filtering. The filter uses the adjacency matrix and defines the exponent of the matrix as the degree polynomial. Effectively these exponents in the polynomial represent the number of edges ('hops') from any node.

Allowing for graph data that is heterogeneous is a flexibility which can produce more interesting applications. Our research contrasts the benefits of taking the single hop (defined edges) and the '2-hop' edges; using the '2-hop' edges allows an extended radius of features influence the classification. The concept of using the adjacency matrix powers is also explored in our work where the exponent is taken over a range of values which represents the number of 'hops'. The authors of [85] also use this concept of the hop number that relies on the number of hidden-layers in the neural network; this is the basis for the comparison approach employed in Chapter 7.

Weak-Ties

After the publication of Granovetter's seminal work on the importance of weak ties [86] there have been multiple follow-up studies exploring how social links affect a member's ability to interact with others in the network and how different types of edges serve disparate roles in transmission and collaboration. Weak ties are important in many types of organizational structures; for instance, Patacchini et al. [87] studied the role of weak ties in criminal collusion. Since innovation requires teamwork and collaboration, organizations need to empower their workers to leverage their weak ties as described in [88]. The work of [89] looks at the effect of weak-ties on the job search process. Extracting information from the network of interactions is not a straightforward process and the work of [90] examines how this can be performed. Given recent evidence for increased social contention [91, 92], the research of [93] considers the important question of how weak-ties can facilitate the increase of emotions such as anger on social media. Although weak-ties can be found to play a role in negative situations, there are other contexts where they play an important positive role, such as psychological well being where casual friendships add to happiness (strong ties plus weak ties) [94].

CHAPTER 4: MEASUREMENTS AND METRICS

For this exploratory analysis, we construct and analyze networks from different domains and social platforms. Specifically, we examine user networks on Twitter, GitHub and Reddit during the training data periods. The three targeted domains are CVE, Cyber and Crypto. We analyze these networks using five types of measurements. We begin by providing details on the network construction. We then provide definitions and some interpretations of these measurements. Finally, we describe each of the networks with all the measurement values, annotated with interpretations specific to the measurement values for these networks.

GitHub networks are typically bipartite networks with edges between users and repos. We construct a user-user network based on the bipartite projection on the user nodes' side. This process connects users who have had interactions with common repo / repos. The edges for the Twitter networks are constructed by linking participating users based on retweet / mention / reply activity, after filling in the cascade reconstructions for the retweet activity. Finally for Reddit, the edges are constructed by linking comment authors with the parent comment / post authors. For this exploration, we treat all the networks as undirected and unweighted.

Measurements

We use several measurements which are widely employed in literature for a robust understanding of the network characteristics as follows:

- **Mean shortest path length**: This quantity measures the average shortest path length between nodes in the graph. The shortest path between two points is called the geodesic. It is likely that many geodesics exist between a pair of nodes, but, by definition, they all have the

same length d_{ij}. We consider our GitHub network as an undirected graph. Given N users, the mean path length is

$$\ell = \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij},$$

where the sum is over all pairs of distinct nodes.

- **Clustering coefficient**: measures the nodes tendency to cluster together; it is a way to determine the importance of a user in our network. The average clustering coefficient is a scalar value between 0 to 1 and represents the ratio of the actual number of triangles in the neighborhood of a node to the possible number of triangles.

  For our undirected GitHub projected graph $G = (U; E)$, assuming the neighbor set of $u_i \in U$, denoted by N_{u_i} , to be the set of users connected to u_i:

$$N_{u_i} = \{u_j | (u_i, u_j) \in E\}.$$

  the clustering coefficient of u_i, denoted by $CC(u_i)$, is the fraction of user pairs in N_{u_i} that share an edge and eventually are connected to a common repository:

$$CC(u_i) = \frac{|\{(u_j, u_k)\}|}{\binom{|N_{u_i}|}{2}} : u_j, u_k \in N_{u_i}, (u_j, u_k) \in E. \tag{4.1}$$

- **Degree distribution**: measures the distribution of the number of users that each user is connected to.

- **Assortativity**: with assortativity we measure the preference for a user to attach to others that are similar in terms of degree. Here assortativity is the Pearson Correlation Coefficient (which ranges from -1 to +1) evaluated over all pairs of degree values of users that are linked.

A large positive value of the assortativity coefficient signifies that high degree users tend to get connected to high degree users and similarly low degree users tend to connect to other low degree users whereas a large negative value means the opposite.

- **Modularity**: Modularity values range from -1 to +1 and measures the strength of division of a network into modules. High modularity indicates dense connections between the users within modules but sparse connections between users in different modules. Formally, we calculate it as the fraction of edges that lie in the same community for the given graph minus the fraction of edges in the same community for a null model with equivalent degree distribution.

Several measurements have been employed to evaluate model accuracy via measuring engagement, reputation and trust, and popularity measurements which we elaborate as follows. Measurements include scalar values, discrete distributions, continues distribution, time series sequences and ranked lists.

- **Engagement**: user engagements over time affects their aptitude of information spread. We measure the engagement by the following measurements.
  - The ratio of the probability that a user's next action will be an Issues event to the probability that it will be a Push event as a function of how many previous interactions the user has had with a repo in GitHub
  - Repo growth: the number of daily contributions to a repo as a function of time
  - Number of daily unique contributors to a repo as a function of time
  - Distribution of total events by week day/hour
  - Counts of specific events by repo

  - Number of unique repos that users contribute to
  - Daily contribution counts of the user over time

- **Reputation**:
  - Top K users with the highest ratio of accepted pullRequests for users with at least a certain level of pull request activity
  - Top K repos with the highest ratio of accepted pullRequests for repos with at least a certain level of pull request activity

- **Popularity**:
  - Distribution of watch events across repos
  - Top K of most watched repos
  - Distribution of fork events across repos
  - Top K most popular users: number of fork events plus number of watch events across repos that the users own

Table 4.1 show the measurements and how they were used briefly.

Metrics

The measurements are not only scalar values, but also include time series and ranked lists; hence we use several metrics to compare model performance against real data. Our metrics consist of absolute difference, Jensen-Shannon divergence, Kolmogorov-Smirnov test, Coefficient of determination, Root-mean-square deviation and rank-biased overlap.

- **Absolute difference**: is used to measure the difference between two continuous variables observed in ground truth and simulation.

- **Jensen-Shannon divergence (JSD)**: is used to compare two distributions [95]. It measures the similarity between two distributions observed in the ground truth and the simulation. Assuming P represents the distribution of a target phenomena in the ground truth and S corresponds to the simulated network. Given sample x is to occur in either P or S, the likelihood-ratio (LR) where $LR = P(x)/S(x)$ measures how likely is x happens in ground truth vs. the simulation. If $LR > 1$, $P(x)$ is more likely while $LR < 1$ indicates $S(x)$ is more likely. Formally, the overall ratio for the data set x, is:

$$LR = \prod_{i=1}^{n} \left(\frac{P(x_i)}{S(x_i)}\right) \tag{4.2}$$

Log ratio improves the calculation:

$$log(LR) = \sum_{i}^{n} log\left(\frac{P(x_i)}{S(x_i)}\right) \tag{4.3}$$

Where log(LR) values > 0 indicate that $P(x)$ better fits while values > 0 indicates that $S(x)$ better fits the data. Using this value, we can better quantify how much better one model is over the other by answering how much will each sample on average indicate that $P(x)$ better describes the data than $S(x)$ if you sample from $P(x)$, also called its predictive power.

$$log(LR) = \frac{1}{n}\sum_{i}^{n} log\left(\frac{P(x_i)}{S(x_i)}\right) = \int P(x) log\left(\frac{P(x)}{S(x)}\right) dx \tag{4.4}$$

When $n \to \infty$ the expected value is:

$$D(P(x), S(x)) = E_p(x)\left\{log\left(\frac{P(x)}{S(x)}\right)\right\} \tag{4.5}$$

JSD symmetrizes and smooths this by:

$$JSD(P(x), S(x)) = 1/2D\Big(P(x), 1/2P(x), S(x)\Big) + D\Big(S(x), 1/2P(x), S(x)\Big) \quad (4.6)$$

$JSD(P(x), S(x))$ ranges from 0 to 1 given that the distributions use base 2 logarithms.

- **Kolmogorov-Smirnov test (KS-test)**: The Kolmogorov-Smirnov Test is based on the cumulative distribution function of the underlying distribution [96]. We use it to quantify the distance between the distributions of a given sample from simulation and its reference probability distribution from ground truth. Having the cumulative distribution function $F(x)$, KS-test calculates the maximum absolute difference between the two cumulative probabilities of the ground truth and simulation.

- **Coefficient of determination**: The coefficient of determination (R^2) quantifies how good the simulation is compared to a baseline model with no independent variables that always predicts the expected value of y [97]. We use in regression, one-to-one comparisons between the predicted value and ground truth value.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{\sum_{i=1}^{n}(x_i - \hat{x})^2} \quad (4.7)$$

R^2 is any value between 0 and 1, where values closer to 1 indicate a greater proportion of variance accounted for by the model.

- **Root-mean-square deviation**: Root mean squared error (RMSE) is the measurement of the difference between a predicted value from a simulation and a ground truth value. Its main use is for regression on scalar values. The first step to calculating the RMSE is by calculating the residuals $\hat{y}_t - y_t$.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}} \quad (4.8)$$

The next step is to average the squares of those residuals and then then taking the square root of that average. The purpose of the square and then square root is to remove negative values.

- **Rank-biased overlap**: Ranked-biased overlap (RBO) measures similarity between infinite ranked lists e.g. top K most active users in ground truth and simulation which may or may not contain the same items by calculating overlap at various depths, bounding the average overlap of those depths by using a geometric series, a type of convergent series. RBO falls in the range 0 to 1 where 0 means maximally disjoint and 1 means identical.

Table 4.1: Evaluation measurements

Measurement category	Measurement
Engagement	The ratio of the probability that a user's next action will be an Issues event to the probability that it will be a Push event
	Repo growth: the number of daily contributions to a repo as a function of time
	Number of daily unique contributors to a repo as a function of time
	Distribution of total events by week day/hour
	Counts of specific events by repo
	Number of unique repos that users contribute to
	Daily contribution counts of the user over time
Reputation	Top K users with the highest ratio of accepted pullRequests for users with at least a certain level of pull request activity
	Top K repos with the highest ratio of accepted pullRequests for repos with at least a certain level of pull request activity
Popularity	Distribution of watch events across repos – for how many hops of passive information spread e.g., watches
	Top K of most watched repos
	Distribution of fork events across repos
	Top K most popular users: number of fork events plus number of watch events across repos that the users own
Network topology	Clustering coefficient
	Degree distribution
	Assortativity
	Modularity

CHAPTER 5: MODELING SOCIAL CODING DYNAMICS WITH SAMPLED HISTORICAL DATA

The chapter describes our initial steps towards the creation of an accurate at-scale simulation of the spread and evolution of online information related to cybersecurity threats on social coding platforms [98, 99]. The spread of online information can be positive, encouraging social activism and increased community engagement, but these platforms can also be misused to target vulnerable people, spread misinformation, and propagate dangerous technological information related to weapon-making. We are interested in the use of social media platforms as an avenue for amplifying cybersecurity threats. Not only can social media be used as part of phishing attempts, it can also be used to facilitate collaborative work between developers of software exploits. Our research centers around studying the movement of information related to Common Vulnerabilities and Exposures (CVEs); CVEs are information-security vulnerabilities in publicly released software packages that are assigned a unique identifier by MITRE.

Although information spread has been traditionally studied on platforms that encourage rapid proliferation of content, such as Twitter, we believe that social coding platforms, such as GitHub, exhibit interesting information propagation patterns as well. Analyzing patterns on GitHub is particularly important for understanding cybersecurity threats since it is employed both by developers attempting to counter threats and by hackers developing new exploits.

To that end, we have developed an extensible agent-based model, Deep Agent Framework (DAF) [100] to model the user activity on social media platforms. The initial version of the DAF framework combined agent-based modeling with parameter tuning to simulate social phenomena. DAF relies on a set of behavioral rules inspired by social science theories to determine whether users propagate information. This approach was very successful at modeling large scale information movements

but failed to capture other types of patterns.

For instance, on some social media outlets, humans exhibit repetitive, cyclic usage patterns that are strongly influenced by time of day and the day of the week [101]. This is particularly true of social coding platforms where the software developers submit code changes according to a development schedule [102]. Hence when this social media data is encoded into a network, many aspects of the global network evolution reflect its users' daily habits. To model these patterns, we started mixing the output of our agent-based model with sampled historical data. The sampled historical data is used to simulate periods by "replaying" the data—taking a stream of continuous snapshots of data from a time slot in the past and replicating them within a target time slot in the future. The advantage of directly replaying the past data to simulate the future time series is that it more accurately captures aspects of the data distributions at different granularity levels than a more general model. We believe that cyclic usage patterns exhibit great consistency within a single community of users; closely connected users within the network exhibit similar usage habits since these user communities are often formed by participants with the same goals, interests, or geographic location [103]. Hence global data distributions may be more accurately modeled by composing data from multiple communities.

In this research, we use data from GitHub to evaluate the benefits of our community-based data sampling approach. GitHub is one of the most commonly used services for asynchronous team-based software development. It provides a space for developers to store source code and interact with collaborators to complete software engineering projects. GitHub contains a wealth of data from communities of software developers that collaborate on open source projects. Networks of users working towards similar goals use the communication features built into GitHub to discuss software changes, technological advances, and new vulnerability patches. Researchers also use GitHub to promote their work by releasing open source versions of code developed as part of research projects. A software engineer may work on multiple open source projects using different

Table 5.1: GitHub network structure similarities to Facebook and Twitter

Measurement\Platform	GitHub	FaceBook	Twitter
Degree distribution	1.4	1.5	2.4
Average clustering coefficient	0.203	0.164	0.106
Average path length	6.2	4.2	4.12

programming languages; similarly researchers will often participate in several endeavors. This rich tapestry of social interactions extends beyond merely coordinating version control. Previous studies [104, 105] have noted that the Software Social Network (SSN) hosted on GitHub has many commonalities with conventional social network platforms such as Twitter or Facebook. GitHub users interact with other users through contributing to repositories. These content-generating interactions are comparable to tweeting on Twitter or posting on Facebook. Users interact with each other directly through following or becoming a member of a specific project or indirectly through working with common repositories. [105] showed that social structure and temporal dynamics of key processes on GitHub are similar to other popular social platforms. Phenomena such as popularity, contribution, engagement, influence, reputation and trust can be studied on GitHub as well as Twitter and Facebook. Table 5.1 illustrates some of the comparisons between GitHub, Twitter and Facebook on topological characteristics, based on the aforementioned work.

We compare several methods for forecasting future trends on the GitHub social coding platform. When social coding data is encoded into a network, many aspects of the global network evolution reflect its users' daily habits. Our experiments on data replaying demonstrate that sampling historical data in which samples are drawn from time periods that are similar based on forecasting community-level features outperforms several of the best agent-based models created with the DAF framework.

Problem Formulation

We aim to predict the dynamics of the GitHub network and simulate the activity of its users for a target period of time. The training data contains the users' activities and their interactions with the repositories during the past, and the goal is to predict network dynamics including events triggered by users and the time each event has been triggered during time slice Δt in future. More formally:

Given the large bipartite network of GitHub over a training period $[\tau, T]$ *denoted as* $G_{(\tau,T)} = (U, C, E)$ *in which* U *and* C *represent users and the created repositories and* E *stands for events triggered by users to interact with the repositories where* $E = \{e_{l,t}\}$ *and* l *indicates event type e.g. Fork and* t *denotes the time that the event was triggered, our goal is to simulate* $G_{(T,T+\Delta t)}$ *where* Δt *is a target simulation period.*

Model Framework

In this section, we introduce the framework that we use and then detail the steps in an algorithmic manner. Figure 5.2 shows the overview of our model framework. Our methodology relies on three observations about social coding platforms:

- The software developers submit code changes according to a development schedule. Hence when this social media data is encoded into a network, many aspects of the global network evolution reflect its users' daily habits. Figure 5.1 shows the total set of events users created related to the Crypto domain. They exhibit a smooth trend when measured over a weekly time period; this emerges from a combination of routine events created by current users plus extra events resulting from network growth.

- Closely connected users within the network exhibit similar usage habits since these user

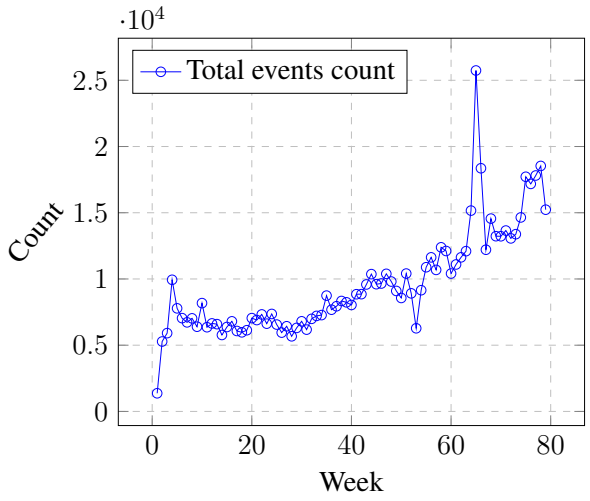


Figure 5.1: Total events created by users in GitHub/ Crypto exhibits a smooth trend

communities are often formed by participants with the same goals, interests, or geographic location [103].

- As communities are a fundamental building block of human social networks rather than modeling social networks as being composed of interacting users, it can be useful to abstract the network as being comprised of communities.

With these assumptions, we put communities under the magnifying glass and investigate their changes over time.

Network sequence generation: Network dynamics can be investigated at any time resolution; we assume Δt is sufficiently large to include both weekdays and weekends. One week includes both weekday and weekends, hence can be a good representative of human activity patterns. As such we defined our time slice as one week. However, time granularity should be selected with regard to the target network. Accordingly, we split the original graph $G_{(\tau,T)} = (U, C, E)$ into the sequence of graphs as $G_{(\tau,T)} = \{g_{(\tau,\Delta t)} \cup g_{(\Delta t,2*\Delta t)} \cup ... \cup g_{(T-\Delta t,T)}\}$. For simplicity we denote it

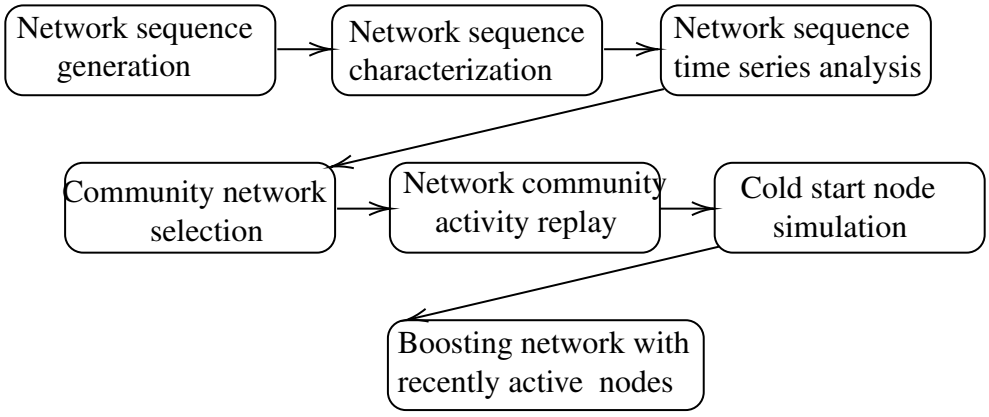


Figure 5.2: Community feature based modeling steps overview

as $G_T = \{g_1 \cup g_2 \cup ... \cup g_T\}$ assuming that T denotes the number of time steps here.

Network sequence characterization: We represent each network by characterizing its communities. There is a large body of work on how to detect communities in both static and dynamic networks (see [106, 107] for an overview). Although most of these approaches are based on analysis of structural or neighborhood properties, for this research we employ a topic based approach using the profiles of the GitHub repositories in order to generate a fixed set of communities. Example topics include programming languages, operating systems, and profile keywords. Communities can be characterized using a set of features such as $F = \{f_1, f_2, ..., f_l\}$ where f_i stands for a feature of interest and l is the number of features. Community dynamics are described by three features: 1) burstiness, 2) modularity, and 3) number of contributing users. Burstiness is the intermittent increase or decrease in activity, or "bursts" in activity as a function of time. We consider the community burstiness as the intermittent increases and decreases in the frequency of all contribution events in a community, measured using inter-event time statistics. Contribution is measured by types of events a user can engage in that represent additions to the repository, such as a *pull request* or a *commit*. *Pull requests* inform all developers involved in a project about changes that the event creator pushed to a branch in a repository on GitHub, and *commit* is an event generated

when an individual changes a file. The number of contributing users is computed by aggregating the users who engage in contributing activity.

User activities have been shown to have an intrinsically bursty nature; yet bursts may also be a signal of topics that are of growing real-world interest. Barabasi describes bursts as one of the most valuable resources with which to study human nature using big data since they illuminate the transition between individual and collective action [108]. Burstiness of inter-contact time between nodes in a time-varying network can be indicative of information spreading processes over the community. The number of contributing users marks the level of involvement of the community—-are the activities mostly generated by a small group of people or the whole community? Modularity is a measure of community structure that measures the proportion of intra/inter community edges. Clauset-Newman-Moore modularity was used to measure how strongly the network resolves into communities or modules [109]. As such, given the set of communities $C = \{c_1, c_2, ..., c_m\}$ in our training data where m is the total number of communities, each community at time step t is represented with a feature vector $V_c^t = [f_{1,c}^t, f_{2,c}^t, ..., f_{l,c}^t]$ in which $f_{l,c}^t$ is the value of feature l of the community c measured at time step t.

Community feature prediction: We prepared time series data of each community's features over the entire training period. To predict the activity of communities as quantified by these features during the testing period, the time series forecasting model, Auto Regressive Integrated Moving Average (ARIMA), was used. Figures 5.3 and 5.4 show an example of forecasting two measurements for a community named as Freelancer for five time steps. Using this forecast, we generate a predicted feature vector for each community as $V_c^p = V_c^{(T+1)} = [f_{1,c}^{T+1}, f_{2,c}^{T+1}, ..., f_{l,c}^{T+1}]$. This feature vector represents the expected trends of community activity in the future network.

Community network selection:

Rather than using the forecasting model to predict all the desired metrics, we use the predicted

feature vectors to identify the weekly segments of the training data that are the best match for test period and then replay those segments. Using the predicted value of each measurement, we aim to find $g_t \in G$ for each $c \in C$ where its nodes and their links best represent c for next time step according to our predicted feature vector. To do so we use Euclidean distance metrics to find the network with the most similar feature vector of c.

$$\arg\min_t f(t) = D(V_c^t - V_c^P), t \in \{1, 2, ..., T\}, \forall c \in C \tag{5.1}$$

where D is Euclidean distance of two feature vectors of community c and V_c^P is the predicted feature vector. This method allows us to retrieve the community data samples that best match our predicted network evolution for each week. We employ the framework shown in Figure 5.5 for each community. Assume that community 1 (C_1) has a feature vector similar to its feature vector within the whole network at $t = 4$ and feature vector of community 2 (C_2) has the minimum distance with its feature vector within the whole network at $t = 7$ and similarly for C_n and $t = 1$. Therefore, N_4, N_7 and N_1 are borrowed to replay C_1, C_2, and C_n user activities respectively. The same process is executed for all pre-defined communities.

Cold start nodes simulation: Cold start is a potential problem in computer-based information systems [110]. Specifically, it concerns the issue when insufficient information has been collected about the users to make recommendations. Many approaches to solve the cold-start problem have been proposed such as profile completion, feature mapping, and hybrid feature weighting (see [111] for review). Unfortunately this data set does not contain user profile information. To address this problem, we augment the sampled historical data with an appropriate number of additional nodes.

The problem with relying exclusively on sampled historical data to simulate the evolving network

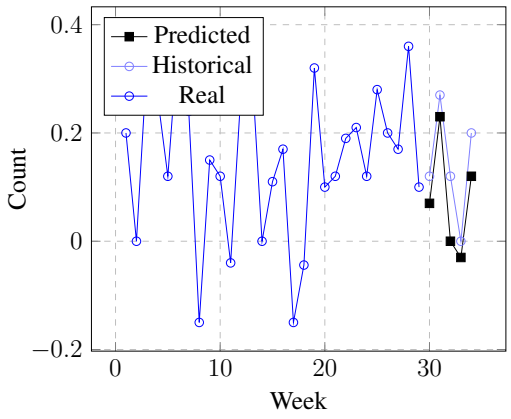


Figure 5.3: Burstiness feature from the Freelancer community as predicted by ARIMA

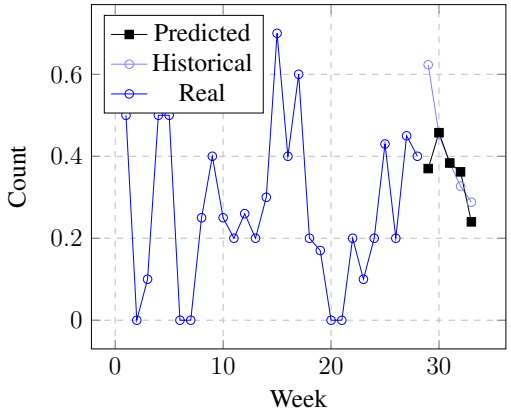


Figure 5.4: Contributing user feature (number of contributing users) from the Freelancer community as predicted by ARIMA

is that most time-evolving networks extracted from social media platforms are rapidly growing. During each time step, a tremendous number of new users join the network and a large amount of content is created. However, there is little information about these nodes' activities. As such we are not able to apply the previous approach of replaying sampled data. However, we can estimate the number of these nodes by employing time series prediction techniques. Figure 5.6 shows the cumulative number of users and content nodes in our Crypto data set. We then fit a curve to the series of new cold start users and content data points. The constructed curve is a smooth function

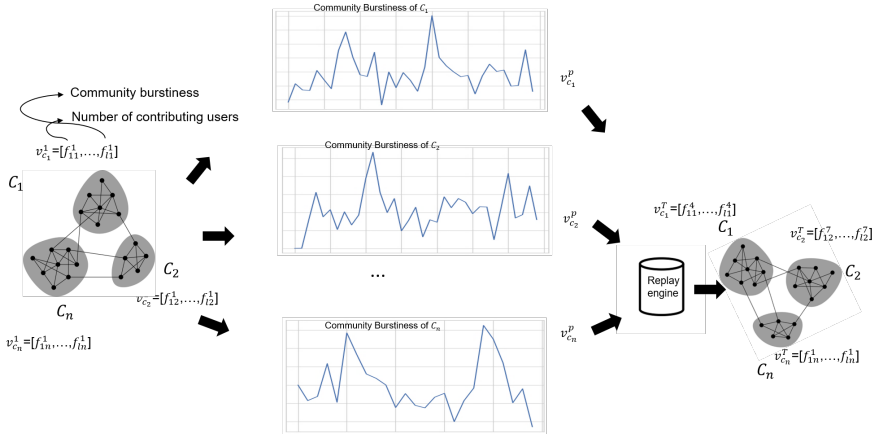


Figure 5.5: Community feature forecasting and replay framework

$F(t_i)$ that approximately fits the data. The number of cold user/content nodes in each time step t_i can be calculated by $F_{U_n}(t_i) - F_{U_n}(t_{i-1})$ and $F_{C_n}(t_i) - F_{C_n}(t_{i-1})$ in which $F_{U_n}(t)$ and $F_{C_n}(t)$ are the total number of active users and content within the network at time step t.

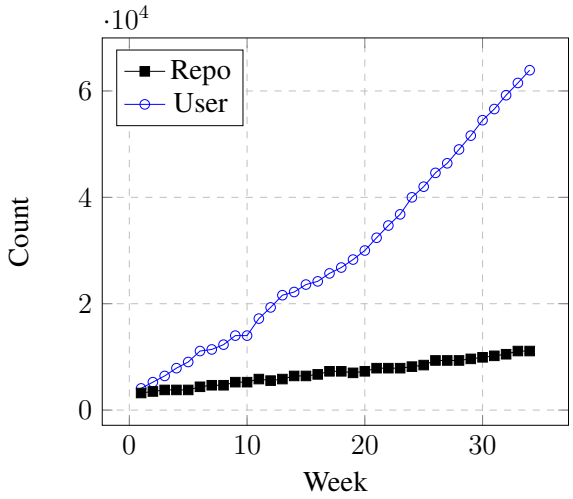


Figure 5.6: Crypto data set cumulative number of user/repo

Boosting network with recently active nodes: Even after selecting the closest data partition based on community features and injecting the new content and user nodes, our method often generates

fewer nodes in the target week than are required to match the time series prediction. To address this problem, we supplement the generated data with samples of recently active nodes. The number of these nodes can be estimated as follows:

$$O_u = E_u(t+1) - N_u - C_u \tag{5.2}$$

$$O_c = E_c(t+1) - N_c - C_c \tag{5.3}$$

where:

- $E_u(t{+}1)$: estimated user population for time step $t{+}1$ using time series prediction techniques (here ARIMA)
- $E_c(t+1)$: estimated amount of content for time step $t+1$ using time series prediction techniques (here ARIMA)
- N_u: number of cold users
- N_c: number of cold content nodes
- C_u: number of users from the detected communities
- C_c: number of content nodes from the detected communities

We investigate the number of common nodes between two sequential time slices using the Jaccard index which is commonly used for computing the similarity of two set of nodes:

$$J(U_i, U_{i+1}) = \mid (U_i \cap U_{i+1}) \mid / \mid (U_i \cup U_{i+1}) \mid \forall i \in T \tag{5.4}$$

where U_i is the set of users present in the network corresponding to time step i. We performed the same analysis with $J(C_i, C_{i+1})$. The results show a significant number of active users in both sets. For the remaining nodes, we select them from the most recent time step, proportional to their activity. For example, those who were active recently, are more likely to be active in the next time step as well and therefore should have a higher chance of being selected. On the other hand, if the number of selected and created nodes is bigger than estimated population ($O_u < 0$ or $O_c < 0$) we drop the extra nodes. Similarly, the assumption is that those who were less active, are less likely to be active in next time step and therefore should have a higher chance to be dropped. For example, some individual users may create a repository on GitHub and then remain inactive for a while. At the end of this step, the nodes match the estimated population of our target time step network. Now we can proceed with the edge prediction as described in the following section.

Replay activities and new links modeling The sampled events are used to create the edges of the network. Looking at the set of nodes, there are some nodes which are cold and some that are not. For users drawn from the sampled historical data, we simply replay the same daily activity within the new time period. In this way we maintain both the event and time of event occurrence distributions to match the original. However, this approach cannot be applied to the cold nodes since there is no information about their activity pattern in the data set. To address this challenge we utilize the event distributions of the historical cold start nodes in the training data.

To clarify, see Figures 5.7 and 5.8. They show the differences between the event distributions for historically sampled users vs. cold users. For example cold users mostly create a watch event during their initial activities whereas the existent users create push events. They have projects in progress therefore use the push command to apply their changes on their projects. By replaying the activities of the old users, the correct event distribution is applied by default. For the cold users, we use the cold nodes event distribution in the training data to generate events. In addition to the type of links, the number of links needs to be determined. For that, we estimate the average degree

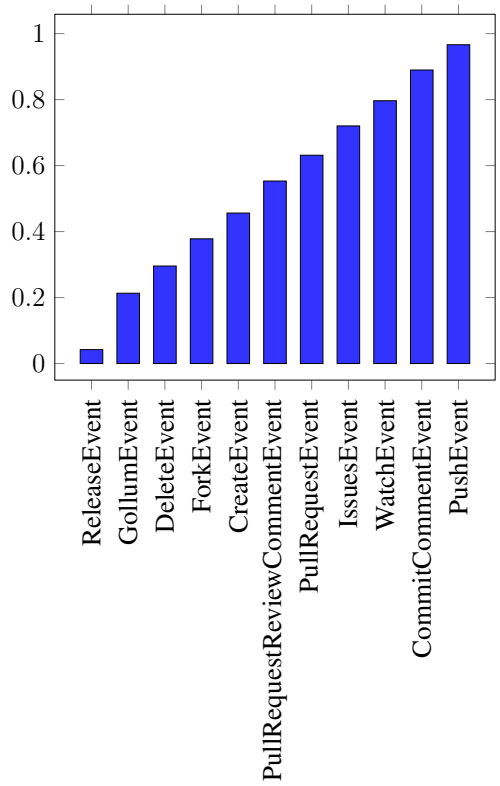


Figure 5.7: Normalized event type counts in Crypto

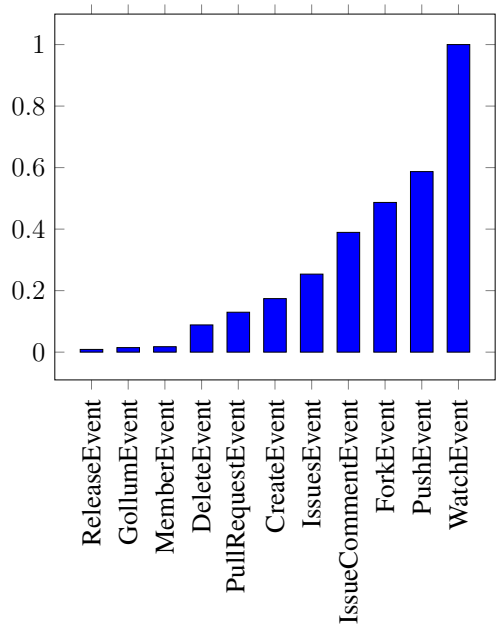


Figure 5.8: Cold users normalized event type counts in Crypto

of cold nodes over time from the training data.

At the conclusion of this step, the events, their types, and their timestamps have been generated. Next, the events need to be attached to node pairs. Here, we utilize the popularity of nodes by employing preferential attachment as follows:

- A new content is:
  - connected to the number of users equal to the average degree of new content nodes in the training data
  - connected to the users proportional to their activity level
- A new user is:
  - connected to the number of content nodes equal to the average degree of historical cold users
  - the activity type and time of occurrence is sampled from the history proportionally to the corresponding distribution
  - a content node is selected from the set of content nodes including both old and cold
  - content nodes are selected proportionally to the attention they have received; if a cold content is selected, the cold user is replaced with another user to prevent exceeding its average degree.

Psuedocode presented in Algorithm 1 summarizes our methodology.

Algorithm 1 PCFM algorithm

Input: The set of social users, U;
The set of users' content, C;
The set of features of interest, F;
Events created by users to interact with the content, E_n;
Time step, τ;

Output: Network representation for, $(T, T+1)$;

$C \leftarrow [c_1, c_2, c_3, \ldots, c_n]$ *identify the communities
$G \leftarrow [g_1, g_2, g_3, \ldots, g_N]$ *generate temporal networks with respect to τ
$V_c^t \leftarrow [f_1^t, f_2^t, f_3^t, \ldots, f_l^t]$ $\forall c \in C$ *measure the temporal features of each community
predict f^{T+1} $\forall f \in V_c$
set $V_c^P = [f_1^{T+1}, f_2^{T+1}, f_3^{T+1}, \ldots, f_l^{T+1}]$
$w_c \leftarrow \arg\min_t f(t) = D(V_c^t - V_c^P)$ $\forall c \in C$ *D is the distance metrics
replay activities from w_c $\forall c \in C$
generate cold start nodes
boost simulated data with replay of recently active users activities
simulate cold start nodes with regard to a) historical cold start nodes statistics b) preferential attachment

Measurements and Metrics

The model accuracy is evaluated via different measurements. Several metrics have been employed to deal with non scalar measurements alongside scalar ones to compare models against the ground truth as well as each other. There are three levels of granularity used to evaluate the simulated output: user/content level, community level, and population level. Measurements include scalar values, discrete distributions, continuous distributions, time series sequences and ranked lists corresponding to nodes engagement, reputation and trust, popularity and network topology measurements. The measurements described here can be at calculated at multiple levels. For example, "Engagement" can apply both at the community level and the user level. The difference is in looking at the engagement of a community as a whole versus each individual in a community or population. Network topology is equivalent to population measurements since it considers the entire network. Popularity can be at content and user level, the popularity of a specific user or the popularity of a specific piece of content. Reputation and trust can be relevant at user, content and

community level. Each combination of measurement and metric are normalized by metric and measurement group so that different levels can be more accurately compared.

The network graphs are constructed having users, content and events created by users. GitHub network is a bipartite network whose vertices composed of two disjoint and independent sets of users and repos. The network is constructed based on the bipartite projection on the user nodes' side. We use several measurements which are widely employed in literature for a robust understanding of the network characteristics. Table 4.1 show the measurements and how they were used.

Since the measurements are not only scalar values we use seven metrics to be able to compare multiple model performance against real data. Each measurement is a different aggregation of the resulting network simulation. Because of this, for each measurement there are a set of metrics that are most appropriate. Sometimes, this is a comparison between two distributions in which JSD and KS is the most appropriate. Other times it is a single value in which the absolute difference is most appropriate. For example, the number of commits during a given time period will be a single value while the number of commits per day would be a distribution in which JSD and KS would be more appropriate. Redundant metrics are removed from evaluation based on how appropriate the metric is for the measurement and level of granularity. For example, some measurements can be evaluated by more than one metric, however we choose the most appropriate metric to the given context. All metrics are normalized by metric type and measurement type. This means that the aggregated metric values are grouped by the type of metric being used and the type of aggregation the measurement requires. Each metric value is then normalized respective of their assigned group.

Table 5.2 show the metrics that have been employed.

Table 5.2: Evaluation metrics

Metrics	Evaluation
Absolute difference	Difference between two continuous variables observed in ground truth and simulation
Jensen-Shannon divergence (JSD)	Two measurements of type distribution
Kolmogorov-Smirnov test (KS-test)	Two measurements of type distribution based on the cumulative distribution function of the underlying distribution
Root mean squared error (RMSE)	The difference between a predicted value from a simulation and a ground truth value
Rank-biased overlap	Similarity between infinite ranked lists e.g. top K most active users in ground truth and simulation

Results

Resolutions and Measurements

To evaluate social media interactions, we focus on four resolutions. The first resolution is at the population level, looking at the social network as a whole. Measurements at the population are related to the network evolution and structure for the predicted time period. The second resolution is at the community level, our primary interest. These measurements focus on community activity as a whole and interactions between users within a community. The next resolution is at the user level. These measurements are specific to user activity and interactions. The final resolution is at the content level. The measurements here are very similar to the user level because they both are represented as nodes in the simulated network.

For each group of measurements, metrics were calculated between the model output and the ground truth (GT). Depending on the measurement type, different metrics were used. Metrics used include Jensen-Shannon Divergence, Kolmogorov–Smirnov test, and absolute percentage error. To compare the approaches over all metrics and measurements, we normalized the results by measurement group, metric type and platform. We then averaged them across the measurements and their respective metrics for the different resolutions. This resulted in the metrics ranging from 0 to 1 in which lower means better performance.

Benchmarks

We compared our modelling approach to three alternative models. Two of these models are previous frameworks designed to simulate social media behavior using the same data, measurements and metrics. Our Proposed Community Features-based Model (**PCFM**) uses community member

identifiers to create weekly data for each community separately. Using this, we created a time series of features to predict the next time step for that community. Therefore a benchmark approach to this model would also be a community feature based model (**CFM**), but simplified [112]. CFM used community based features extracted from the week before the target period to find the week with the highest similarity to replay the weeks proceeding it.

Another benchmark for comparison is the Multi-Action Cascade model
(**MACM**), a diffusion of information model described in [100]. The MACM model simulates the diffusion of information through the network using individual-scale probabilities of social actions. Therefore its features are focused on user-level activity. The model also extracts user interactions, dependent on processing messages received from other connected users. The primary probabilities processed are derived from the Independent Cascade Model [113], using the endogenous p and exogenous q probabilities of activity. All individuals, in addition to being influenced by others, can introduce new content into the network with a probability p, which adds a focus to the content-level resolution.

The final comparison model is the Multiplexity-Based Model (**MBM**) described in [114]. This model focuses on the population-level and user-level resolution. It simulates network evolution represented by a multi-layered bipartite graph with each layer corresponding to an activity a user can perform. The driving forces of this model is preferential attachment and preferential decay, both probabilities are a function of the node's degree and age.

In summary, we compare our approach to a simplified community based model and two models that focus on other resolution types, MACM with user-level and content-level and MBM with user-level and population-level.

Table 5.3: Benchmark methods

	Features Used by Resolution			
Model	Population	Community	User	Content
PCFM		X		
CFM		X		
MACM			X	X
MBM	X		X	

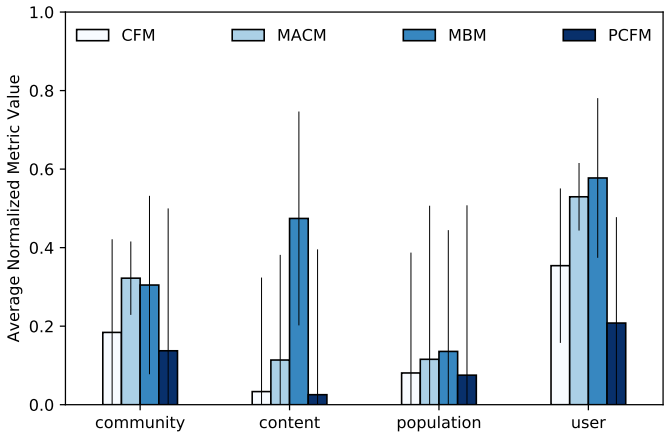


Figure 5.9: Overall performance for each model at each resolution.

Comparison

Figure 5.9 displays the overall performance of each model based on the average, normalized metric value. It is clear that both CFM and PCFM perform better than MBM and MACM on all resolutions. The most improvement between PCFM and CFM is seen at the user-resolution.

The results for population-level measurements are shown in Figure 5.10. PCFM performed on average best at population-measurements that resonate with communities, such as the number of connected components and average clustering coefficient. Focusing on the community significantly improved the network density of the predicted time period for both CFM and PCFM compared to MACM and MBM. The results for community-level measurements are shown in Figure 5.11.

PCFM and CFM outperform MBM and MACM on all measurements on average. As expected, PCFM outperforms CFM in user related community measures. Finally, user-level measures are shown in Figure 5.12. PCFM performed as well as or better than all models on average.

Conclusion

This chapter introduces a versatile community-based data mixture model for predicting social network evolution. Community features are used to forecast and retrieve the most similar historical data partition; new nodes and edges are added to the data based on a combination of intelligent sampling and preferential attachment. Our experimental results show that by simulating communities separately, all resolutions of the network are more accurately simulated. This was supported by comparing community based models to agent-based models that used features focused on other resolutions, MACM and MBM. Both community based models performed better on average on all resolutions, with few exceptions. By comparing the simplified community model, which extracts community based features from the network as a whole rather than per community, our model outperformed in user-level performance. This indicates that by extracting an historical representation of different communities in a social network, the whole network can be more accurately simulated. The strengths of our approach are in its versatility, convenience, and computational complexity. We have been able to generalize our method to model activity on multiple social platforms including Twitter, Reddit, and YouTube.

The main weakness is that, unlike other aspects of the Deep Agent Frameworks [115, 116], it does not provide an explanation for the users behavior nor does it capture peer to peer influence. Thus in practice, we have used sampled historical data to complement our other simulation techniques, rather than as a standalone alternative. Our work could be extended by investigating the use of new features such as diffusion delay of user actions and distribution of user types. Stochastic features

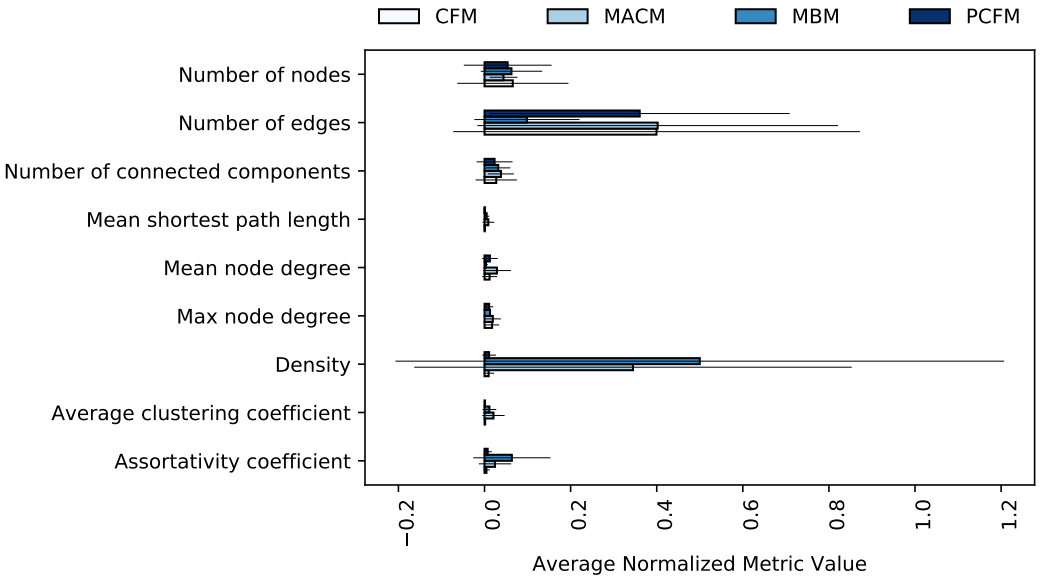


Figure 5.10: Population-level Results

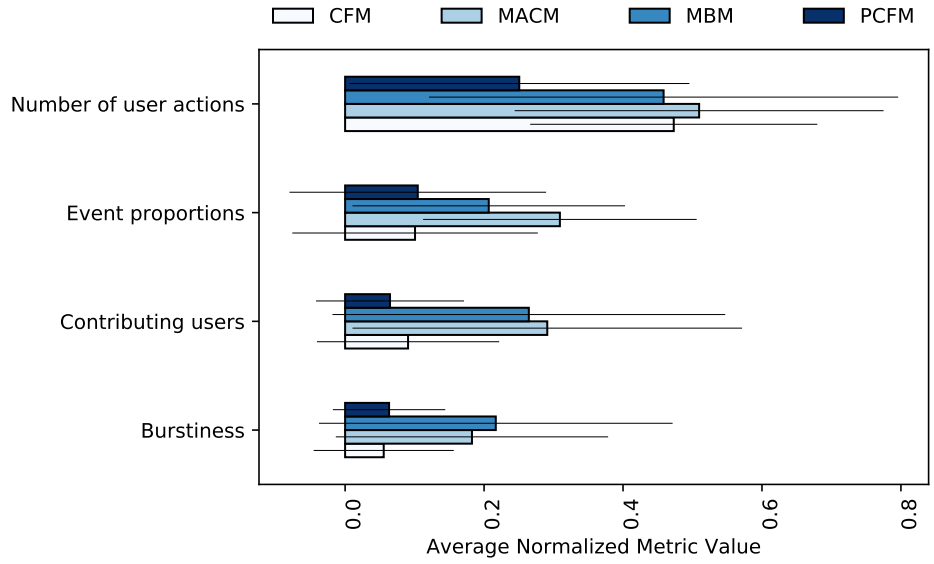


Figure 5.11: Community-level Results

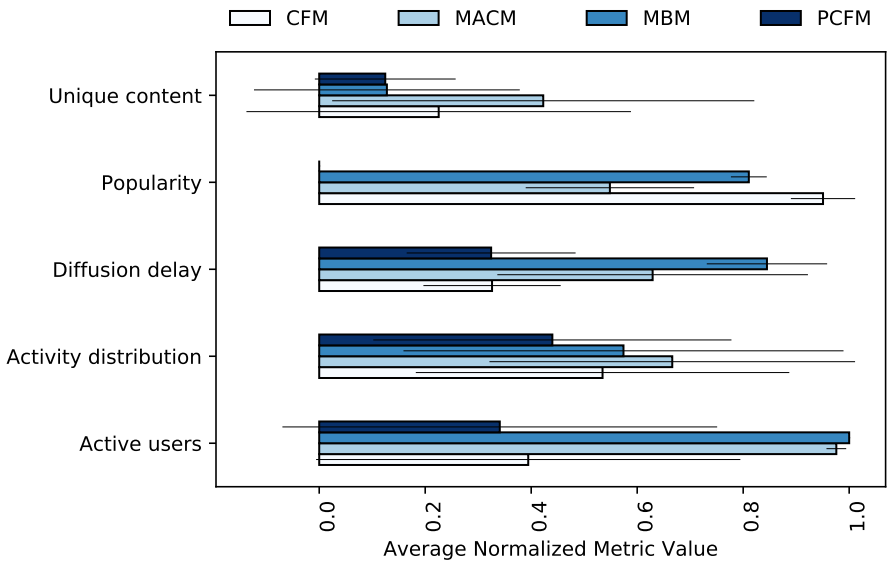


Figure 5.12: User-level Results

such as the probability that users will contribute to information spread could also be employed to represent the network. Increasing the number of features used to represent the network would facilitate the use of machine learning techniques, potentially reducing prediction error.

CHAPTER 6: AN LSTM MODEL FOR PREDICTING CROSS-PLATFORM BURSTS OF SOCIAL MEDIA ACTIVITY

Social media platforms are among the most widely used communication channels and have become an indispensable part of our everyday life, due to the speed and reduction in cost that these services provide to its users [117]. This transformation of the communication modality can affect fundamental dynamics of social engagement such as that of weak ties in the context of online social networks [118, 119]. Information spread through social media can be good for rapidly raising awareness of important issues such as environmental protection [120] but also can be misused for malicious content spreading activities [121]. Over time, this can render society vulnerable to rumors through misinformation campaigns that can polarize communities [122]. Coordinated scams such as pumping cryptocurrency on social media to inflate prices [69, 123] can provide large payoffs.

The seminal work presented in [124] sets a foundation for how 'bursts' of activity are a key component in human dynamics and are an ubiquitous phenomenon in data acquired from social systems. It is intriguing how "intense activity followed by longer periods of inactivity" can manifest in social coding platforms [55] from complex timelines of work interspersed with communication about version control. These changes are non-linear in that they do not follow an accumulated trend from previous time points and also manifest themselves within subcomponents of the network such as the 'boundary-nodes' (community spanners) [125]. Although the non-linearity poses a direct challenge to accurately predicting their occurrences, their impact affect our societies at large. The work presented in this chapter, also appeared in [126], proposes a new model in which activity traces across multiple platforms can be predicted for the platform aggregate and for specific community level associated users (hashtags, repos, and posts). The model is based upon the

LSTM (Long-Short-Term-Memory [127]) model which is a sequential model for the associations of an input output response function with temporal dependencies. The goal is to predict activity trends which fall into the characterizations of 'bursts' or 'spikes' [128, 129] due to the observed real world events that they are stimulated by [130]. The data circulated on these platforms can be broadly encapsulated by the term *content*; this can denote user posts, microblogs, message sharing, or links to other data sources such as repositories on GitHub. It is assumed that these productions and exchanges occur due to both exogenous and endogenous information. There are considerable challenges in tracing the changes between exogenous platform events and the endogenous signals, but it is hypothesized that some of the exogenous signals may be endogenous signals from another platform. This can reduce the degree to which the information of the platform's exchanges are exogenous if the data from another platform are known to be associated with its own content sharing patterns. This hypothesis is explored in this paper, and the results show that sporadic bursts of activity in a platform can be associated with the cross pollination of content sharing between platforms.

The model is applied to the data that comprises activity trace data from multiple platforms. Although the corpus contains more information including anonymous content that can be used to more accurately quantify the relationship significance of network edges, the simplification of topic participation associations is assumed. This also helps promote the search for model explorations which is less likely to infringe upon privacy constraints for ethical usage. The use of GitHub, Reddit and Twitter combines different content-sharing paradigms and use cases. A feature of the data is that there are traces of activity profiles which are related to cyber security which produce activity on all three of the networks and assist in observing the cross platform burst associations. The model is used to examine the temporal distance between subsequent cross platform activities. The results present an overview of the performance of the model in predicting bursts by utilizing trace data from external platform burst information.

Data

To examine the model's ability to encapsulate the platform association of the bursty activity based upon aggregate and topic specific engagements, a large collected corpus of activity traces is necessary. A large dataset is required since the burst activity is not produced by linear dynamics in which gradual changes can be accumulated as evidence between events. Another important consideration for this study is that limited data on the phenomenon can produce conclusions based upon sporadic associations which would not be observed to be consistent in the presence of more data [124].

This study utilizes a dataset which contains trace information from the user activities of three networks, Reddit, Twitter, and GitHub as each is a social network platform with different content sharing and storage option services. We can characterize social media datasets by their size (number of nodes and edges) and by their level of connectivity, as measured by the number of connected components and mean node degree. A *connected component* refers to the set of maximal nodes such that every node is connected by a path. Node degree refers to the number of connections per node. Our Twitter Common Vulnerabilities and Exposures (CVE) dataset contains 8,543 nodes (users) with 14,225 edges (connections), 228 connected components and a mean node degree of 3.33 with the largest being 691. Equivalently for GitHub, there are 16,856 nodes, 207,317 edges, number of connected components 9,263, mean degree of 1.45 and max degree of 24,5986. For Reddit there were 98,855 nodes, 98,855 edges, 86 connected components, and mean degree of 4.3 and max degree of 17,438. The data is collected from user activities corresponding to the following domains; Software Vulnerability, Cryptocurrency, and Cybersecurity. Each domain exhibits distinct patterns of burst production (spikes) that are indications of sudden user activity. The Software Vulnerability and Cybersecurity domains are particularly relevant for system administrators, since discussions about these topics can alert users about potential software susceptibilities. Often solutions and 'patches' to currently affected systems can be shared in a social mechanism that of-

fers rapid response which can therefore produce the observed bursts. The cryptocurrency domain also displays bursts but in a different nature than typical regulated financial products. There is a phenomenon, the 'pump-and-dump' scheme [131], which is used to bring the value of a crypto currency up from the purchase price enjoyed by early investors and adopters so that a rapid sale allows them to profit from a recent increase. This poses a threat to financial stability and can possibly be used for suspicious financial transfers.

The Software Vulnerability domain is referred to as Common Vulnerabilities and Exposures (CVE) in the results and is centered around publicly known information about security vulnerabilities and exposures. There are around 2,600 different CVEs contained in this dataset. There may be similarities in particular CVEs in that malware does evolve from common code bases but this work treats them independently as the model does not assess domain interactions or correlations by looking at the aggregate activity. The cryptocurrency data relates to crypto coins of different popularity such as BitCoin and other less known coins by aggregating them into a single domain based upon the common theme keywords.

Community level activity for bursts is also investigated for the Github platform. This Github data contains relatively distinctive community associations especially around large projects that have corporate sponsorship or support. The community activity may not have 'tags' which are known to define topic categories for networks like Twitter, but there are keywords and meta-data for repositories such as the programming languages employed per repository. From this aggregate (concatenation) of the topic domains a prediction can be produced for the community level activity which is a continuous number for the ratio of active communities over time.

Data Preprocessing

To prepare the CVE data sequence we used accounts for user activities associated with 20 target CVEs. To prepare the community data sequence, we used two different measurements. These measurements are designed to characterize a network at community resolution. Communities were pre-defined and organized by location and topics. Examples of topic based communities include a programming language called Scala and Android. Communities formed by topic are in the domain space of cyber-security. Community based measurements are aggregations of activity at the community level. The measurements look at the activity by users and spread of information within the community. The first measurement is the distribution of user-level burstiness within a community. Burstiness is the intermittent increase or decrease in activity, or bursts in activity as a function of time. The second measurement is the number of contributing users within the community.

Methodology

The methodology proposed to study the cross network burst associations is presented here. A key aspect of the methodology is a pre-processing stage in which the social network platform activity traces over time are transformed into a binary state transition (square wave) based upon the determined temporal presence of a burst. The LSTM is then used to look at the independent platform burst activity traces over time and use the data to predict the bursts in subsequent time steps. This approach is then compared to an independent probability state model in which the state transition probabilities are estimated from an empirical estimate. For community traces this becomes a continuous number representing the scale of the community impacts.

A 'burst' (or 'peak') identification methodology which was employed as a preprocessing stage in order to transform the activity traces from a discrete count for the individual users per platform

over time windows into a binary signal over the same time windows. The methodology of gradient sign change for local minima and maxima was explored initially to see if a simplistic approach was feasible. This involves looking for the gradient along the activity traces and taking the sign changes as indicators for the burst identifications. Although quick to compute and good in avoiding the problem of relative normalization, the method did not prove capable of avoiding false positives. It was susceptible to the sporadic changes in the activity traces which displayed a far greater number of gradient changes than actual bursts. This was expected and abandoned as the methodology should provide an ability to smooth over changes and ignore the minor mode state transitions for the platform level bursts rather than sporadic changes. A parabolic peak detection algorithm was also tested; however our tests show that it displays an excessively large number of components when noise is observed in the signal. A sinusoid based method for peak detection was also explored for finding the local maxima and minima in the signals with a model function: $y = A*\sin(2\pi f(x-\tau))$ where τ is the peak position in time.

We used the Kleinberg burst detection algorithm [132] for analyzing the traces. This technique provides a robust organizational framework to analyze the underlying data in terms of the binary state transitions, whereby the stream is modeled as a two-state automaton. Kleinberg's burst detection algorithm models the stream using an infinite-state automaton A which at any point in time can be in one of an underlying set of states, and emits messages at different rates depending on its state. Specifically, the automaton A has a set of states that correspond to increasingly rapid rates of emission (activity), and the onset of a burst is signaled by a state transition — from a lower state to a higher state (of activity rate). By assigning costs to state transitions, one can control the frequency of such transitions, preventing very short bursts which would be a type of overfitting, therefore making it easier to identify long bursts despite transient changes in the rate of the stream. s and g in the context of the Kleinberg algorithm represent the costs to control the burst detection based upon the onset of a burst and the length of it. After applying the burst detection algorithm,

the output stream consists of a sequence of the two values showing two states, *burst* and *baseline*. A burst at time step t indicates that the corresponding piece of information can be considered to be of popular interest at t and the baseline state indicates a reduction of popularity.

In a sequence void of bursts the message arrival would be evenly spaced so that the n messages over time T could be placed on the specified time domain according to 'gaps' $\hat{g} = \frac{T}{n}$. A consequence of a 'burst' is that greater activity in certain parts of the time will decrease the distance between these 'gaps' (separations between activity stamps). Regions with less activity would have $\hat{g}$ values that are larger due to the proportionally fewer activity stamps residing in that time domain. These rates can be indexed for the different regions with $a_i = \hat{g}^{-1}s^i$ and a base rate to compare with for the extent of the measurement of the deviation $a_0 = \hat{g}^{-1}$. A *cost* function is then defined for the rate transitions of region activity ratios, $\tau(i, j)$ with a state change cost to penalize the overfitting $(j - i)\gamma \ln n$ (where $\gamma > 0$). An automaton structure is then defined for the data $\mathbf{x}$ as gaps between activity time stamps for the q_i states to produce the overall cost function that is optimized for: $c(\mathbf{q}|\mathbf{x}) = (\sum_{t=0}^{n-1} \tau(i_t, i_{t+1})) + (\sum_{t=1}^{n} -\ln f_{i_t}(x_t))$. The cost sequence is optimized via the use of dynamic programming using a procedure similar to the forward backward algorithm within the hidden Markov model. The algorithm is used upon an aggregate of all the platform topic domains but applied to each platform independently. This allows the topic and platform specific analysis of the activity to be produced. Then for each platform domain topic the burst activity traces are created and a binary time series (square wave) is produced which is then the input for the LSTM units.

The output from the Kleinberg burst detection produces independent binary signals that are the input to the LSTM. Each social media platform is represented by a separate input variable; hence in our study, there are three input variables. The temporal difference for the training is set to a 10 day separation so the training and testing had a forward lag of the 10 day period. The testing looked at predicting the bursts of each platform separately and was CVE specific. The accuracy for the

topic domain burst predictions in subsequent time steps are noted and examined as the benchmark of success. Each LSTM layer has 100 hidden units, and there were 2 LSTM layers. There was a dropout layer for regularization between the LSTM layers.

Recurrent neural networks (RNNs) have become an increasing popular tool for sequence prediction, and LSTMs reside within this family of models. A strength of LSTMs for use in this context is their ability to address the *vanishing gradient* problem. It is of particular interest for this application since bursts do not typically follow specific temporal patterns and long delays between events makes parameter estimation a challenge. The hypothesis is tested within two cases; burst prediction with respect to CVE propagation within three social platforms, and community developer activity prediction characterized by a set of features, within three different topic domains in GitHub. The community level activity is only examined for GitHub which makes the investigation less ambiguous since there are repositories which attract a relatively large amount of attention with easily determined memberships. This is assessed through the number of commits to the code repository. Although there are cross repository overlaps, false negatives caused by temporary or missed membership are reduced.

Choosing the best architecture for the LSTM depends on several factors determined through empirical investigations. The proposed network architecture includes two stacked LSTM layers. The optimal number of layers can be different based on the use cases, however two layers have been shown to be generally enough to detect even more complex burst association features over time. Each LSTM layer has been equipped with a *Dropout* layer in which a random selection of outgoing edges are set to zero at each update of the training phase. Dropping some of the neurons during the training process is a regularization method which prevents overfitting [133]. A dropout layer also reduces the sensitivity to the specific weights of individual neurons by randomly excluding them from activation and weight updates when training the network. Loss and activation functions are chosen for each of the cross platform applications separately since the community level analysis in

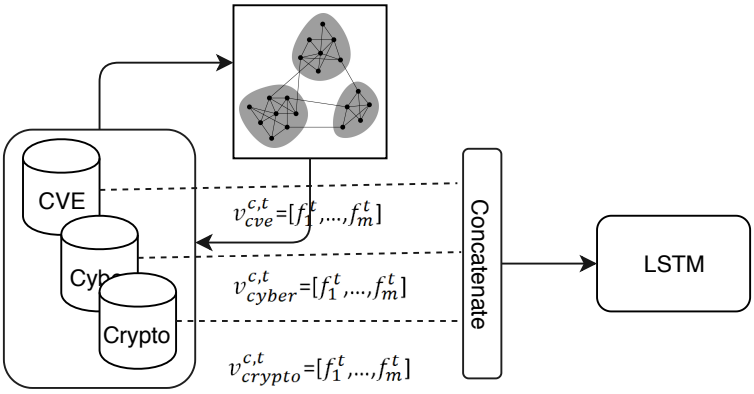


Figure 6.1: The methodological architecture for the investigation of the community responses in Github

GitHub deals with continuous data, and the topic domain analysis has a binary response variable. Non-community accuracy is measured using the binary cross entropy loss function. Community level loss is measured with mean squared error (MSE), since the target value ranges according to the subset of the communities that are activated over time. The rectified linear unit (ReLU) was chosen for the hidden layers. The adam and rmsprop optimization functions were chosen for the parameter training algorithm, and the sigmoid function was selected for the case of a binary classification problem.

The Keras framework was used to implement the model and it offers multiple accuracy functions with which to judge the model performance. The Root Mean Square Error (RMSE) option is an easy option to interpret as well as resulting in sufficiently accurate model performance for the community level. The overview of the process for GitHub community analysis is presented in Figure 6.1.

The data is stored as time indexed activity counts in a database, after the collection preprocessing has completed and the Kleinberg algorithm delivers the binary time series for each platform. These time stamped binary burst trajectories, for a specific social network platform s, are represented by

$\mathbf{x}_s = [x_{t=1,s}, x_{t=2,s}, \ldots, x_{T,s}]$ where $x_{t,s} \in \{0, 1\}$. The relationship which the research question seeks to establish is whether the subsequent future time step's bursts can be predicted using the current and past data, and the time into the future is ten days. $\mathbf{x}$ represents the data for the full set of platform trajectories. The model is fit with data that is a lag of 10 days from the designated 'current' time point t_c,

$$x_{t_c,s} = F(\mathbf{x}_{\mathbf{t}',s}) + \epsilon \tag{6.1}$$

where $\mathbf{t}'$ is $\mathbf{t}' = [t_{c-10}, t_{c-9}, \ldots, 1]$. The ϵ represents the error for the functional mapping expected for the fit to the dataset. This relationship is explored by employing an LSTM in which the parameterized model predicts this lagged response. The LSTM uses $\mathbf{x}$ that includes all of the platform trajectories and not a specific one. The conditional sequence trajectory into new predicted states for a specific platform (based upon a specific platform) is,

$$h(\mathbf{x}_{t_c,s}) = LSTM(h_{(\mathbf{x}_{t',s})}, \mathbf{x}_{t_c,s}), \tag{6.2}$$

and h is the hidden state of the LSTM unit. The *augmented* a_LSTM takes into account the multiple platforms as a model for predicting the bursts of a single specified platform,

$$h(\mathbf{x}_{t_c,s}) = a\_LSTM(h_{(\mathbf{x}_{t'})}, \mathbf{x}_{t_c}). \tag{6.3}$$

LSTM model vs. Markov chain

We compare the proposed LSTM against a Markov chain model. In contrast to the LSTM the Markov chain assumes a prefixed temporal independence structure which is typically first order. Although the current state could conceptually be able to define the probability for the state transitions, the issue of the *vanishing gradient* in temporal signals is a problem which the LSTM directly addresses in the model definition and framework.

In the burst application the target values are binary, and the benchmark model is compared against a stochastic Markov chain model between the 2 states (burst or no burst) over the timeline. The Markov state transition probabilities are calculated based on transition distributions using the empirical estimates in the training data. With the state burst and baseline, then the state transition matrix denoted by **B** is given by:

$$B = \begin{bmatrix} p_{l,l} & p_{l,h} \\ p_{h,l} & p_{h,h} \end{bmatrix}. \tag{6.4}$$

B describes the transition probabilities between the baseline state denoted as l and burst state denoted as h (for low and high). E.g. the low-low state $p_{l,h}$ represents the probability of not being in a burst state and transitioning into a burst during the next time step. These probabilities are calculated via:

$$p_{k_1,k_2} = \frac{\sum_{t=1}^{T-1} \mathbf{1}(x_{t+1} = k_1 \wedge x_t = k_2)}{T}. \tag{6.5}$$

where k is the burst state category and T the number of time points and $\mathbf{1}$ the indicator function. This is for the bursts which are considered to be endogenous and is irrespective of the origin of the content initiation. For the cross network burst association, the index for the social network platform index s is taken into account:

$$p_{k_1,k_2,s_1,s_2} = \frac{\sum_{t=1}^{T} \mathbf{1}(x_{t+1,s_2} = k_1 \wedge x_{t,s_1} = k_2)}{T}. \tag{6.6}$$

There is no restriction on the equality of the network index and $s_1 = s_2$ represents the endogenous signals for burst activity. The transition matrix B is then produced for each of these network pairings, B_{s_1,s_2} so that the independent associations can be examined for specific network burst alignments.

$$p_{k_1,k_2,s_1,*} = \frac{\sum_{t=1}^{T} \frac{\sum_{s_1=s}^{S} \mathbf{1}(x_{t+1,s_2}=k_1 \wedge x_{t,s_1}=k_2)}{S}}{T} \tag{6.7}$$

where S is the number of social media platforms that can exert cross-platform effects. The aim of the procedure is to estimate the probability of one platform affecting any other platform's activity. These empirical probabilities are considered to be the signals which are not exogenous to the data present in the database utilized. These probabilities $p_{k_1,k_2,s_1,*}$ will typically have smaller values than $\max_{s_2}(p_{k_1,k_2,s_1,s_2})$ as it is not expected that each platform will produce an equal contribution towards another platform's activity. The most typical use case is the probability entry of $\max_{s_2}(p_{h,h,s_1,s_2}), s_1 \neq s_2$ (probability of two bursts across different networks). The independent state transition sequence for the LSTM can be compared to this probability state transitions via the product of $\prod_t^T p_{k_1,k_2,s_1,s_2}$.

Results

The first application of the a_LSTM is that of burst prediction of a social network platform given the endogenous historical data for bursts and the external network bursts data. It is also compared to the predictive abilities of the LSTM which uses only endogenous data. Burst analysis and prediction is a fundamental problem in social network analysis since the patterns are typically non-linear and unpredictable. This is due to the fact that user activities have been shown to have an intrinsically bursty nature; yet bursts may also be a signal of topics that are of growing real-world interest. Social media bursts occur when a huge volume of data is exchanged during a short period of time, often due to the occurrence of an external event such as a sports game, election, or a movie release. Bursts can be indicative of both real news stories that are reaching peak public interest or fake news propagated by bad actors each which can have lasting effects on the future directions of society. Bursts of posts and tweets over platforms can be exploited by those wishing to push a disinformation campaign such as the artificial price inflation of a crypto currency. Therefore, analyzing bursts is a promising area to detect and counter such activities. We are particularly

interested in the propagation of information about software vulnerabilities across multiple social media platforms. When a vulnerability is discovered, different groups of actors such as software developers and hackers start posting announcements and discussion in widely used forums over multiple channels such as Twitter, GitHub, and Reddit. Previous work on this data has shown that CVEs exhibit cross platform correlations. [134] investigated 2600 CVEs across multiple platforms including Reddit, Twitter and GitHub. They found that around 24.7% CVE cascades start from GitHub and then jump into Reddit and Twitter respectively. 45.7% start from GitHub and then jump into Twitter and Reddit respectively, 15.5% start from Twitter and go through Reddit and then GitHub. 4.1% follow Twitter, GitHub, Reddit pattern; 7.8% Twitter, Reddit and GitHub pattern and the rest follow Reddit, GitHub and Twitter platform path pattern. As such and since bursts can be caused by exogenous phenomena and are indicative of burgeoning popularity, we hypothesize that leveraging cross platform social media data may be valuable for predicting bursts within a single social media platform.

Table 6.1 provides an overview of the dataset used. There are two main social network platforms summarized by the mean, median and standard deviation for the number of actors (nodes) and events (edges) produced. Although GitHub and Twitter provide services for different use cases, there are comparable numbers for each case—a surprising finding. We attribute this to Twitter allowing for a much lower time to entry for the production and dissemination of content on its platform. Table 6.2 shows the results of applying the a_LSTM model to burst prediction between networks in comparison to the MCM (Markovian based model) described in Section 6. The preprocessing stage using the Kleinberg algorithm is conducted with different parameter values to ensure that the predictive comparison is not dependent upon a specific choice of burst allocations chosen for the preprocessing. In all cases the a_LSTM outperforms the MCM for the different Kleinberg parameterizations.

Figure 6.2 shows two subfigures a) and b) which examine different CVEs, CVE-2017-5638 and

Table 6.1: Overview of the data used by the models in the top 20 CVEs in Github, Twitter and Reddit for the number of events created by the number of actors mentioned for them.

	GitHub		Twitter		Reddit	
	Actors	Events	Actors	Events	Actors	Events
Mean	223.7	487.52	199.05	261.55	14.5	19
Median	162.5	339	195	270	11	14.5
Std	224.36	507.24	134.58	187.79	9.7	12.15

Table 6.2: Model error (RMSE) over 20 CVEs streams on GitHub (s and g represents the hyper parameters used in Kleinberg algorithm for burst detection). The results show that the a_LSTM outperforms the MCM model irrespective of the burst detection parameterizations.

(s, g)	(2,0.50)		(2,0.75)		(3,0.50)		(3,0.75)	
Model	mean	std	mean	std	mean	std	mean	std
a_LSTM	0.09	0.20	0.11	0.22	0.08	0.20	0.09	0.21
MCM	0.36	0.25	0.29	0.18	0.37	0.21	0.39	0.28

CVE-2017-5638 respectively where the top plots are for Twitter and the bottom plots for GitHub. The activity is measured on the y-axis as the event count for each dataset and in the red rectangle outlines are the places where the Kleinberg algorithm detects the presence of a burst in at least one of the pairs of the time series for the predefined window widths. It can be seen how the algorithm does not produce sporadic placements of the burst detection and that the different platforms are not redundant as not all bursts correspond to a cross network burst. The region for the month of July for 2017 shows a lack of bursts of activity on both platforms. Our hypothesis is that 1) there is substantial alignment of the bursts 2) when there is more activity and bursts do not align it is due to a temporal shift manifesting itself as a 'lag' in the burst emergence. This further supports the motivation for the use of the a_LSTM.

Table 6.3 displays the results for the comparison of the a_LSTM and the LSTM which are each

Table 6.3: Community burstiness prediction results (applied to GitHub) where the a_LSTM (augmented multiple platform LSTM) outperforms the LSTM that uses only activity data from a single network to predict bursts of activity. The decrease in the RMSE error indicates that the augmented LSTM is a preferable choice.

Model	Optimizer	Activation function			
		Linear		Softmax	
		Mean	Std	Mean	Std
a_LSTM	adam	0.010	0.018	0.015	0.032
	rmsprop	0.021	0.41	0.010	0.016
LSTM	adam	0.022	0.28	0.025	0.041
	rmsprop	0.030	0.41	0.20	0.021

trained with 2 different parameter optimization schemes and 2 different activation functions. The target data used here is for the GitHub platform only, since the platform features make community topic identifications more distinct. It can be seen that the a_LSTM outperforms the LSTM on each combination of the optimization scheme and activation function listed. This shows not only the value in the a_LSTM but also that GitHub community activities are not isolated events but responsive to the activity which is being discussed in Reddit and Twitter as well.

Figure 6.3 shows the temporal patterns of various community activity domain topics in Github. The set of subplots shows different domain topics for which the content is affiliated; cyber security, crypto, or CVEs. There is a differentiation between the 'burstiness' of the activity which aggregates the total amount of contributions toward the platform activity and the 'users' trace which is for the number of users active over time separated windows. The vertical axis on each plot represents the degree to which the number of communities (connected components) in that topic domain can be considered active for that time frame. The a_LSTM model applied to this dataset incorporates the complete set of trajectories rather than predicting based upon a single trajectory and provides an improvement regardless of whether there is cross platform pattern association that can be inferred

from visual inspection. This study included 20 communities on GitHub within these domains. This activity is based upon *inter-community interactions* according to activity burstiness and the number of users who contribute to the community.

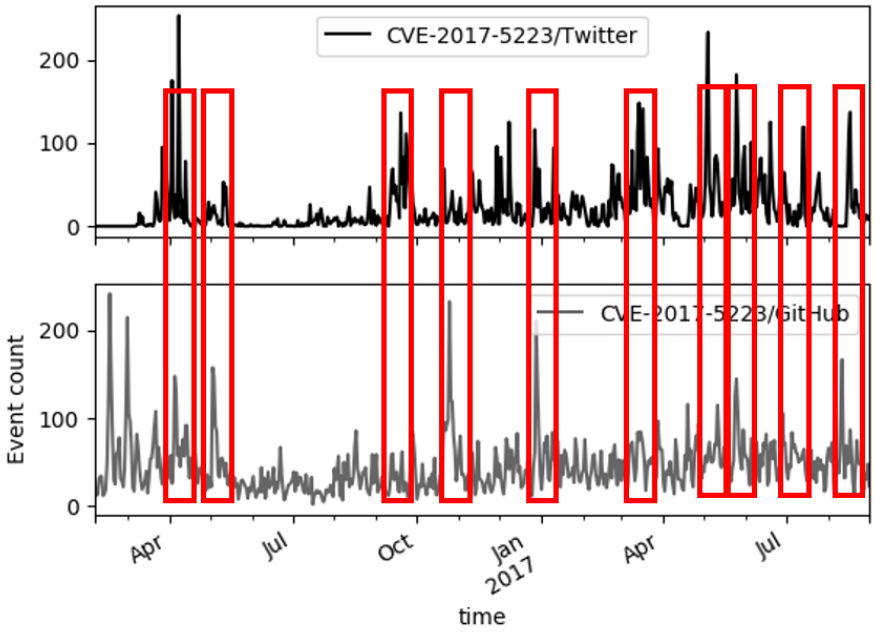
It can be seen that the bursts do not have a regular pattern in one platform or between the different platforms which are obvious and consistent over time. These networks as a whole are not completely governed by endogenous signals and at some time point are affected by exogenous phenomena which promote content creation on the social networks studied. Given that the time frame where relevant information for burst prediction cannot be inferred prior to the analysis, models requiring time lag parameterizations are challenging. The MCM model shows less optimal predictive power for this reason in comparison to the LSTM based models which use information from time dependencies of variable size. We believe that this is the reason that the single platform LSTM model outperforms the MCM. The proposed model, a_LSTM, augments the endogenous data with cross platform information that helps the model to more accurately detect bursts and community level activation.

Conclusion

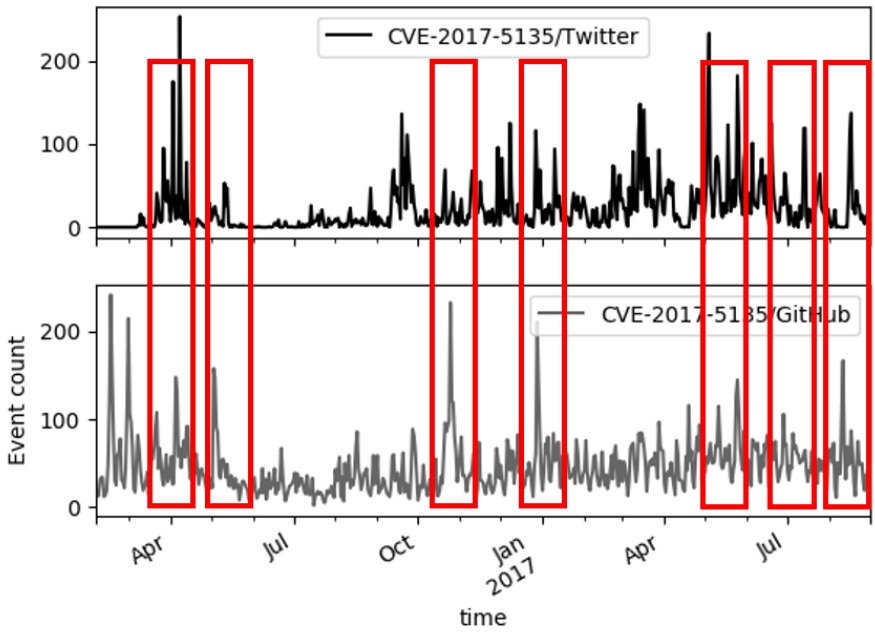
Online social networks display bursts of activity as supported by previous research on human dynamics. This phenomenon manifests itself as 'spikes' in the time series of activity traces that are aggregations of the count of time stamped events. These activity bursts are indicators that the participating users have an increased interest in the content being shared. This content can be related to civil unrest, financial disruptions and other newsworthy events. Our work concerns itself primarily with CVEs that incorporate cyber security, crypto currencies, and an aggregate of those topics. Predicting bursts related to cybersecurity activity may enable system administrators to anticipate and counter system level threats. A general algorithm for predicting social media bursts

is a stepping stone towards understanding how human societies transmit information. Previous research (e.g. [75]) has established how challenging the prediction is due to the lack of repetitive pattern. Here the LSTM is employed to model the burst prediction pattern. This work proposes the *a_LSTM* which considers data from multiple social networks in order to improve prediction performance. The dataset used to test this model is composed of events coming from Twitter, Reddit, and GitHub. The investigation also considers the related question of community topic domain activity where the connected components on a topic are counted as active or not over time providing a non-negative real number time series. This data is collected for GitHub before applying the LSTM and a_LSTM models. In both cases the a_LSTM shows better predictive accuracy. We conclude that the information across social networks can provide valuable information in predicting the bursts within a single network. This is a new finding that has not been discussed in related work, due to the relative shortage of cross-platform datasets.

Predictions of bursts can be improved upon using the proposed a_LSTM. Results are shown using data from three major social networks that have a global coverage. One avenue for future work would be to explore other topics, such as news or entertainment, and other social media platforms, such as YouTube and Instagram, to see if their behavior exhibits similar dynamics. The cross network association may not be exhibited across different forms of content or social media platforms, if the content sharing remains within the networks of the content origin. In this case the proposed a_LSTM model may not perform well. Future investigations could also explore how the data can be clustered into topic groups to see if they exhibit repetitive bursts similar to 'shock' trains. In our future work, we are exploring the use of LSTM event volume prediction for improving the performance of our agent-based simulation of social media platforms.



a)



b)

Figure 6.2: This figure displays the comparison of the event bursts for the activity streams of Twitter and GitHub for two different CVEs. The aligned time series show where both platforms exhibit bursts and where the preset time window exhibits no common burst activity.

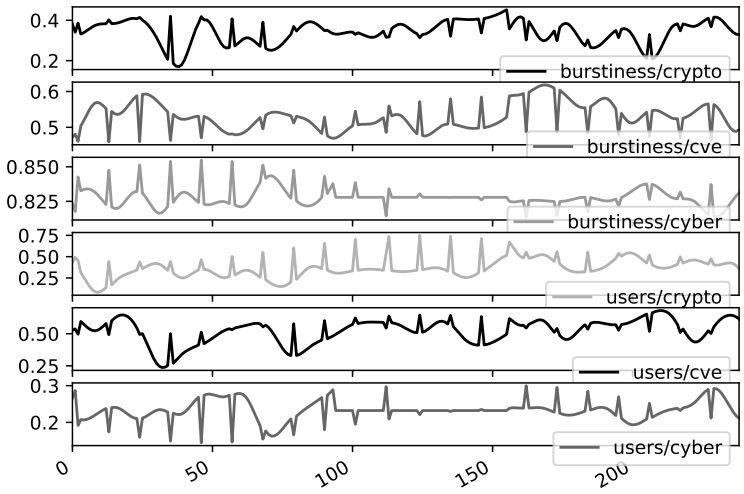


Figure 6.3: Temporal pattern of community burstiness and number of contributing users for the community surrounding Android within the Crypto, Cyber and CVE topics (in Github)

CHAPTER 7: LEVERAGING WEAK TIES USING SIMPLIFIED GRAPH CONVOLUTIONAL NEURAL NETWORKS

Introduction

In addition to providing entertainment and social engagement, social networks also serve the important function of rapidly disseminating scientific information to the research community. Social media platforms such as ResearchGate and Academia.edu help authors rapidly find related work and supplement standard library searches. Twitter not only serves as an important purveyor of standard news [135] but also disseminates specialty news in fields such as neuroradiology [136]. Venerable academic societies such as the Royal Society (@royalsociety) now have official Twitter accounts. Shuai et al. [137] discuss the role of Twitter mentions within the scientific community and the citations that create a topological arrangement between scientific publications. Given that scientific articles possess the potential to change the landscape of technology, it is important to understand the information transference properties of academic networks: can techniques originally developed for social networks yield insights about scientific networks as well?

Pagerank [138] produced a revolution in the ability to search through the myriad of webpages by examining the network structure for relevance. This concept has been applied to citation networks in academic literature, which conceptually has many overlaps with the interlinking of websites, and Ding et al. [139] apply the Pagerank algorithm to citations. The same first author extends this work to investigate endorsement in the network as well [140]. Endorsement as a process produces an effect that not only allows readers to navigate essential information, relevant overlaps or apply appropriate credit but also amplifies readership which is a dynamic seen frequently in online social processes. These links are direct links and also referred to as *strong-ties* [141].

Indirect or *weak-ties* have also been identified as important in social networks, most notably in Granovetter's seminal work: "The strength of weak ties" [86]. These indirect edges can be created through edges that span different communities (clusters of nodes) acting as 'bridges'. They can also be the result of 'triangulation' where 'friends-of-friends' produce a link due to the common connection they share. Figure 7.1 shows an example of a *weak-ties* connection between nodes A-C with a dotted edge representing connectivity due to a shared connection with node B. It can be said that A-C are connected from friends-of-friends, or triangulation creating a *weak-tie*. The work of [142] applies this concept to predicting edge production in a social network of professional profiles and shows that the explicit modeling of this triangulation dynamic leads to improved performance.

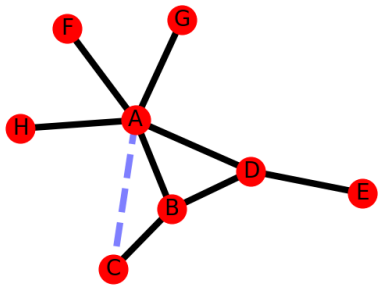


Figure 7.1: This figure illustrates how *weak-ties* can be produced through triangulation.

For our study on academic connections, we used two datasets, Cora [1] and Citeseer [3], which are discussed in more detail in Section 7. In addition to network structure, these datasets contain class labels, relating to the publication venues, which can be used to test machine learning prediction algorithms. In datasets that exhibit homophily, utilizing the features of nodes within a topological arrangement (an adjacency matrix) produces improved classification performance over an instance only framework. Our investigation focuses on whether the explicit addition of weak ties will assist

the inference process since they provide assistance in other network based processes. For instance, Roux et al. [143] investigate whether expertise within groups can cross the 'boundaries' from communities which are cohesive due to strong-tie connections. The work of [144] considers how differentiation of the edge types can improve the accuracy of social recommendations.

Determining the effect of interactions between nodes can be a time consuming process, which requires computational resources to analyze large networks. With the goal of understanding how node connections influence labels, various methodologies, such as [145], have been proposed where nodes iteratively propagate information throughout the network until convergence is achieved. A notable example of such is DeepWalk [146] which uses local information from truncated random walks to learn the latent variable representations. Relational neighbor classifiers such as Social Context Relational Neighbor (SCRN) have been shown to achieve good performance at inferring the labels of citation networks [145]. Graph Convolutional Networks (GCNs) [85] extended the methodology of Convolutional Neural Networks (CNNs) from images to graphs. GCNs, as CNNs, are constructed upon multiple layers of neural networks which makes them less amenable to interpretation [147, 148, 149].

This chapter uses the approach of the Simplified Graph Convolutional Neural Networks (SGC) [150] to investigate the importance of strong vs. weak ties. The methodological framework provides a reduction in the complexity of the model and computation time required. It has an intuitive manner of producing feature projections and generating the non-linearity for different classes. Even though it is a deep learning approach that accounts for multiple-layers, the simplification allows a single parameter matrix to be produced that can more easily be interpreted if desired. An SGC implementation can be found in the DGL library [151, 152].

The SGC uses the features of the nodes and the connectivity in the adjacency matrix to infer the class labels. In this chapter we augment this adjacency matrix so that *weak-ties* are included as

well. This produces a matrix in which the *strong-ties* and the *weak-ties* are treated equally at the start of the inference procedure. Results using this augmented adjacency matrix are compared to the label produced using the original adjacency matrix. Our experiments also consider the possibility of missing nodes. Obtaining complete datasets of networks is a challenge for a wide range of reasons; for instance, online platforms limit the API calls from developer accounts to reduce the website loads. It is then a crucial question as to whether the results of the investigation are sensitive to missing nodes [153]. Therefore, our experiments remove a range of pre-selected percentages of the network to compare the results. Nodes are ranked for removal based on three different centrality algorithms: betweenness, closeness, and VoteRank [2]. These algorithms sort the nodes in descending order and remove the top percentage chosen (e.g., 20%) so that the inference is performed without the influence of these nodes. We seek to answer the question: which gaps in the data are most likely to affect the SGC? The results help provide another piece of evidence towards the utility of *weak-ties* in sociological processes.

An added incentive for exploring the use of the SGC, is that it addresses an issue with the application of GCNs (graph convolutional neural networks), [85], where the increase in the number of layers beyond 2–3 can produce a degradation in the results. The number of layers employed by the GCN also corresponds to the K^{th} order neighborhood used in the SGC, and the results will be compared between both methodologies in Section 7. Although the application with GCNs [85] displays the degradation with an increase in the number of layers, L (corresponding to the K^{th} order neighborhood), the SGC does not display this degradation with an increase in K. The authors of [150] describe how the non-linearity for applications such as social networks may introduce unnecessary complexity.

The next section describes the datasets used in this study, and Section 7 outlines the methodology of the SGC. Then we present results on the effects of augmenting the adjacency matrix with weak ties and removing nodes ranked on a selection of centrality measures. Within the results section is

a subsection which compares the performance of the SGC with that of the GCN.

Data

Two datasets, Cora [1] and Citeseer [3], were used in this study. The Cora dataset is a citation network where the nodes refer to unique authors and the edges represent a weighted value for the mean citation relationship (from scientific publications). These scientific publications are classified and labeled into seven categories. The data were divided into a separate training and test set in order to provide consistent benchmarks between methodologies. The Citeseer dataset is another network dataset based upon citations where the nodes are also authors and edge values represent the mean citation relationship; it includes six different class labels. The SGC methodology described in Section 7 was used to infer the correct labels for a subset of the nodes in these datasets using both the original connectivity matrix and an augmented one. Table 7.1 provides an overview of some of the basic information of the datasets.

Figure 7.2 shows the degree distribution for the Cora dataset and how the distribution changes when different percentages of the network were removed. Those percentages of the network were removed according to different network centrality measures: betweenness in Figure 7.2a, closeness in Figure 7.2b, and VoteRank in Figure 7.2c. Figure 7.3 shows the same operation but using the Citeseer dataset. It is interesting to note how the Cora and Citeseer plots differ between their equivalent subfigures. The plots for the betweenness and the closeness change much more than VoteRank which provides evidence that it is more robust against choosing nodes with many edges as a measure of centrality in different datasets.

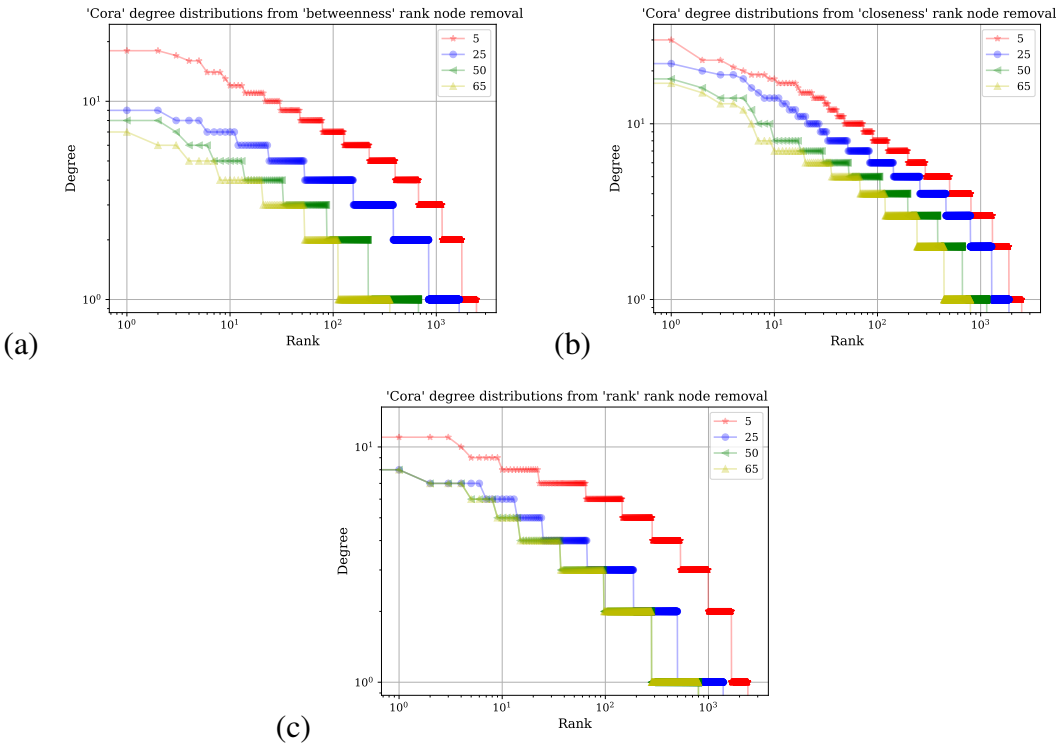


Figure 7.2: These plots show the degree distributions for the Cora network of publications [1] and how those distributions are altered when a certain percentage of the nodes are removed based upon a metric. Each subfigure shows the results of applying a different metric to sort and remove nodes: (**a**) node 'closeness'; (**b**) node 'betweenness'; (**c**) node 'VoteRank' [2].

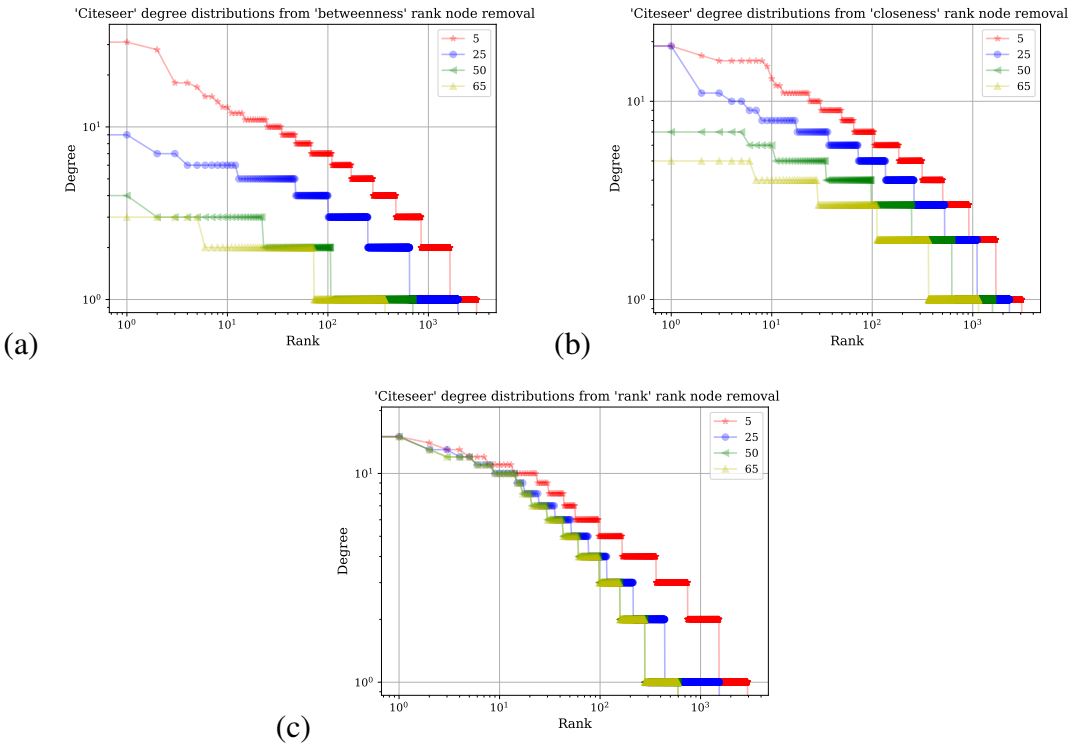


Figure 7.3: These plots show the degree distributions for the Citeseer network of publications [3] and how those distributions are altered when a certain percentage of the nodes are removed based upon a metric. Each subfigure shows the results of using a different metric to sort and remove nodes: (**a**) node 'closeness'; (**b**) node 'betweenness'; (**c**) node 'VoteRank' [2].

Table 7.1: Summary statistics of the networks from the datasets used in this study: Cora [1] and Citeseer [3]. Each of these datasets has a set of classes used to identify groups of publications in Cora as well as with Citeseer.

	Cora	**Citeseer**
# of Nodes	2708	3327
# of Edges	10,556	9228
# of Classes	7	6
Average degree	3.8981	2.8109
Density	0.00143	0.00084
Triadic closure	0.0934	0.13006

Methodology

For a graph $G = (V, \mathbf{A})$, $\mathcal{V}$ is the node within a set of N nodes $V = \{v_1, v_2, \ldots, v_N\}$, and the adjacency matrix is a symmetric matrix, $\mathbf{A} \in \mathbb{R}^{NxN}$. Each element of $\mathbf{A}$, $a_{i,j}$, holds the value of the weighted edge between two nodes v_i and v_j (an absence of an edge is represented by $a_{ij} = 0$). The degree matrix $\mathbf{D} = \text{diag}(d_1, d_2, \ldots, d_N)$ is a diagonal matrix of zero off diagonal entries and each diagonal entry is the row sum of the matrix $\mathbf{A}$; $d_i = \sum_j \mathbf{A}_{ij}$. There is a feature vector, $\mathbf{x}_i$, for each node i so that the set of features in the network of nodes is a $n \times d$ matrix, $\mathbf{X} \in \mathbb{R}^{n \times d}$ where d is the dimensionality of the feature vector. Each node is assigned a class label from the set of classes C; for each node we wish to utilize both $\mathbf{A}$ and $\mathbf{X}$ to infer $\mathbf{y}_i \in \{0, 1\}^C$. $\mathbf{y}_i \in \{0, 1\}^C$ is ideally a one-hot encoded vector which can be supplied data to assist the parameter estimations.

The normalized adjacency matrix with included self-loops is defined as,

$$\mathbf{S} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{7.1}$$

and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \text{diag}(\tilde{\mathbf{A}})$. The classifier employed by the SGC is:

$$\hat{Y} = \text{softmax}(\mathbf{S}^K\mathbf{X}\boldsymbol{\Theta}). \tag{7.2}$$

Here, the softmax can be replaced with σ as used in binary logistic regression when $C = 2$, and for the softmax on multiple categories we have $\text{softmax}(\mathbf{x}) = \exp(\mathbf{x})/\sum_{c=1}^{C}\exp(\mathbf{x}_c)$. The component $\boldsymbol{\Theta}$ is the matrix of parameter values for the projections of the feature vectors so that it is of dimensionality $d \times C$, $\boldsymbol{\Theta} \in \mathbb{R}^{d\times C}$. Intuitively this can be understood as the parameter matrix holding a single vector of parameters of length equal to that of the feature vector and as many of these vectors as there are class labels. This linearization derives from the general concept in deep learning for sequential affine transforms in layers which are subsequent stages,

$$\hat{Y} = \text{softmax}(\mathbf{S}\ldots\mathbf{S}\mathbf{S}\mathbf{X}\boldsymbol{\Theta}^{(1)}\boldsymbol{\Theta}^{(2)}\ldots\boldsymbol{\Theta}^{(K)}). \tag{7.3}$$

It can then be seen how the value of K chosen represents the number of layers in the network employed. More details can be found in [150] where the methodological derivation is elaborated upon. A key requirement in this framework is the setting of the parameter value k. This can be considered as a tuning parameter for varying of the number of propagation steps taken. This relates to the matrix powers of an adjacency matrix which produce in each entry the number of 'walks' between nodes [154, 155].

From the adjacency matrix the matrix including *weak-ties* produced through 'triangulation' ([142]) can be found via the walks of length two with $\mathbf{A}^2$. The original adjacency matrix is said to con-

tain the *strong-ties* and there is considerable sociological research into the value of each type of connectivity [86]. In this work we explore the use of an adjacency matrix which contains both the *strong-ties* and the *weak-ties* via;

$$\mathbf{A}' = \mathbf{A}^2 + \mathbf{A}. \tag{7.4}$$

Figure 7.4 demonstrates this, and it can be seen visually in the subfigures. Figure 7.4a shows a hypothetical network with 4 nodes connected in a chain and Figure 7.4b shows how those nodes are connected when $\mathbf{A}'$ is produced from including both the *strong-ties* and the *weak-ties*.

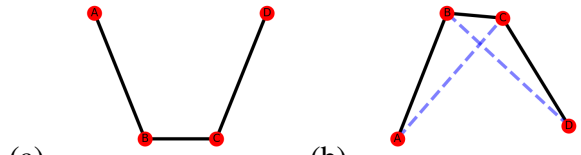
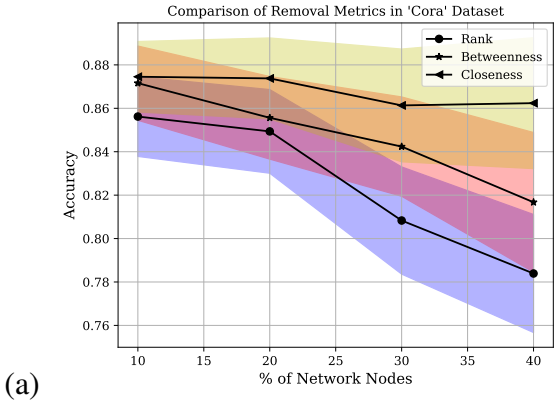
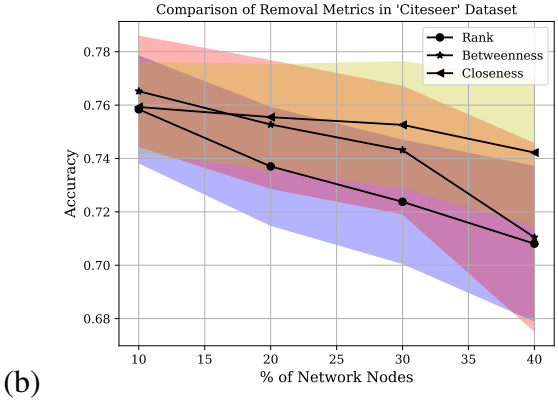


Figure 7.4: These show the concept of *weak-ties* produced through triangulation and how it can affect a small network: (**a**) shows a hypothetical network; (**b**) shows the result of introducing the *weak-ties* into the network as well as the original strong ties which were direct links.

Figure 7.5 shows a demonstration of the SGC methodology in its ability to accurately predict class labels on the Cora and Citeseer datasets. To explore how robust the methodology is, different percentages of the network were removed; nodes were selected for removal based on their rank calculated from different centrality measures: betweenness, closeness, and VoteRank. Each network measure expresses different aspects of a node's position in a network and therefore changes in the prediction accuracy, which assist in understanding empirically how node network placements contribute the most in correct label prediction. The VoteRank algorithm considers local node influences more than betweenness or closeness. Figure 7.5a shows results obtained from running the model on the Cora dataset, and the Citeseer results are shown in Figure 7.5b.



(a)



(b)

Figure 7.5: The Simplified Graph Convolutional Neural Networks (SGC) methodology was applied to predicting the test case labels when a certain percentage of the nodes were removed based upon the metrics closeness, betweenness, and VoteRank with the accuracy on the y-axis shown. These results shown in (**a**) and (**b**) are for the network datasets Citeseer and Cora.

Results

This section explores how the class label prediction accuracy is affected by different removal strategies when the connectivity matrix contains both the links for the strong ties and the weak ties. These results show how the parameter k can affect the accuracy of the prediction of class labels. Section 7 explores how the Simplified Graph Convolutional Neural Network (SGC) methodology performs on the datasets of Cora and Citeseer when different percentages of the nodes are removed.

The nodes are removed according to their rank in terms of network centrality positions: betweenness, closeness, and VoteRank. For example, if 20% are removed using closeness as a measure, the nodes were ordered according to the value of closeness from largest to smallest, and the top 20% of the nodes in that percentile of closeness are removed. The purpose of this manipulation is to explore how robust the methodology is to central node removals whose influence on class labels can extend beyond their immediate vicinity.

As shown in Figure 7.5, we explore how the accuracy is affected by the different network measures used to rank nodes for removal but with the modified adjacency matrix that defines the connectivity for each node. This modification incorporates direct edges (links) called 'strong ties'), as well as links between nodes that have a common friend. These newly introduced edges are the 'weak ties' that are a result of 'triangulation' as shown in Figure 7.4. The changes in the results due to the inclusion of the *weak-ties* can assist in establishing their importance in people's classification efforts in real life. A set of plots compare the accuracy of the SGC prediction of class labels with different network removal rankings given the addition of weak ties. The effect of the parameter value of k on accuracy is also explored to understand the sensitivity of the results to the only parameter that requires tuning in SGC.

Figure 7.6 shows the results of applying the SGC with different values of k for predicting class labels. On the horizontal axis is the value of k and on the vertical axis the accuracy as a percentage of the test class labels predicted correctly. The betweenness metric is used to rank the nodes and different percentages of the network's nodes are removed. The percentage values for each line are indicated in the legend. Figure 7.6a,b shows the results obtained from using the Cora and Citeseer datasets where the adjacency matrix used contains direct links between nodes and their *strong-ties* as well as their *weak-ties* as described in Section 7. Figure 7.6c,d shows the results when the original adjacency matrix containing only the *strong-ties* is used. For $k = 0$ similar results are obtained and for the final k value, $k = 7$, but the progression differs. The difference in

progression is evident for the Cora dataset at $k = 1$ and up to $k = 4$ where the predictive accuracy for Figure 7.6a is reduced. This also applies to the Citeseer dataset, and especially to the scenario where 20% or 30% of the nodes have been removed. When $k = 0$ the SGC operates effectively in a manner similar to logistic regression where the network information is not used and inference is conducted using only the features of the node in question. These results support the conclusion that the augmented network topology of the strong-ties and the weak-ties does not facilitate improved accuracy of label prediction.

Figure 7.7 also shows the results of applying the SGC with different values of k for predicting class labels. The value of k is shown on the horizontal axis and on the vertical axis the accuracy as a percentage of test class labels being predicted correctly. Here the closeness metric is used to rank the nodes for removal. The different percentages for the removal of network nodes for each line is shown in the plot legends. Figure 7.7a,b shows the results obtained from using the Cora and Citeseer datasets where the adjacency matrix used contains direct links between nodes and their immediate neighbors (*strong-ties*) as well as their *weak-ties* (edges obtained via triangulation) as described in Section 7. Figure 7.7c,d shows the results when the original adjacency matrix containing only the *strong-ties* is used. For $k = 0$ similar results are obtained between the different pairs as the connectivity of the adjacency is not incorporated and node inference looks only at the features obtained from the node of concern. For $k = 7$ similar values are obtained through the extended radius of the adjacency power, but the progression of the trace differs between pairs of the plots. The difference between the pairs of traces can be easily seen by inspection of the application to the Cora dataset at values $k = 1$ and up to $k = 4$ where the predictive accuracy for Figure 7.7a is reduced. This also applies to the Citeseer dataset, and is attenuated when 20% or 30% of the nodes have been removed. These results also support the conclusion that the augmented network topology of the strong-ties and the weak-ties does not facilitate improved accuracy of label prediction and that these conclusions are robust according to removal with a different network centrality ranking.

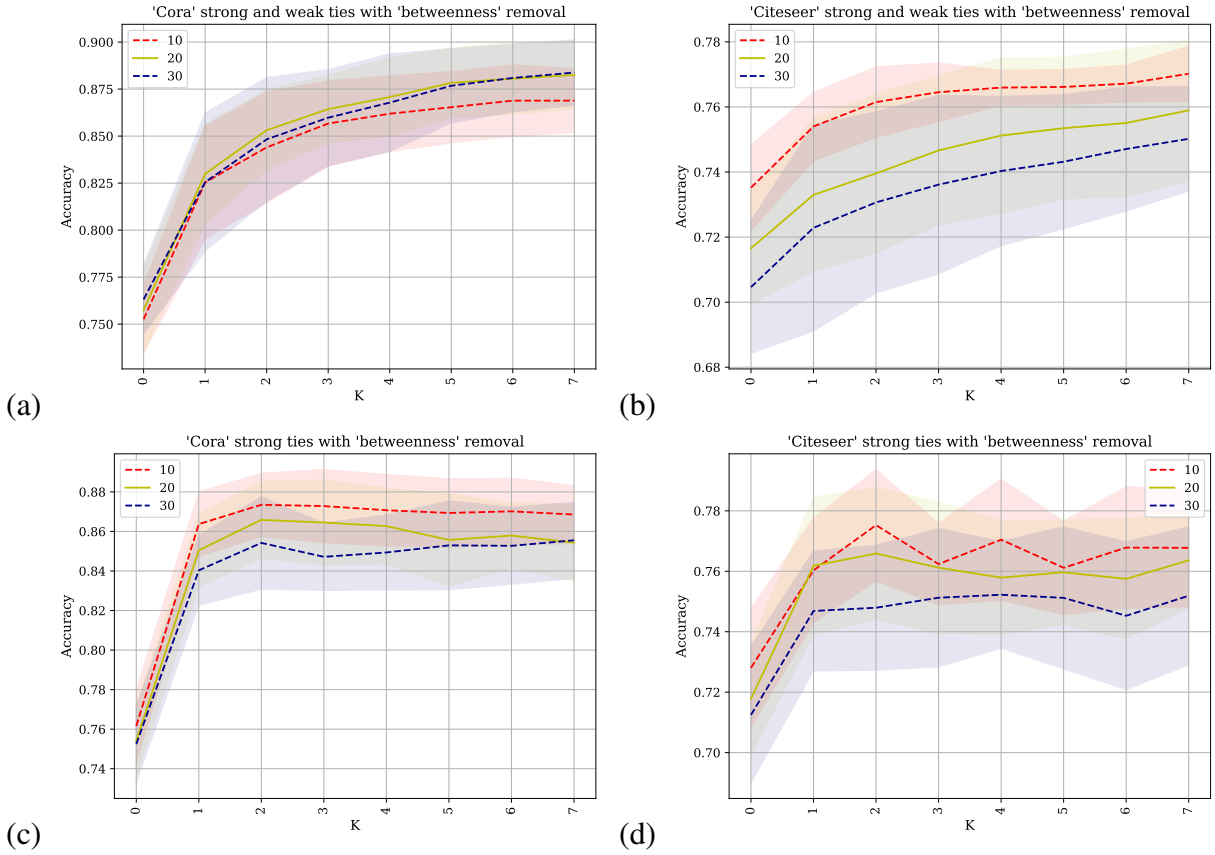


Figure 7.6: The SGC methodology was applied to predicting the class labels of the datasets Cora and Citeseer where the accuracy was plotted against the parameter k. The betweenness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show how the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

Figure 7.8 also shows the results of applying the SGC with different values of k for predicting class labels but uses the VoteRank centrality metric to rank the nodes for removal. The different percentages of node removal for each line are shown in the plot legends. Figure 7.8a,b shows the results obtained from the Cora and Citeseer datasets where the adjacency matrix used contains direct links between nodes and their immediate neighbors (*strong-ties*) as well as their *weak-ties* (edges obtained via triangulation) as described in Section 7. Figure 7.8c,d shows the results when

the original adjacency matrix containing only the *strong-ties* is used. When $k = 0$ similar results are obtained between the different pairs as the connectivity of the adjacency is not incorporated and node inference looks only at the features from the node of concern. The application of VoteRank changes the interpretation of the previous results where both applications to Cora and Citeseer have improved results for the augmented adjacency matrix (*strong-ties* and *weak-ties*) from $k = 3$ and upwards.

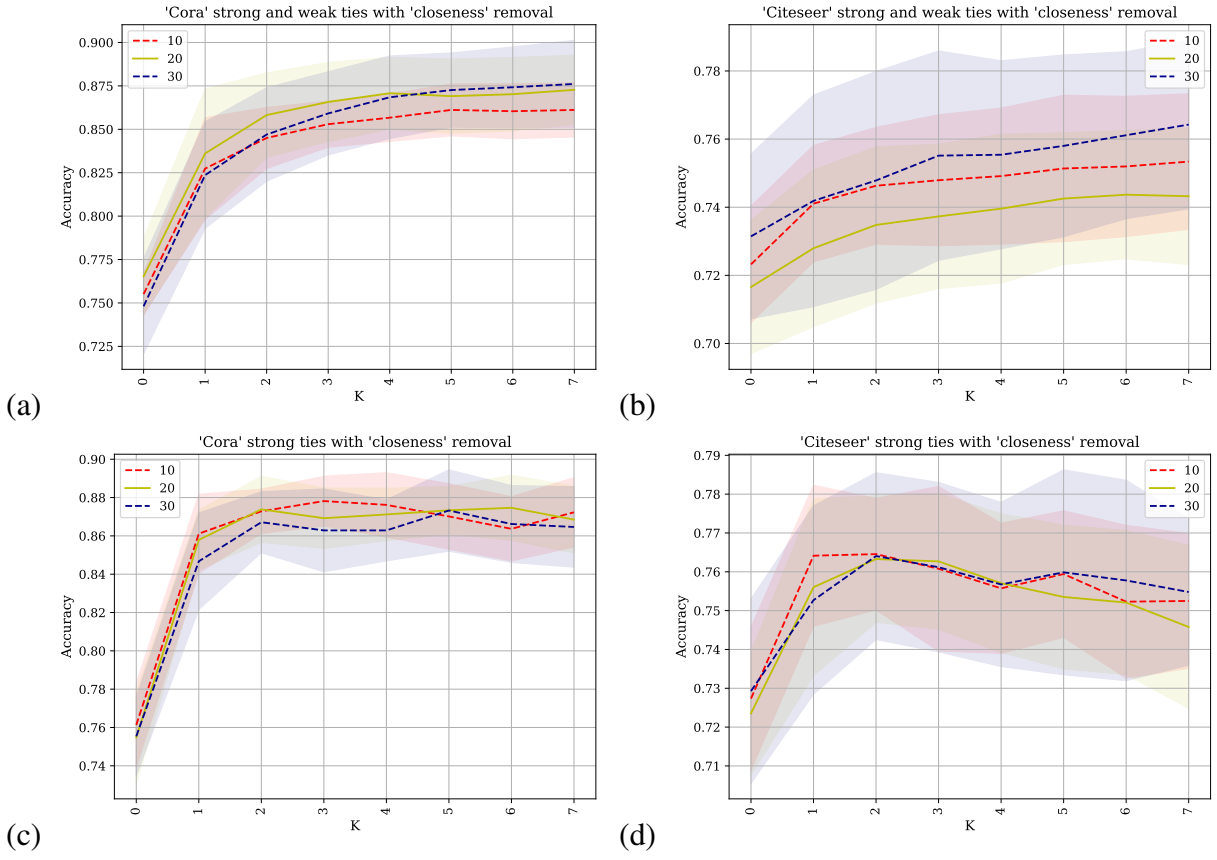


Figure 7.7: The SGC methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the accuracy was plotted against the parameter k. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

The set of results show that for $k < 3$ the adjacency matrix containing the set of original *strong-ties* edges suffices to produce the best results. For larger values of k the augmented adjacency, which contains both the *strong-ties* and the *weak-ties*, can show improved performance when nodes are removed according to the VoteRank algorithm and not according to betweenness or closeness. This emphasizes that there is a complex interplay between how node centrality is measured and the manner in which the inference methodology operates. It cannot be considered an *a priori* principle that *weak-ties* can provide an increased predictive power due to its support from the social science domain and its adherence to it. On the contrary, they induce a requirement for larger values of k to reach the maximum accuracy implying that the SGC requires more 'layers' which effectively aggregates information from more distant nodes in order to counter balance the introduction of *weak-ties* as *strong-ties*. This can provide anecdotal evidence that those two types of edges may require separate treatment. Further experiments conducted, working with a starting network of only the *weak-ties*, produced networks with an increased number of disconnected components.

These results also support the claims of the authors of 'VoteRank' when they state that the methodology identifies a set of decentralized spreaders as opposed to focusing on a group of spreaders which overlap in their sphere of influence. This is why the VoteRank targeted node removal was more effective in reducing the accurate label inference since more locally influential nodes for classification were identified; the *weak-ties* provided extra information about local labels in the absence of these essential *strong-tie* connected nodes.

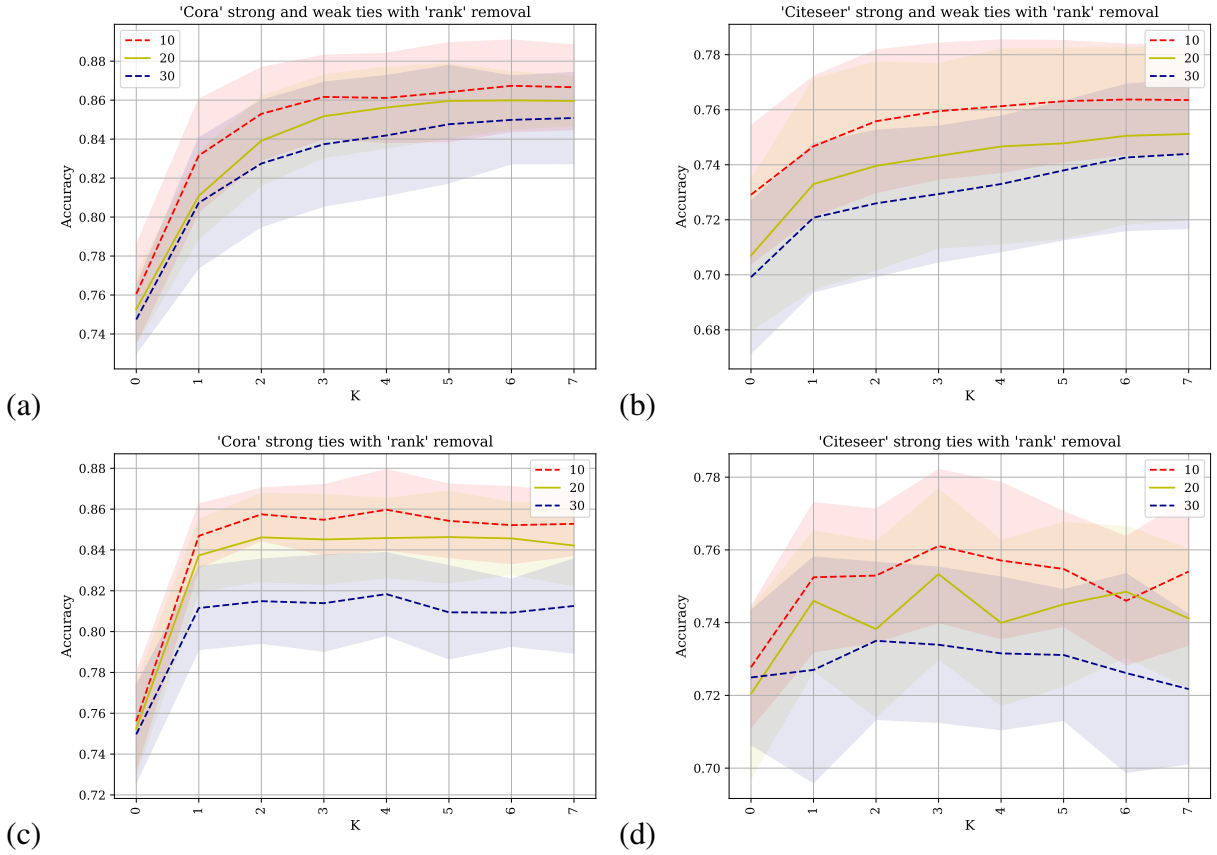


Figure 7.8: The SGC methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the accuracy is plotted against the parameter k. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

Comparison to GCN

This section compares results from applying SGC [150] vs. using the original GCN framework [85]. [85] discusses the effect of adding more network layers on accuracy. It states that the best choice is 2–3 layers and that after 7 layers there is steep degradation of accuracy. The number of layers corresponds to the number of 'K' hops as explored with the SGC previously. The SGC method-

ology encapsulates the K hop neighborhood without the non-linearity and therefore avoids the degradation of accuracy with increased K or L. Figures A.16–A.20 present the results of applying the GCN in the same set of situations that we evaluated with SGC. The number of layers L is on the x-axis (corresponding analogously to the K in SGC) and the y-axis is the accuracy. In each of these figures, the Cora and Citeseer datasets are used when examining the strong with weak ties in an augmented network as well as using the original network containing only the strong ties. Each figure removes a percentage of the nodes based upon the rank of the nodes with the centrality measures of betweenness, closeness, and VoteRank respectively. The plots have three lines per plot where there are different percentages (10%, 20%, and 30%) of the nodes removed based upon the centrality measure. In each of the Figures A.16–A.20 it can be seen how the accuracy degrades after $L = 1$ showing how the SGC is able to include more network information about each node without introducing unnecessary complexity which degrades accuracy. The degradation of the accuracy in relation to the choice of centrality measure is comparable between the results, showing that the GCN is less specific to the node network positions than the SGC is, which can be attributed to the non-linearity the GCN introduces via the layers. More plots with additional metrics to the accuracy -F1 micro, F1 macro, precision macro and precision micro- are provided in the Appendix.

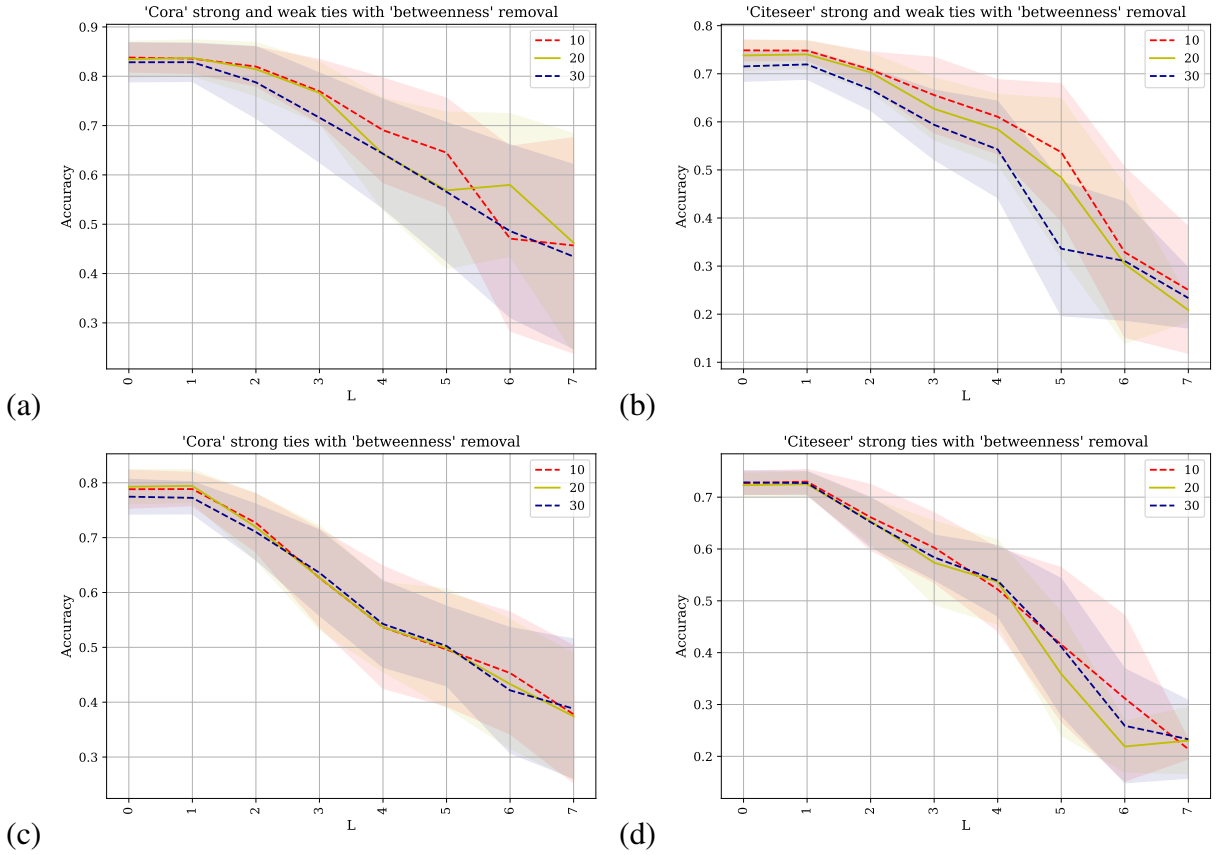


Figure 7.9: The GCN methodology was applied to predicting the class labels of the datasets Cora and Citeseer where the accuracy is plotted against the parameter L. The betweenness metric was used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

In 7, an additional set of tables are provided in order to see the comparison between the effectiveness of the SGC and the GCN over the values of K and L respectively. The two datasets and the centrality measures with the different edge sets are examined.

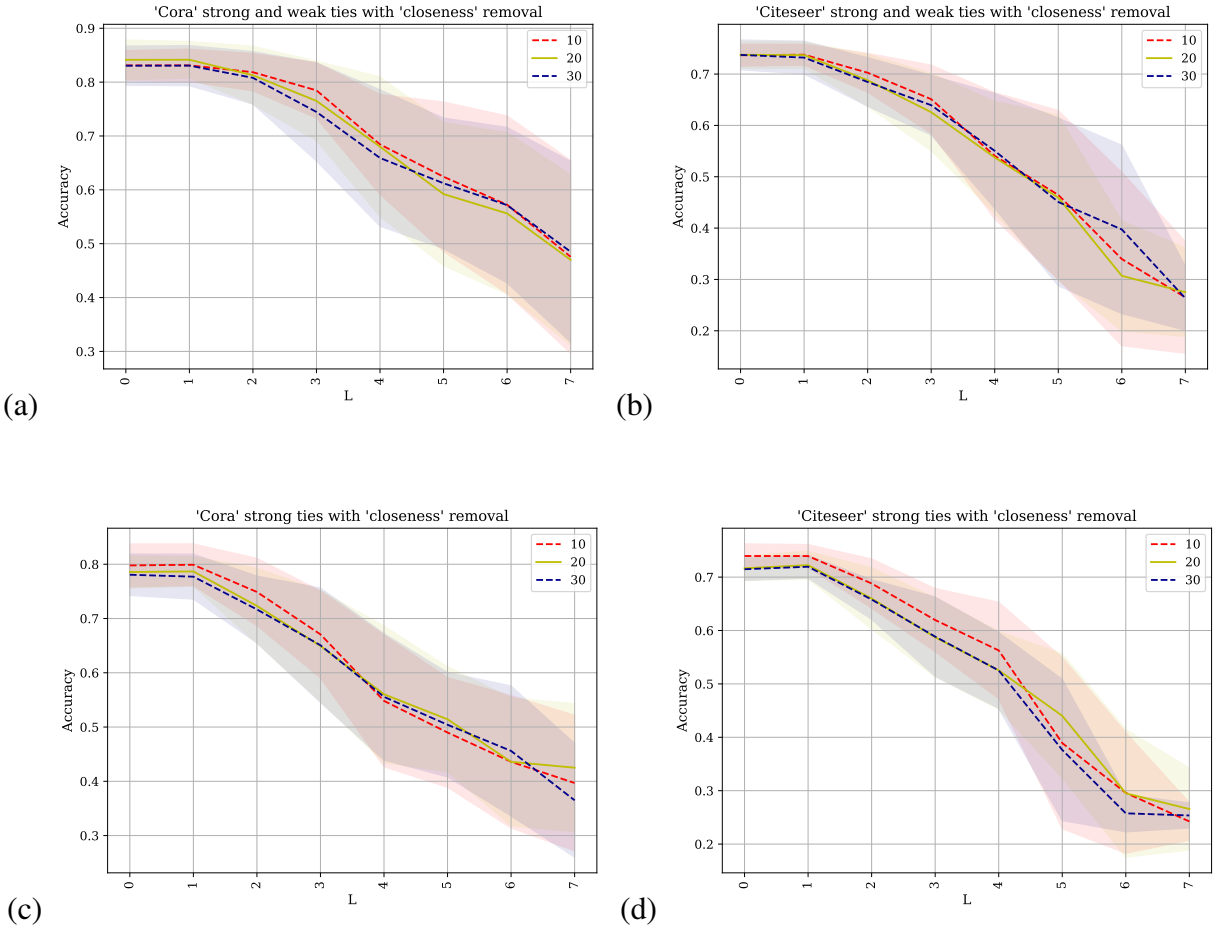


Figure 7.10: The Graph Convolutional Networks (GCNs) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the accuracy is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

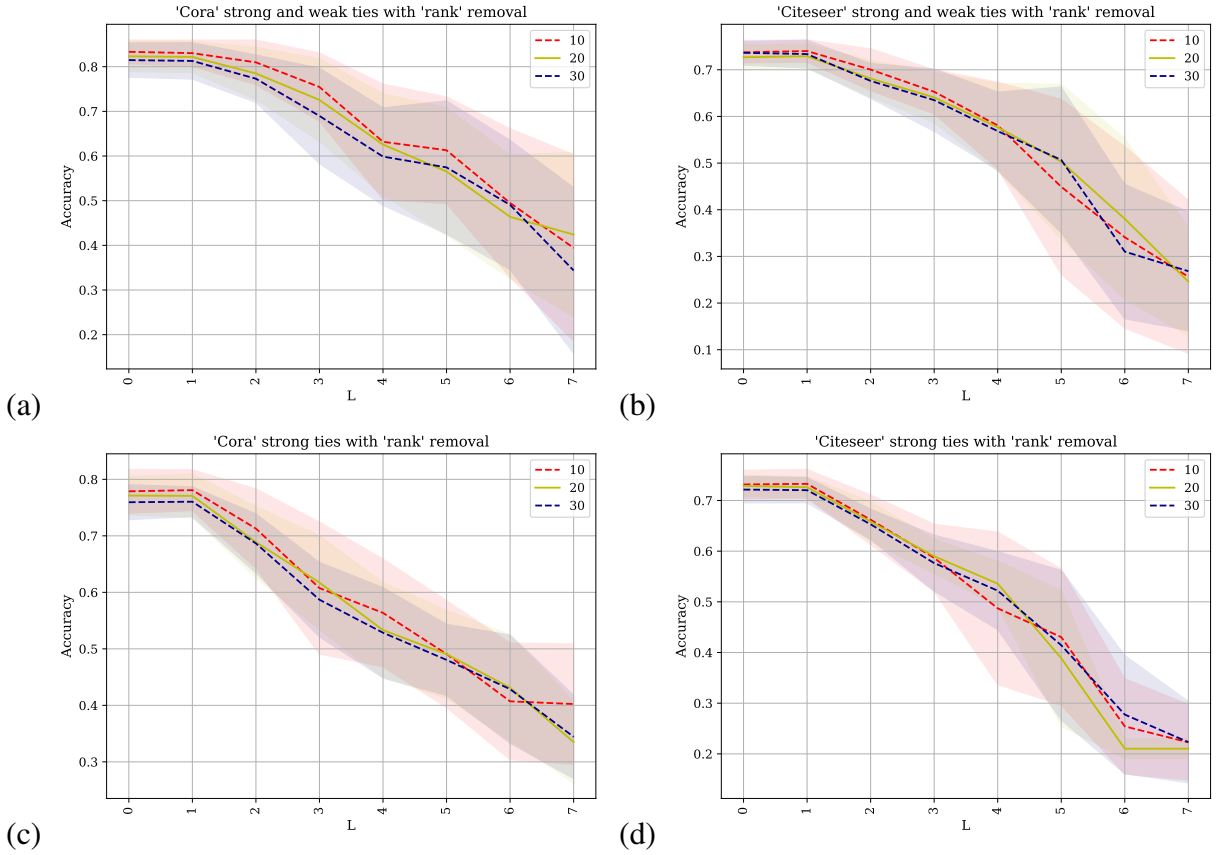


Figure 7.11: The GCN methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the accuracy is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

Comparison of Results between the SGC and GCN

The following tables present the comparison of the results between the use of the SGC and the GCN (shown separately in the Results section). Each table lists the centrality metric used to remove nodes based upon its ranking: betweenness, closeness and VoteRank. The column 'P' identifies the percentage of the network nodes removed based upon that centrality metric. The column 'L'

represents the number of layers used by the GCN and the column 'K' is for the exponent of the normalized adjacency matrix in the SGC. Under the columns 'GCN' and 'SGC' which refer to the graph convolutional neural network and simplified graph convolutional neural network respectively are the columns 'S' for the strong-ties used and 'SW' for the strong-ties and weak-ties aggregated. In Tables 7.2–7.7, the cell entries show the accuracy; each centrality measure is reported for the two datasets Cora and Citeseer.

Table 7.2: The GCN and SGC methodologies were applied to predicting the class labels of the Cora dataset. The betweenness metric is used to rank and remove different percentages of the network. S stands for when the network has the initial strong connections only and SW represents when the network is augmented with weak ties alongside the strong ties. L and K denotes the number of layers and the power in GCN and SGC framework respectively.

Metric	P	GCN					SGC			
		L	S		SW		K	S		SW	
			Mean	std	Mean	std		Mean	std	Mean	std
Betweenness	10	0	0.79	0.03	0.84	0.03	0	0.76	0.02	0.74	0.01
		1	0.79	0.03	0.84	0.03	1	0.86	0.02	0.75	0.01
		2	0.73	0.05	0.82	0.04	2	0.88	0.01	0.76	0.01
		3	0.63	0.08	0.77	0.06	3	0.87	0.02	0.76	0.01
		4	0.54	0.08	0.69	0.11	4	0.87	0.02	0.77	0.01
		5	0.50	0.07	0.65	0.11	5	0.87	0.02	0.77	0.01
		6	0.45	0.11	0.47	0.19	6	0.87	0.02	0.77	0.01
		7	0.38	0.13	0.46	0.22	7	0.87	0.01	0.77	0.01
	20	0	0.79	0.03	0.83	0.04	0	0.76	0.02	0.72	0.02
		1	0.79	0.03	0.84	0.04	1	0.85	0.02	0.73	0.02
		2	0.72	0.06	0.81	0.05	2	0.86	0.02	0.74	0.02
		3	0.63	0.09	0.77	0.06	3	0.86	0.02	0.75	0.02
		4	0.54	0.08	0.64	0.11	4	0.86	0.02	0.75	0.02
		5	0.50	0.11	0.57	0.16	5	0.86	0.02	0.75	0.02
		6	0.43	0.12	0.58	0.14	6	0.86	0.02	0.76	0.02
		7	0.37	0.11	0.46	0.22	7	0.85	0.02	0.76	0.02
	30	0	0.77	0.03	0.83	0.04	0	0.76	0.02	0.70	0.02
		1	0.77	0.03	0.83	0.04	1	0.85	0.02	0.72	0.03
		2	0.71	0.05	0.79	0.07	2	0.85	0.02	0.73	0.03
		3	0.64	0.09	0.72	0.09	3	0.85	0.02	0.74	0.03
		4	0.54	0.11	0.64	0.11	4	0.85	0.02	0.74	0.02
		5	0.50	0.10	0.57	0.14	5	0.85	0.02	0.74	0.02
		6	0.42	0.11	0.49	0.17	6	0.85	0.02	0.75	0.02
		7	0.39	0.12	0.43	0.19	7	0.86	0.02	0.75	0.02

Table 7.3: The GCN and SGC methodologies were applied to predicting the class labels of the Cora dataset. The closeness metric is used to rank and remove different percentages of the network. S stands for when the network has the initial strong connections only and SW represents when network is augmented with weak ties alongside the strong ties. L and K denotes the number of layers and the power in GCN and SGC framework respectively.

Metric	P	GCN					SGC				
		L	S		SW		K	S		SW	
			Mean	std	Mean	std		Mean	std	Mean	std
Closeness	10	0	0.80	0.04	0.83	0.03	0	0.77	0.02	0.76	0.01
		1	0.80	0.04	0.83	0.03	1	0.85	0.02	0.83	0.03
		2	0.75	0.06	0.82	0.03	2	0.87	0.01	0.85	0.02
		3	0.67	0.08	0.78	0.05	3	0.87	0.01	0.85	0.01
		4	0.55	0.12	0.68	0.09	4	0.87	0.02	0.86	0.01
		5	0.49	0.10	0.62	0.14	5	0.88	0.01	0.86	0.01
		6	0.44	0.12	0.57	0.16	6	0.86	0.02	0.86	0.02
		7	0.40	0.12	0.48	0.18	7	0.87	0.02	0.86	0.02
	20	0	0.79	0.03	0.84	0.04	0	0.76	0.02	0.77	0.02
		1	0.79	0.03	0.84	0.03	1	0.86	0.01	0.84	0.04
		2	0.72	0.07	0.81	0.05	2	0.88	0.02	0.86	0.02
		3	0.65	0.10	0.77	0.07	3	0.87	0.02	0.87	0.02
		4	0.56	0.13	0.68	0.13	4	0.88	0.02	0.87	0.02
		5	0.51	0.10	0.59	0.13	5	0.87	0.01	0.87	0.02
		6	0.44	0.12	0.56	0.15	6	0.87	0.02	0.87	0.02
		7	0.43	0.12	0.47	0.16	7	0.87	0.02	0.87	0.02
	30	0	0.78	0.04	0.83	0.04	0	0.75	0.02	0.75	0.03
		1	0.78	0.04	0.83	0.04	1	0.86	0.02	0.82	0.03
		2	0.72	0.06	0.81	0.05	2	0.87	0.02	0.85	0.03
		3	0.65	0.11	0.74	0.09	3	0.87	0.02	0.86	0.02
		4	0.56	0.12	0.66	0.13	4	0.87	0.03	0.87	0.02
		5	0.50	0.10	0.61	0.12	5	0.87	0.03	0.87	0.02
		6	0.46	0.12	0.57	0.14	6	0.87	0.02	0.87	0.02
		7	0.36	0.10	0.49	0.17	7	0.87	0.02	0.88	0.03

Table 7.4: The GCN and SGC methodologies were applied to predicting the class labels of the Cora dataset. The VoteRank metric is used to rank and remove different percentages of the network. S stands for when the network has the initial strong connections only and SW represents when network is augmented with weak ties alongside the strong ties. L and K denotes the number of layers and the power in GCN and SGC framework respectively.

Metric	P	GCN					SGC				
		L	S		SW		K	S		SW	
			Mean	std	Mean	std		Mean	std	Mean	std
Vote Rank	10	0	0.78	0.04	0.83	0.03	0	0.75	0.02	0.76	0.02
		1	0.78	0.04	0.83	0.03	1	0.86	0.01	0.83	0.03
		2	0.71	0.07	0.81	0.05	2	0.87	0.02	0.85	0.02
		3	0.61	0.12	0.75	0.08	3	0.87	0.02	0.86	0.02
		4	0.56	0.10	0.63	0.13	4	0.87	0.02	0.86	0.02
		5	0.49	0.09	0.61	0.12	5	0.87	0.01	0.86	0.03
		6	0.41	0.10	0.49	0.17	6	0.87	0.02	0.87	0.02
		7	0.40	0.11	0.39	0.21	7	0.87	0.02	0.87	0.02
	20	0	0.77	0.03	0.82	0.03	0	0.75	0.02	0.75	0.02
		1	0.77	0.04	0.82	0.04	1	0.85	0.02	0.81	0.02
		2	0.69	0.06	0.78	0.06	2	0.86	0.02	0.84	0.02
		3	0.62	0.08	0.73	0.09	3	0.86	0.02	0.85	0.02
		4	0.53	0.08	0.63	0.11	4	0.86	0.03	0.86	0.02
		5	0.49	0.08	0.57	0.14	5	0.85	0.03	0.86	0.02
		6	0.43	0.09	0.46	0.14	6	0.85	0.02	0.86	0.01
		7	0.34	0.07	0.42	0.18	7	0.85	0.02	0.86	0.01
	30	0	0.76	0.03	0.81	0.04	0	0.76	0.02	0.75	0.02
		1	0.76	0.03	0.81	0.04	1	0.84	0.02	0.81	0.03
		2	0.69	0.05	0.77	0.05	2	0.85	0.01	0.83	0.03
		3	0.59	0.07	0.69	0.11	3	0.84	0.02	0.84	0.03
		4	0.53	0.08	0.60	0.11	4	0.85	0.02	0.84	0.03
		5	0.48	0.06	0.57	0.15	5	0.85	0.02	0.85	0.03
		6	0.43	0.10	0.49	0.14	6	0.85	0.02	0.85	0.02
		7	0.34	0.07	0.34	0.19	7	0.84	0.01	0.85	0.02

Table 7.5: The GCN and SGC methodologies were applied to predicting the class labels of the Citeseer dataset. The betweenness metric is used to rank and remove different percentages of the network. S stands for when the network has the initial strong connections only and SW represents when network is augmented with weak ties alongside the strong ties. L and K denotes the number of layers and the power in GCN and SGC framework respectively.

Metric	P	GCN					SGC				
		L	S		SW		K	S		SW	
			Mean	std	Mean	std		Mean	std	Mean	std
Betweenness	10	0	0.73	0.02	0.75	0.02	0	0.72	0.02	0.74	0.01
		1	0.73	0.02	0.75	0.02	1	0.76	0.02	0.75	0.01
		2	0.66	0.06	0.71	0.04	2	0.78	0.02	0.76	0.01
		3	0.60	0.07	0.66	0.08	3	0.76	0.01	0.76	0.01
		4	0.52	0.08	0.61	0.08	4	0.77	0.01	0.77	0.01
		5	0.42	0.15	0.54	0.14	5	0.77	0.02	0.77	0.01
		6	0.31	0.16	0.33	0.18	6	0.76	0.02	0.77	0.01
		7	0.21	0.02	0.25	0.13	7	0.77	0.02	0.77	0.01
	20	0	0.72	0.02	0.74	0.03	0	0.71	0.01	0.72	0.02
		1	0.73	0.02	0.74	0.03	1	0.76	0.02	0.73	0.02
		2	0.65	0.04	0.70	0.04	2	0.76	0.02	0.74	0.02
		3	0.57	0.08	0.63	0.06	3	0.77	0.02	0.75	0.02
		4	0.54	0.08	0.58	0.07	4	0.75	0.02	0.75	0.02
		5	0.36	0.12	0.48	0.16	5	0.76	0.02	0.75	0.02
		6	0.22	0.05	0.30	0.16	6	0.76	0.02	0.76	0.02
		7	0.23	0.06	0.21	0.02	7	0.77	0.02	0.76	0.02
	30	0	0.73	0.02	0.72	0.03	0	0.71	0.02	0.70	0.02
		1	0.73	0.02	0.72	0.03	1	0.74	0.02	0.72	0.03
		2	0.65	0.05	0.67	0.04	2	0.75	0.02	0.73	0.03
		3	0.58	0.04	0.59	0.07	3	0.75	0.02	0.74	0.03
		4	0.54	0.07	0.54	0.10	4	0.75	0.01	0.74	0.02
		5	0.41	0.13	0.34	0.14	5	0.76	0.02	0.74	0.02
		6	0.26	0.11	0.31	0.12	6	0.74	0.03	0.75	0.02
		7	0.23	0.08	0.23	0.06	7	0.75	0.02	0.75	0.02

Table 7.6: The GCN and SGC methodologies were applied to predicting the class labels of the Citeseer dataset. The closeness metric is used to rank and remove different percentages of the network. S stands for when the network has the initial strong connections only and SW represents when network is augmented with weak ties alongside the strong ties. L and K denotes the number of layers and the power in GCN and SGC framework respectively.

Metric	P	GCN					SGC				
		L	S		SW		K	S		SW	
			Mean	std	Mean	std		Mean	std	Mean	std
Closeness	10	0	0.74	0.02	0.74	0.02	0	0.72	0.02	0.72	0.02
		1	0.74	0.02	0.74	0.02	1	0.76	0.02	0.74	0.02
		2	0.69	0.05	0.70	0.04	2	0.76	0.02	0.75	0.02
		3	0.62	0.06	0.65	0.07	3	0.76	0.01	0.75	0.02
		4	0.56	0.09	0.54	0.12	4	0.76	0.01	0.75	0.02
		5	0.39	0.16	0.46	0.16	5	0.76	0.02	0.75	0.02
		6	0.30	0.11	0.34	0.17	6	0.74	0.02	0.75	0.02
		7	0.24	0.04	0.27	0.11	7	0.74	0.02	0.75	0.02
	20	0	0.72	0.02	0.74	0.03	0	0.72	0.02	0.72	0.02
		1	0.72	0.03	0.74	0.03	1	0.76	0.03	0.73	0.02
		2	0.66	0.06	0.69	0.05	2	0.76	0.01	0.73	0.02
		3	0.59	0.07	0.63	0.08	3	0.77	0.02	0.74	0.02
		4	0.53	0.07	0.54	0.11	4	0.75	0.02	0.74	0.02
		5	0.44	0.12	0.46	0.16	5	0.75	0.01	0.74	0.02
		6	0.30	0.12	0.31	0.11	6	0.74	0.02	0.74	0.02
		7	0.27	0.08	0.28	0.09	7	0.74	0.03	0.74	0.02
	30	0	0.71	0.02	0.74	0.03	0	0.71	0.03	0.73	0.02
		1	0.72	0.02	0.73	0.03	1	0.75	0.02	0.74	0.03
		2	0.66	0.04	0.68	0.05	2	0.76	0.03	0.75	0.03
		3	0.59	0.07	0.64	0.06	3	0.77	0.01	0.76	0.03
		4	0.53	0.07	0.55	0.11	4	0.76	0.03	0.76	0.03
		5	0.38	0.13	0.45	0.16	5	0.75	0.02	0.76	0.03
		6	0.26	0.03	0.40	0.16	6	0.76	0.02	0.76	0.02
		7	0.25	0.02	0.26	0.06	7	0.76	0.02	0.76	0.02

Table 7.7: The GCN and SGC methodologies were applied to predicting the class labels of the Citeseer data set. The VoteRank metric is used to rank and remove different percentages of the network. S stands for when the network has the initial strong connections only and SW represents when network is augmented with weak ties alongside the strong ties. L and K denotes the number of layers and the power in GCN and SGC framework respectively.

Metric	P	GCN					SGC				
		L	S		SW		K	S		SW	
			Mean	std	Mean	std		Mean	std	Mean	std
Vote Rank	10	0	0.73	0.03	0.74	0.02	0	0.72	0.02	0.73	0.03
		1	0.73	0.03	0.74	0.02	1	0.76	0.01	0.75	0.03
		2	0.66	0.05	0.70	0.04	2	0.76	0.01	0.76	0.03
		3	0.59	0.07	0.65	0.05	3	0.74	0.01	0.76	0.02
		4	0.49	0.15	0.58	0.09	4	0.76	0.01	0.76	0.02
		5	0.43	0.13	0.45	0.19	5	0.75	0.02	0.76	0.02
		6	0.25	0.09	0.34	0.19	6	0.76	0.02	0.76	0.02
		7	0.22	0.07	0.26	0.16	7	0.75	0.02	0.76	0.02
	20	0	0.73	0.02	0.73	0.03	0	0.73	0.01	0.71	0.03
		1	0.73	0.02	0.73	0.03	1	0.71	0.01	0.73	0.04
		2	0.66	0.04	0.68	0.04	2	0.75	0.02	0.74	0.04
		3	0.59	0.03	0.64	0.05	3	0.75	0.01	0.74	0.03
		4	0.54	0.04	0.58	0.09	4	0.77	0.02	0.75	0.04
		5	0.39	0.13	0.50	0.17	5	0.75	0.02	0.75	0.03
		6	0.21	0.02	0.38	0.17	6	0.76	0.02	0.75	0.03
		7	0.21	0.02	0.25	0.12	7	0.75	0.04	0.75	0.03
	30	0	0.72	0.03	0.74	0.03	0	0.73	0.01	0.70	0.03
		1	0.72	0.02	0.73	0.03	1	0.71	0.03	0.72	0.03
		2	0.65	0.03	0.68	0.04	2	0.75	0.01	0.73	0.03
		3	0.58	0.06	0.63	0.07	3	0.72	0.03	0.73	0.02
		4	0.52	0.08	0.57	0.08	4	0.73	0.01	0.73	0.02
		5	0.41	0.15	0.51	0.16	5	0.74	0.03	0.74	0.03
		6	0.28	0.12	0.31	0.14	6	0.74	0.04	0.74	0.03
		7	0.22	0.08	0.27	0.13	7	0.74	0.02	0.74	0.03

Conclusion

This chapter explores the uses of the recently introduced methodology, the Simplified Graph Convolutional Neural Network (SGC); class label inferences are produced based on the network structure, represented by an adjacency matrix, in combination with node feature vectors. There is interest in exploring this model in more depth since it provides a succinct yet expressive formulation for describing how nodes can influence class label prediction within a network. Besides the parameters fitted in order to optimize the target label prediction, there is only a single parameter value k, which requires manual tuning. This parameter is related to the number of layers S^k (described in Section 7).

The exploration conducted here investigates the degree to which the accurate prediction of class labels is reduced by removing percentages of the network ranked by centrality metrics. This provides evidence to the practitioner who collects data, that may contain gaps in the network, and needs to know if the conclusions can be drastically affected by missing data on key nodes as to whether the the SGC is sensitive to such issues. Three different network centrality measures are used to select removal nodes: betweenness, closeness, and VoteRank. We find that the methodology does manage to produce analogous predictions based upon different percentages of removal (10/20/30). The largest observed changes were when the nodes were selected for removal with the VoteRank algorithm and not with betweenness or closeness. This shows that the SGC label assignments are more sensitive to the local label information derived from the features of the local nodes than well connected groups of nodes in the center of the network. This also explains why it has displayed the ability to be robust in its predictions.

The other question explored is whether the results would change if the SGC was supplied an adjacency matrix that contained the 'triangulated edges' to begin with. The existing edges in the adjacency matrix can be referred to as *strong-ties* as they are direct links; the edges that connect

friends-of-friends (produced from triangulation A^2), can be referred to as *weak-ties*. A matrix with both of these edge sets was supplied to the SGC to compare the accuracy predictions. There is considerable sociological literature discussing the importance of these edges in helping to discover important connections. Our results show a degraded outcome with the exception of when nodes were removed with the VoteRank algorithm. This indicates that the inclusion of the *weak-ties* provides a more robust edge set when important local nodes are removed. The results do not show an ability to improve the prediction of class labels for low removal percentages when *weak-ties* are included.

The datasets used in this study contained monolithic graphs, where every node is reachable from any other node. There are many datasets where the data contains disjoint graphs, and this can be particularly common when the observational capabilities are limited in comparison to the process. A notable example is with protein interaction graphs. Applying the investigation taken here with such data would alter the adjacency matrix but not in a way that would cause a failure in its ability to follow the procedures described. Since the exploration did not depend upon a small fraction of the number of nodes, the study could continue with such data as long as the distribution of the relative betweenness and closeness is not excessively skewed for the subgraphs. The investigation therefore can be conducted on a wide range of datasets to explore the role of weak ties in the networks. Corporate networks are an interesting avenue for extensions as the nodes would be more 'complex' entities which may rely on their network connections in different ways. A key aspect of the extendibility is the overhead of the approach. Since the parameter, feature and adjacency matrix are combined with linear operators with a non-nested set of intermediate features, inferences are relatively cheaper than other approaches that build deeper trees and introduce further non-linearities.

CHAPTER 8: CONCLUSION

This dissertation introduces several new stochastic sampling and machine learning techniques for predicting the evolution of social networks. This chapter presents an overview of the research contributions.

First we introduce a versatile community-based data mixture model for predicting social network evolution. Community features are used to forecast and retrieve the most similar historical data partition; Our experimental results show that by simulating communities separately, all resolutions of the network are more accurately simulated. This was supported by comparing community based models to agent-based models that used features focused on other resolutions, MACM and MBM. Both community based models performed better on average on all resolutions, with few exceptions. By comparing the simplified community model, which extracts community based features from the network as a whole rather than per community, our model outperformed in user-level performance. This indicates that by extracting an historical representation of different communities in a social network, the whole network can be more accurately simulated. The strengths of our approach are in its versatility, convenience, and computational complexity.

The main weakness is that, unlike other aspects of the Deep Agent Framework, it does not provide an explanation for the users behavior nor does it capture peer to peer influence. Thus in practice, we have used sampled historical data to complement our other simulation techniques, rather than as a standalone alternative. Our work could be extended by investigating the use of new features such as diffusion delay of user actions and distribution of user types. Stochastic features such as the probability that users will contribute to information spread could also be employed to represent the network. Increasing the number of features used to represent the network would facilitate the use of machine learning techniques, potentially reducing prediction error.

Chapter 6 proposes the *a_LSTM* which considers data from multiple social networks in order to improve prediction performance. The dataset used to test this model is composed of events coming from Twitter, Reddit, and GitHub. The investigation also considers the related question of community topic domain activity where the connected components on a topic are counted as active or not over time providing a non-negative real number time series. This data is collected for GitHub before applying the LSTM and a_LSTM models. In both cases the a_LSTM shows better predictive accuracy. We conclude that the information across social networks can provide valuable information in predicting the bursts within a single network. This is a new finding that has not been discussed in related work, due to the relative shortage of cross-platform datasets.

Predictions of bursts can be improved upon using the proposed a_LSTM. Results are shown using data from three major social networks that have a global coverage. One avenue for future work would be to explore other topics, such as news or entertainment, and other social media platforms, such as YouTube and Instagram, to see if their behavior exhibits similar dynamics. The cross network association may not be exhibited across different forms of content or social media platforms, if the content sharing remains within the networks of the content origin. In this case the proposed a_LSTM model may not perform well. Future investigations could also explore how the data can be clustered into topic groups to see if they exhibit repetitive bursts similar to 'shock' trains. In our future work, we are exploring the use of LSTM event volume prediction for improving the performance of our agent-based simulation of social media platforms.

Chapter 7 explores the uses of the recently introduced methodology, the Simplified Graph Convolutional Neural Network (SGC); class label inferences are produced based on the network structure, represented by an adjacency matrix, in combination with node feature vectors.

The exploration conducted in 7 investigates the degree to which the accurate prediction of class labels is reduced by removing percentages of the network ranked by centrality metrics. This provides

evidence to the practitioner who collects data, that may contain gaps in the network, and needs to know if the conclusions can be drastically affected by missing data on key nodes as to whether the the SGC is sensitive to such issues. Three different network centrality measures are used to select removal nodes: betweenness, closeness, and VoteRank.

The other question explored is whether the results would change if the SGC was supplied an adjacency matrix that contained the 'triangulated edges' to begin with. The existing edges in the adjacency matrix can be referred to as *strong-ties* as they are direct links; the edges that connect friends-of-friends (produced from triangulation A^2), can be referred to as *weak-ties*. A matrix with both of these edge sets was supplied to the SGC to compare the accuracy predictions. There is considerable sociological literature discussing the importance of these edges in helping to discover important connections.

The datasets used in this study contained monolithic graphs, where every node is reachable from any other node. There are many datasets where the data contains disjoint graphs, and this can be particularly common when the observational capabilities are limited in comparison to the process. A notable example is with protein interaction graphs. Applying the investigation taken here with such data would alter the adjacency matrix but not in a way that would cause a failure in its ability to follow the procedures described. Since the exploration did not depend upon a small fraction of the number of nodes, the study could continue with such data as long as the distribution of the relative betweenness and closeness is not excessively skewed for the subgraphs. The investigation therefore can be conducted on a wide range of datasets to explore the role of weak ties in the networks. Corporate networks are an interesting avenue for extensions as the nodes would be more 'complex' entities which may rely on their network connections in different ways. A key aspect of the extendibility is the overhead of the approach. Since the parameter, feature and adjacency matrix are combined with linear operators with a non-nested set of intermediate features, inferences are relatively cheaper than other approaches that build deeper trees and introduce further non-

linearities.

Although the techniques proposed in this dissertation can be employed to simulate users' online behaviors in the short term, creating a high-fidelity simulation of user behavior over the long term remains an open research challenge.

APPENDIX A: PLOTS ILLUSTRATING THE EFFECT OF STRONG VS. WEAK TIES WITH DIFFERENT NODE REMOVAL ORDERINGS

Betweenness Removal:

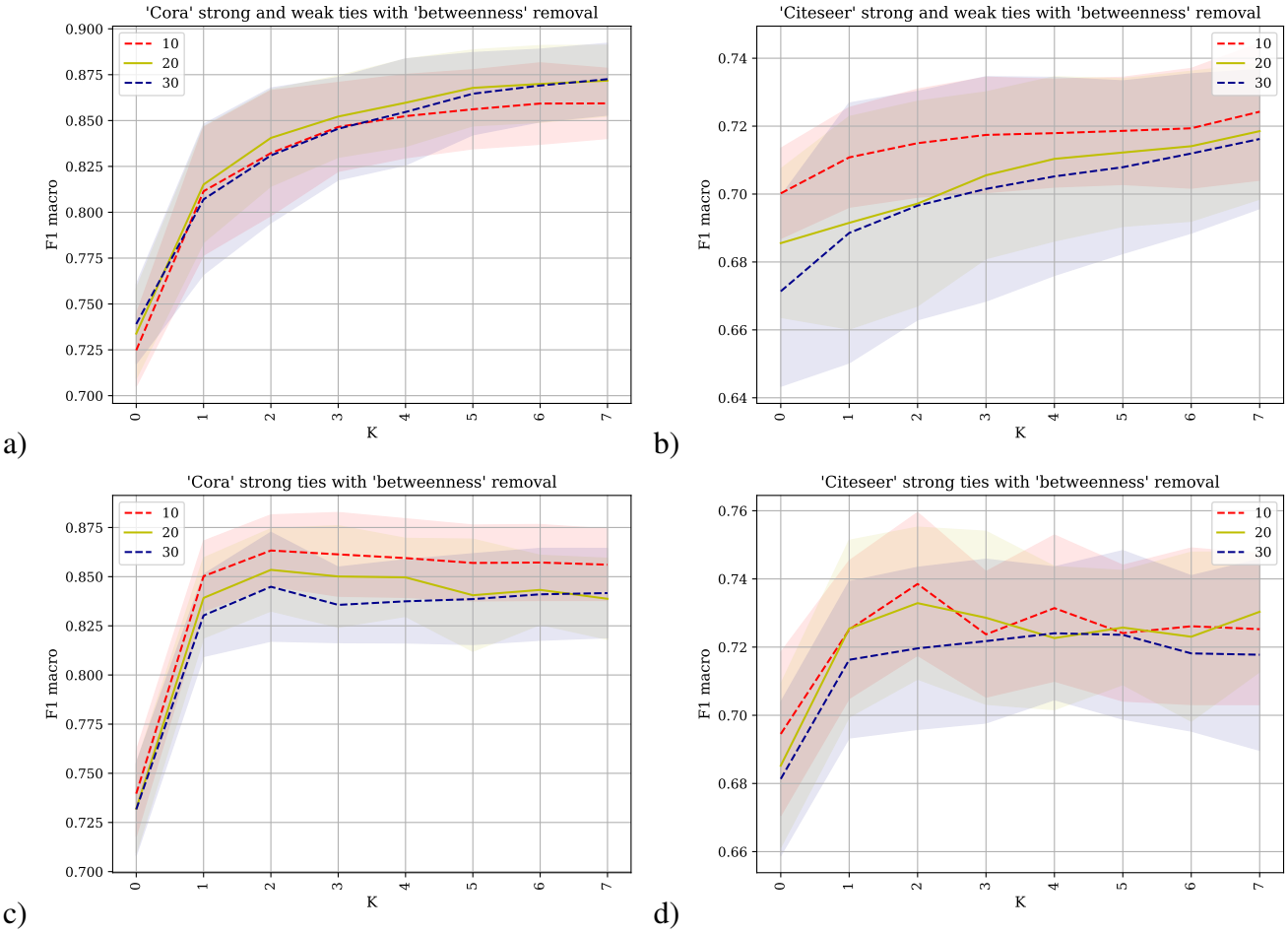


Figure A.1: The SGC methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the F1 macro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

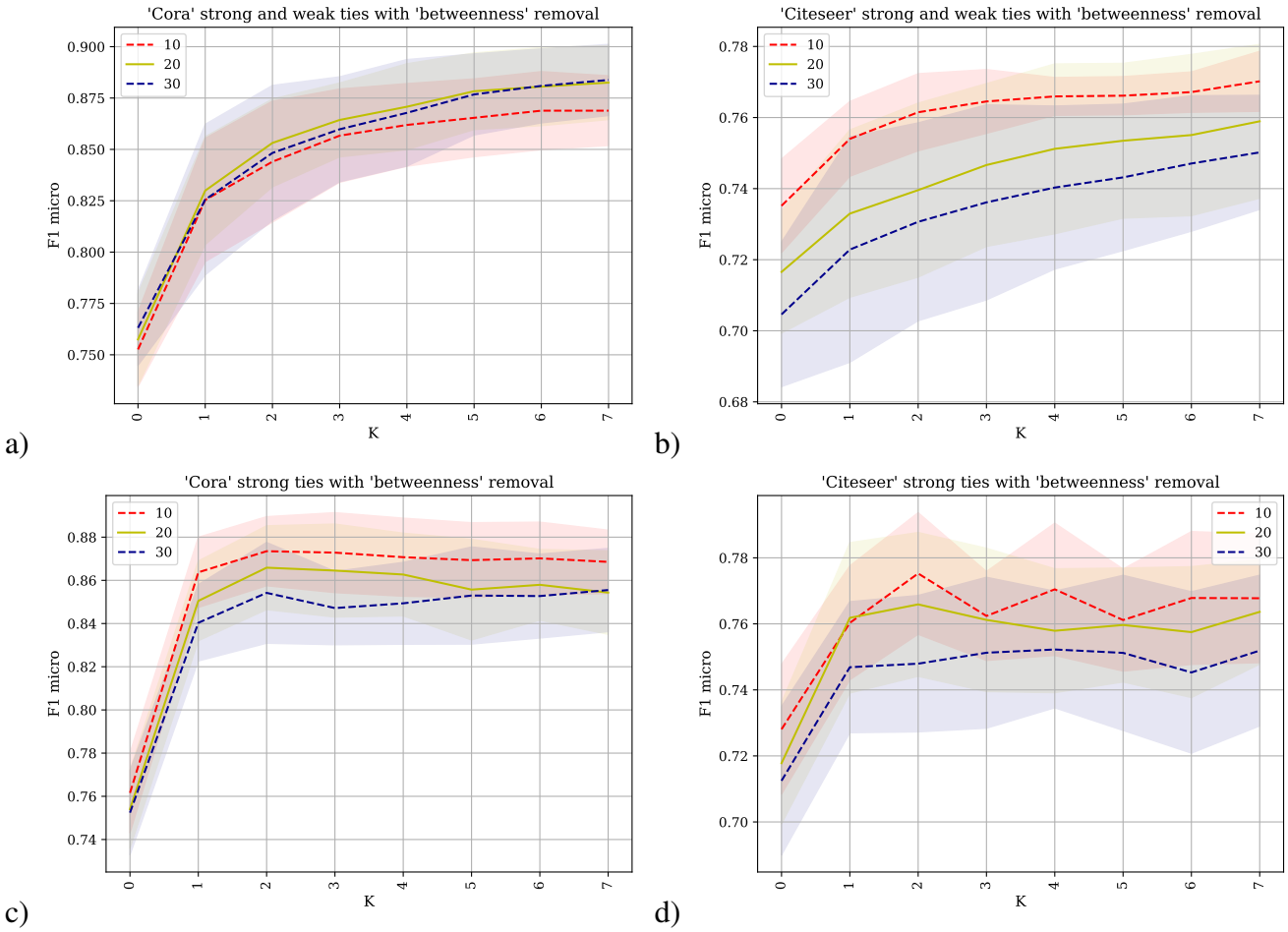


Figure A.2: The SGC methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the F1 micro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

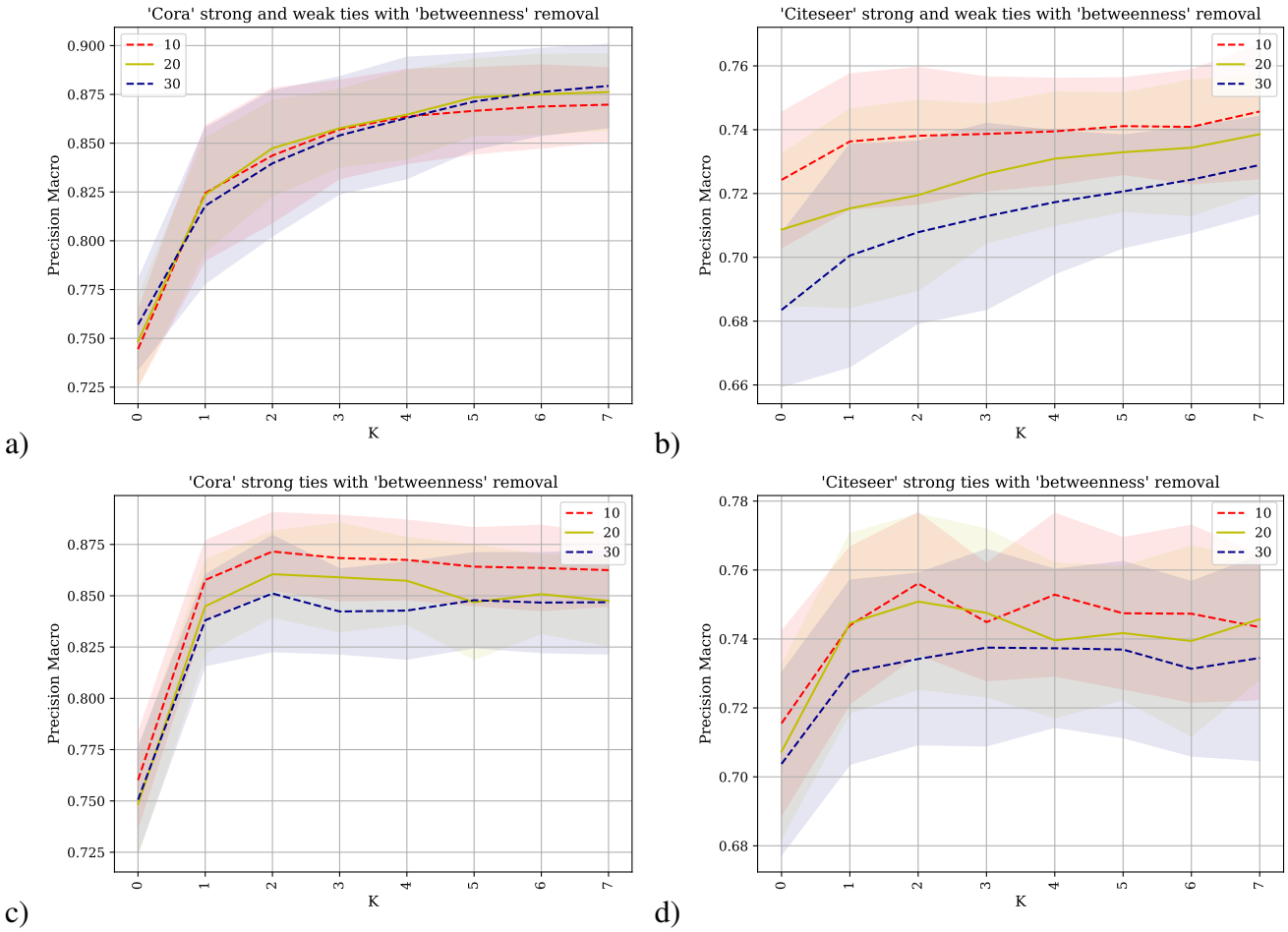


Figure A.3: The SGC methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the precision macro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

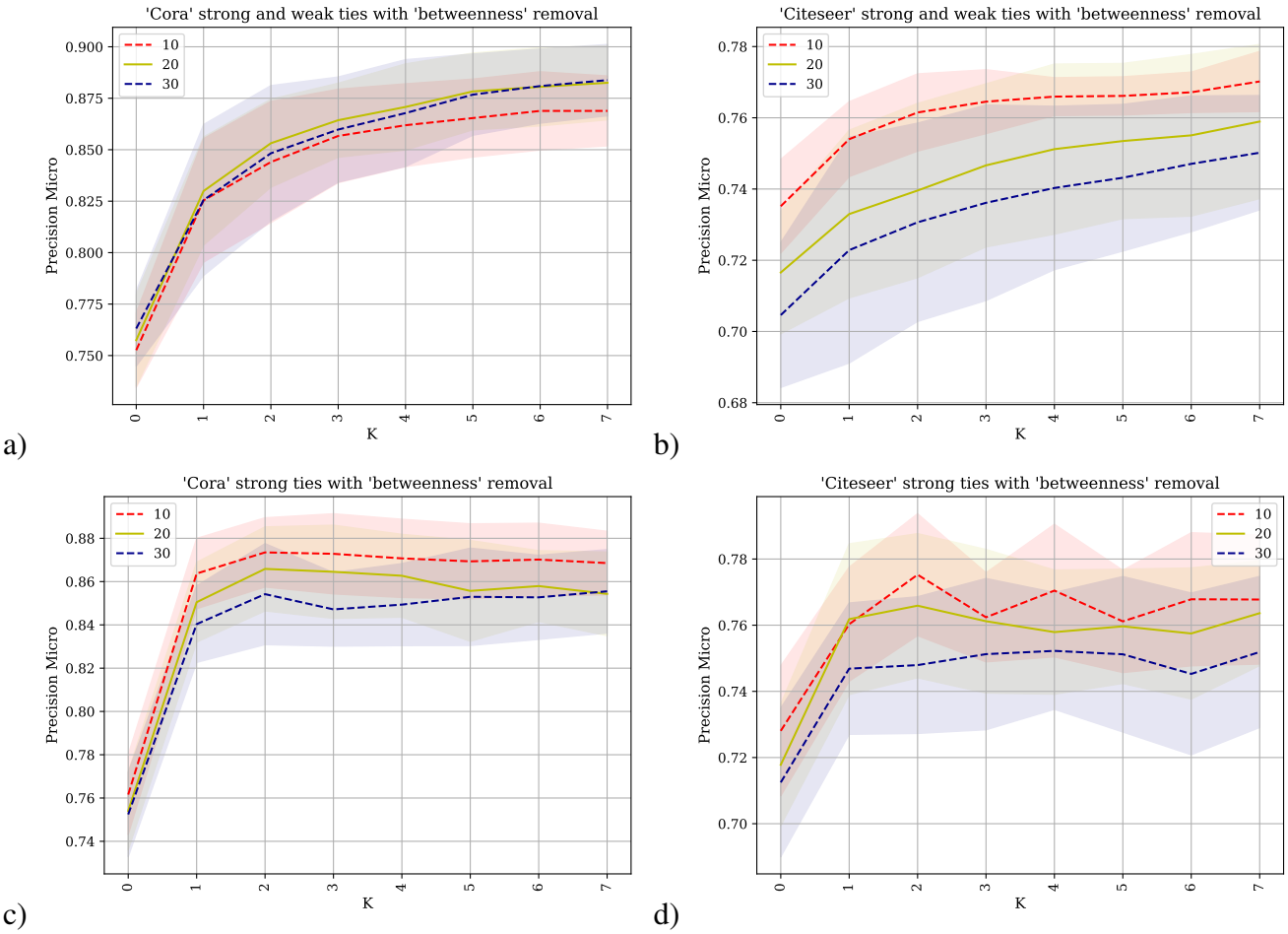


Figure A.4: The SGC methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the precision micro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

Voterank Removal:

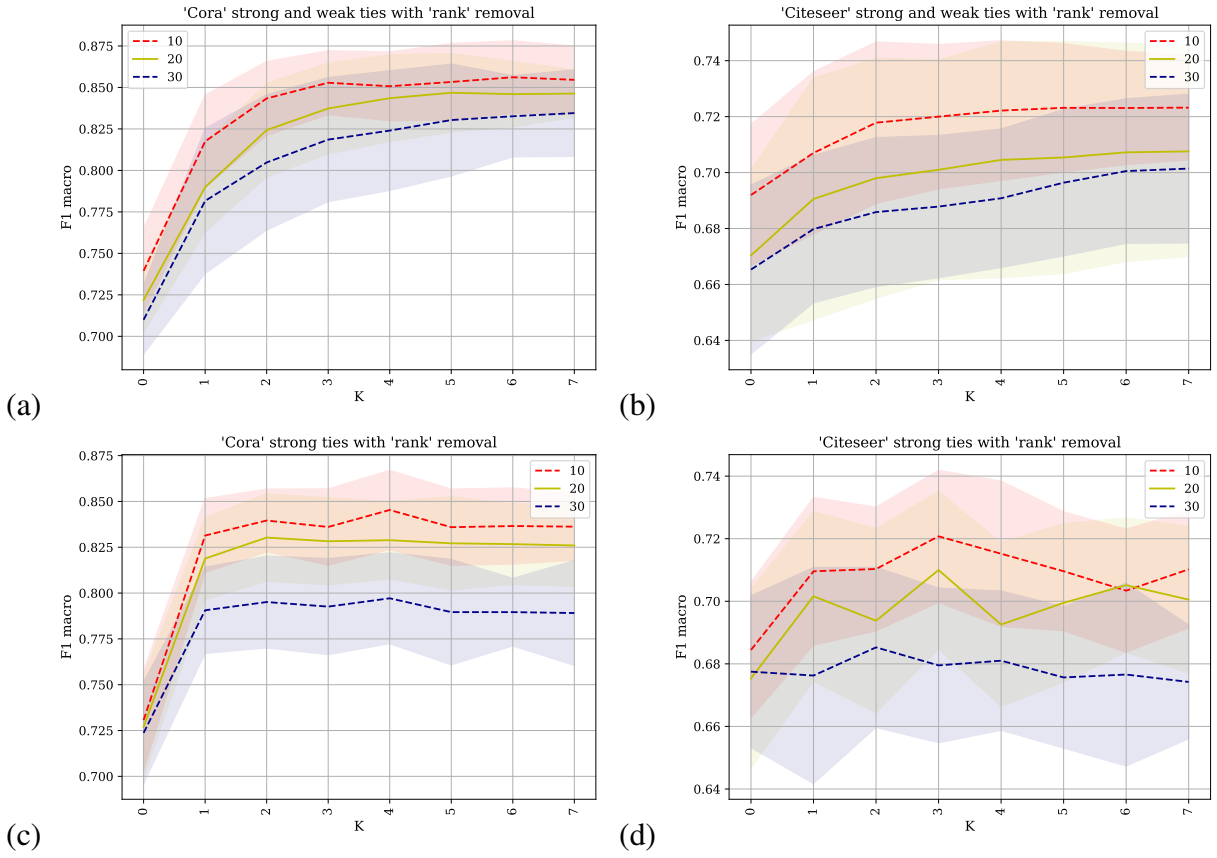


Figure A.5: The SGC methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 macro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

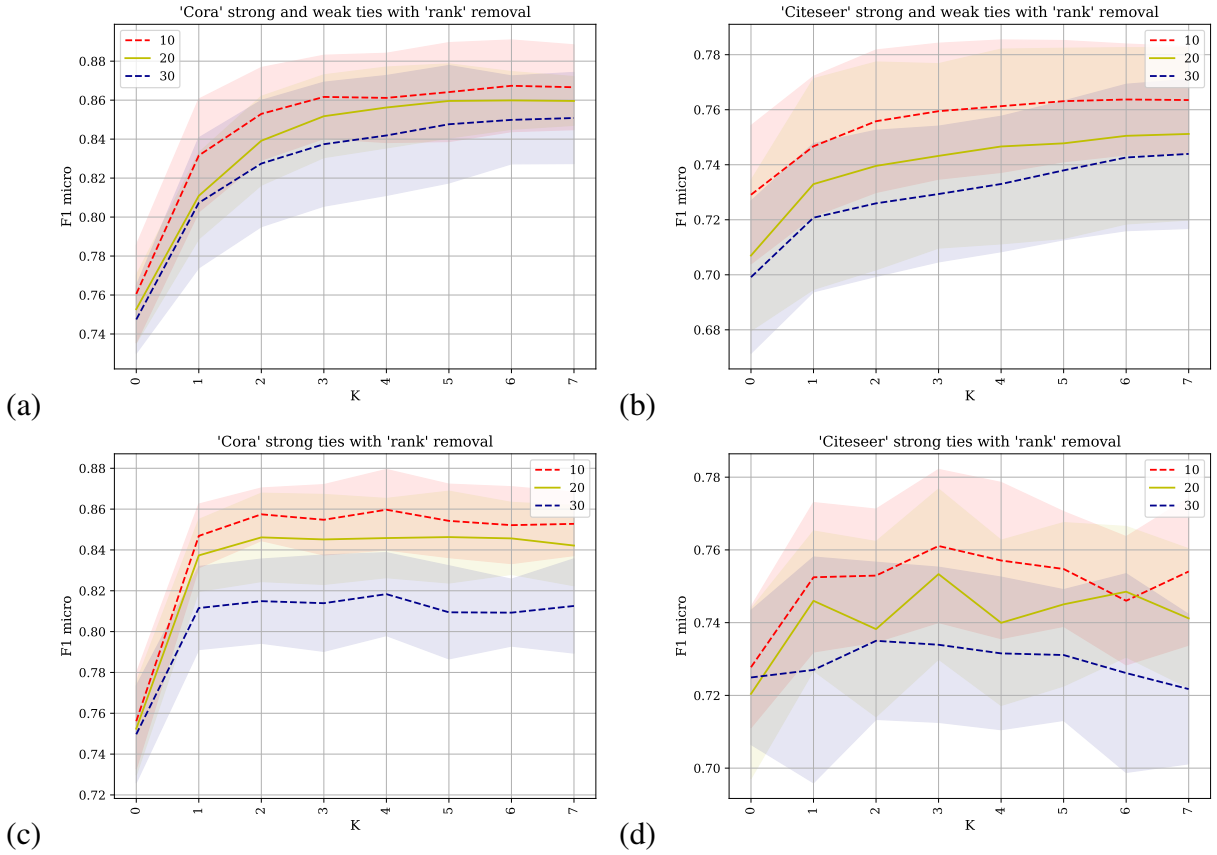


Figure A.6: The SGC methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 micro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

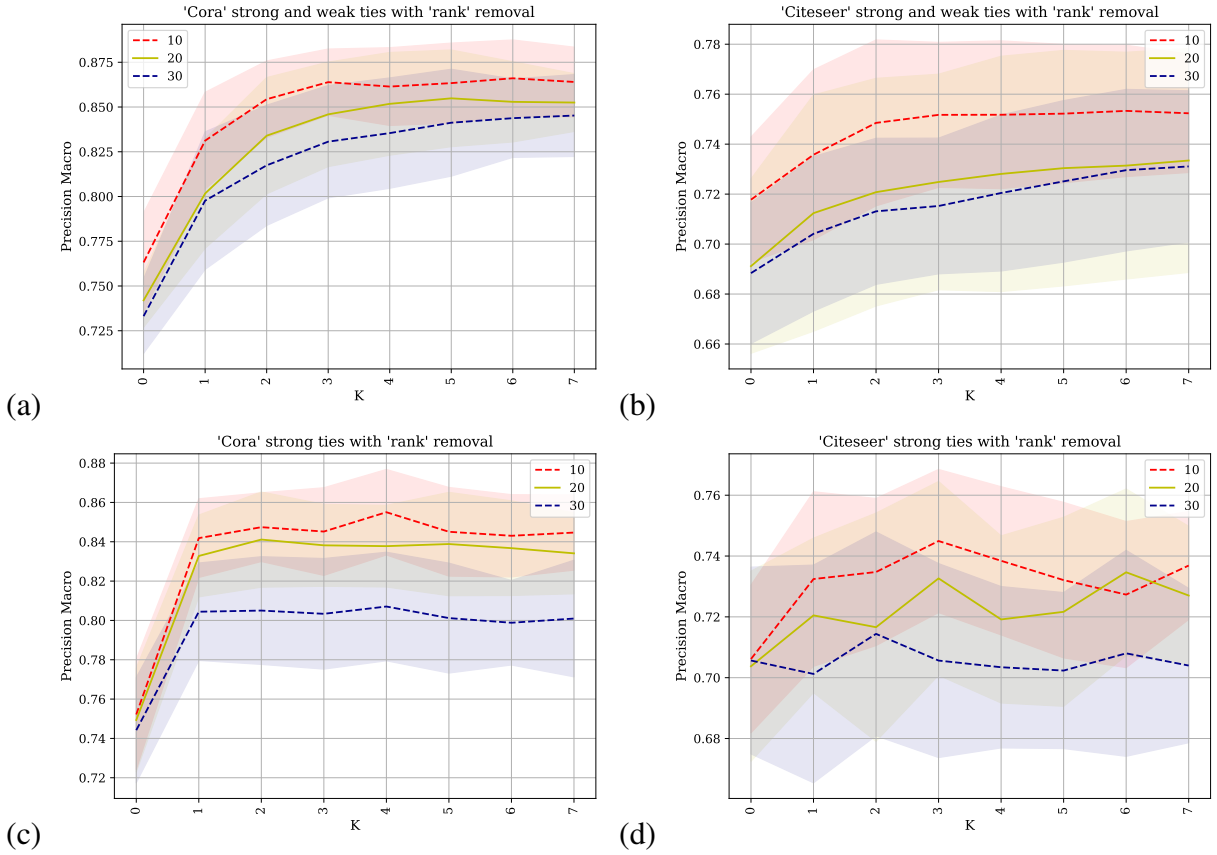


Figure A.7: The SGC methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision macro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

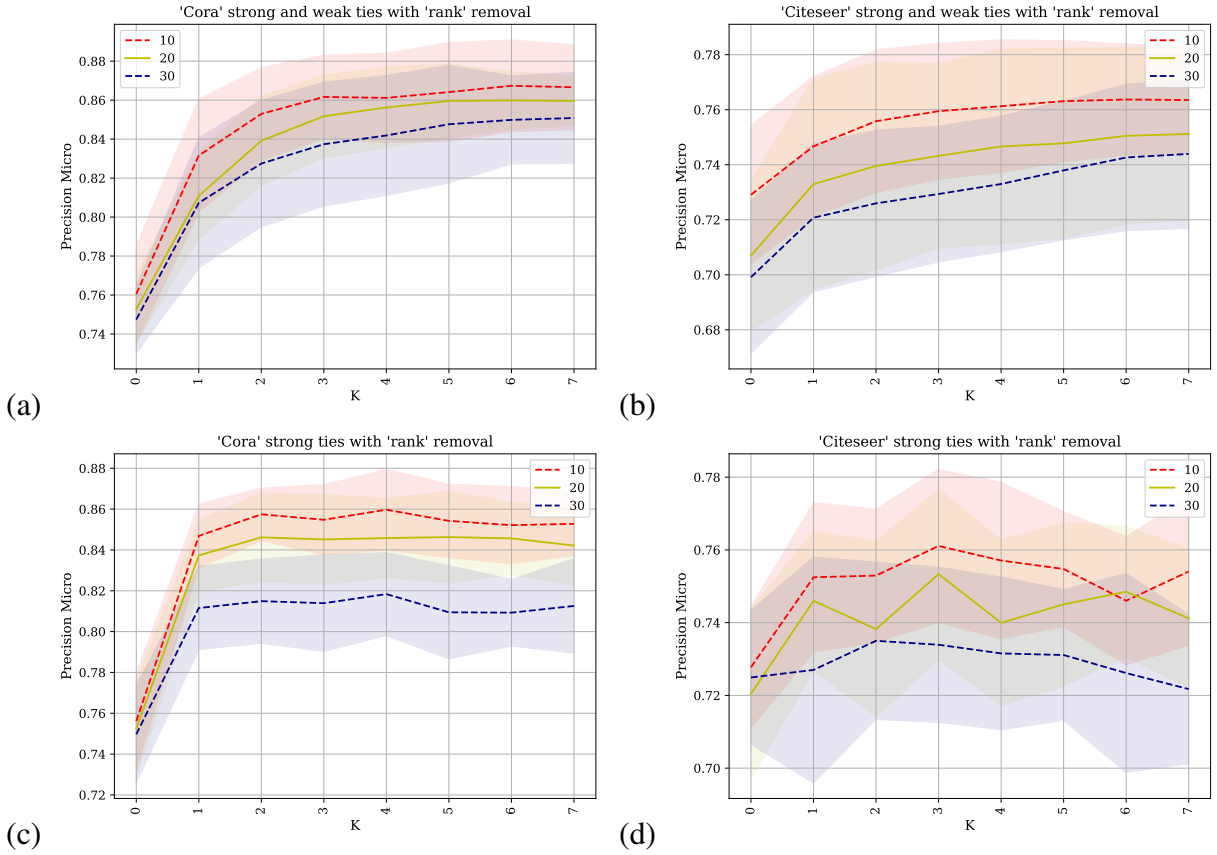


Figure A.8: The SGC methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision micro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

Closeness Removal:

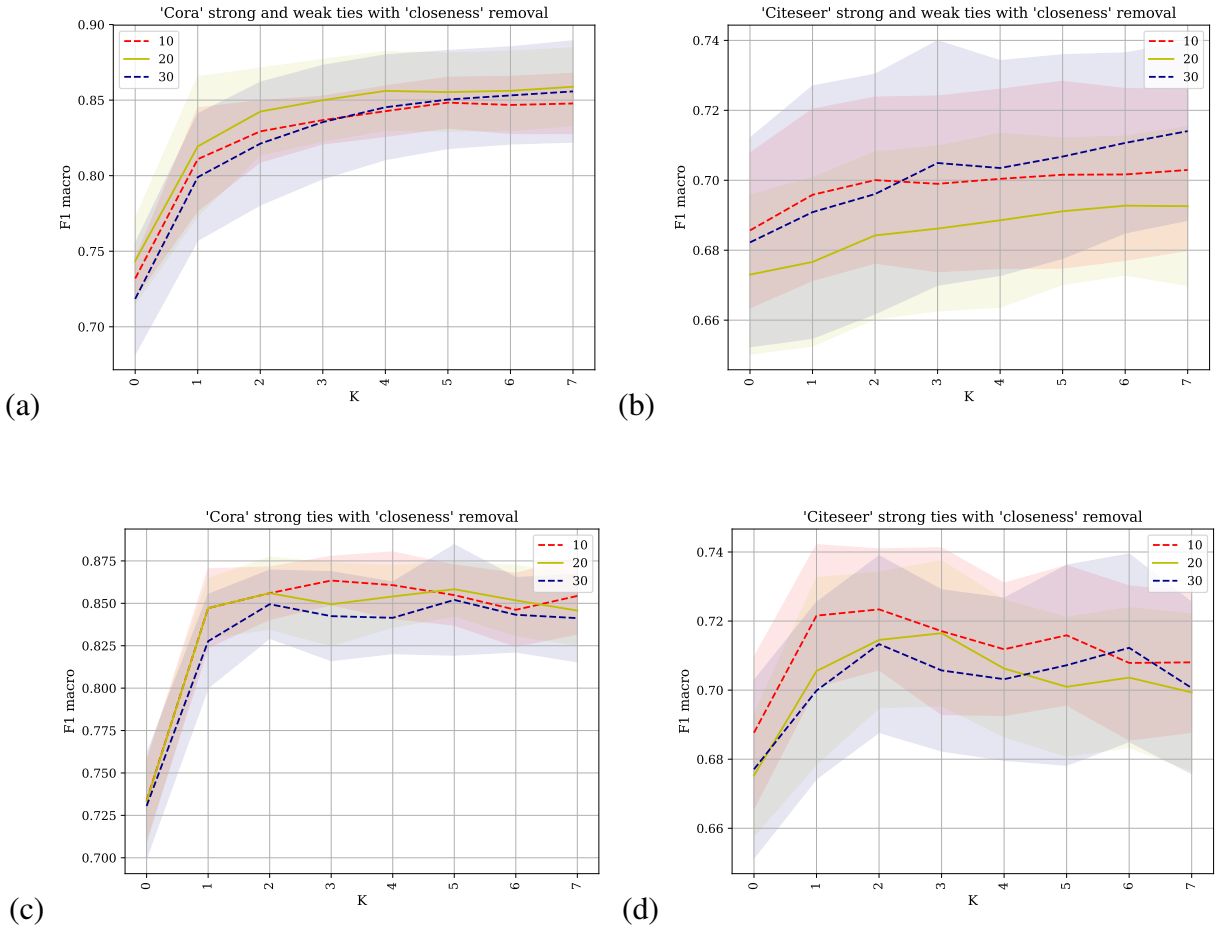


Figure A.9: The Graph Convolutional Networks (SGC) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 macro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

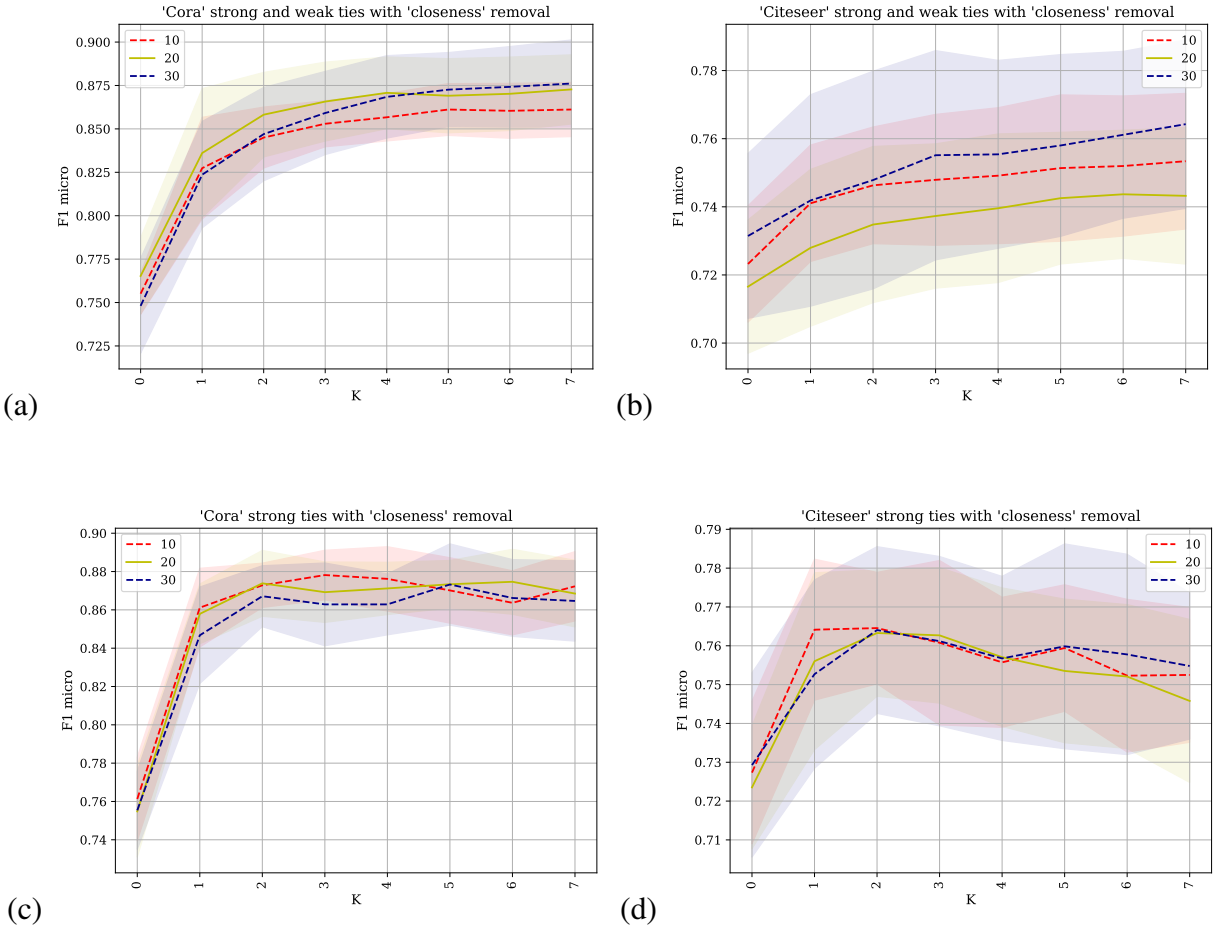


Figure A.10: The Graph Convolutional Networks (SGC) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 micro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

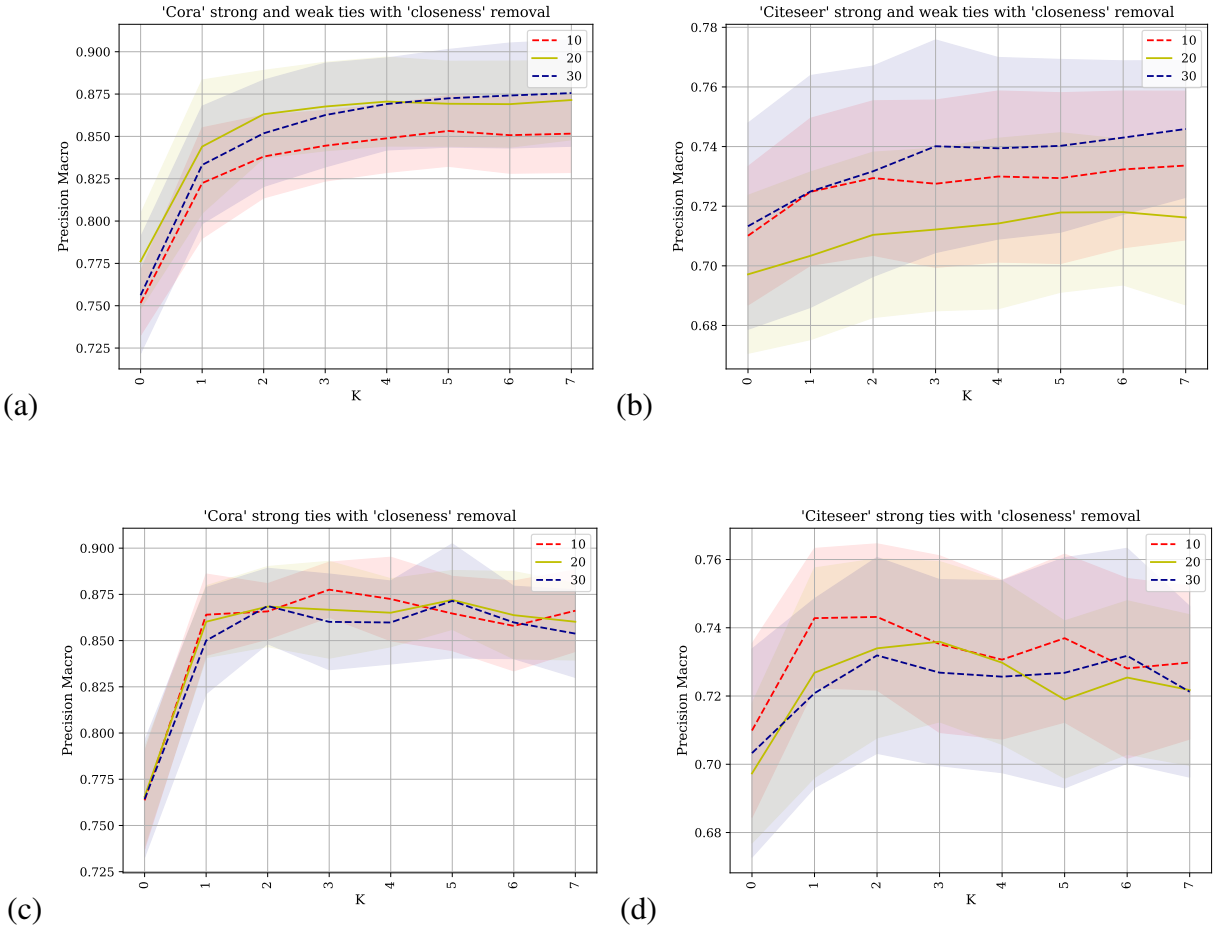


Figure A.11: The Graph Convolutional Networks (SGC) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision macro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

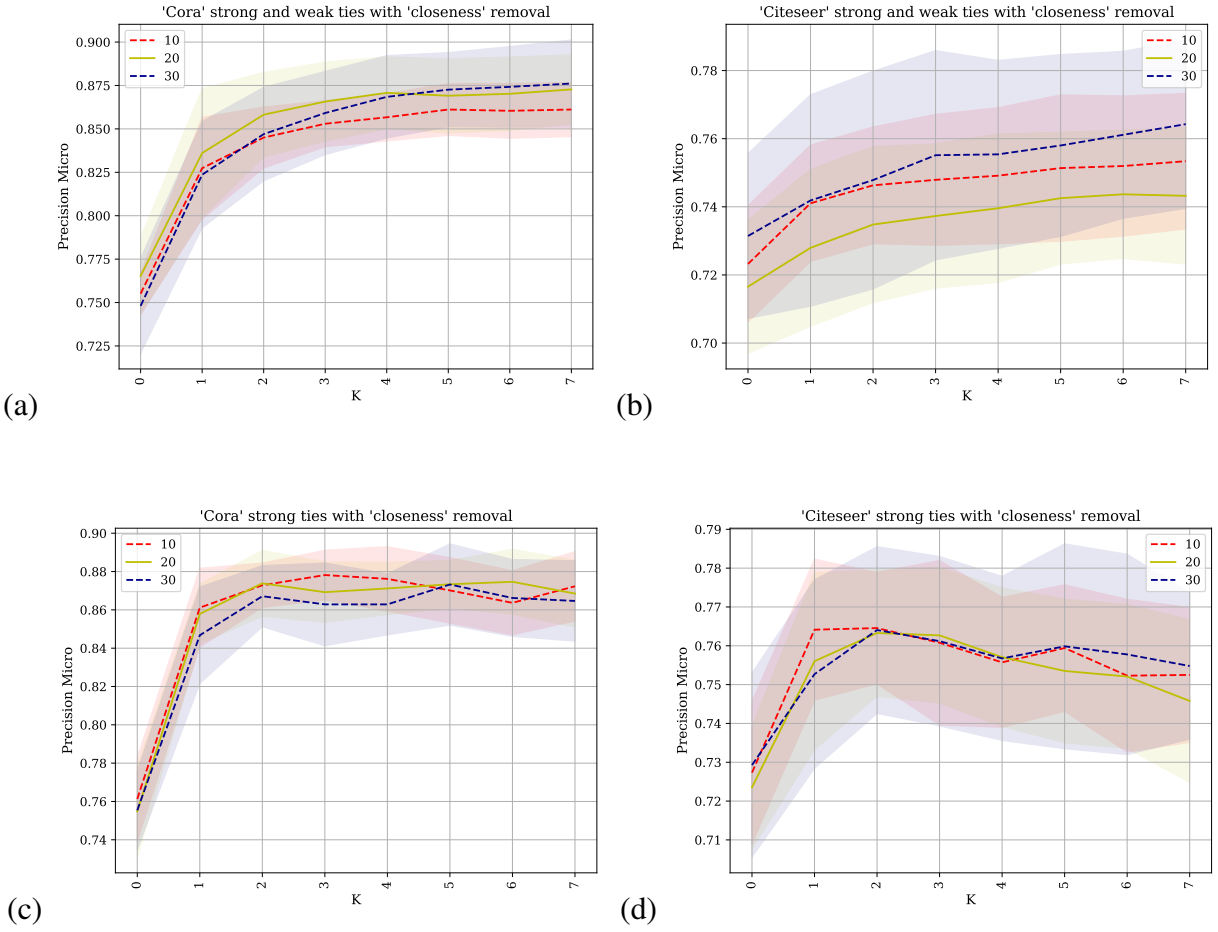


Figure A.12: The Graph Convolutional Networks (SGC) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision micro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

Betweenness Removal:

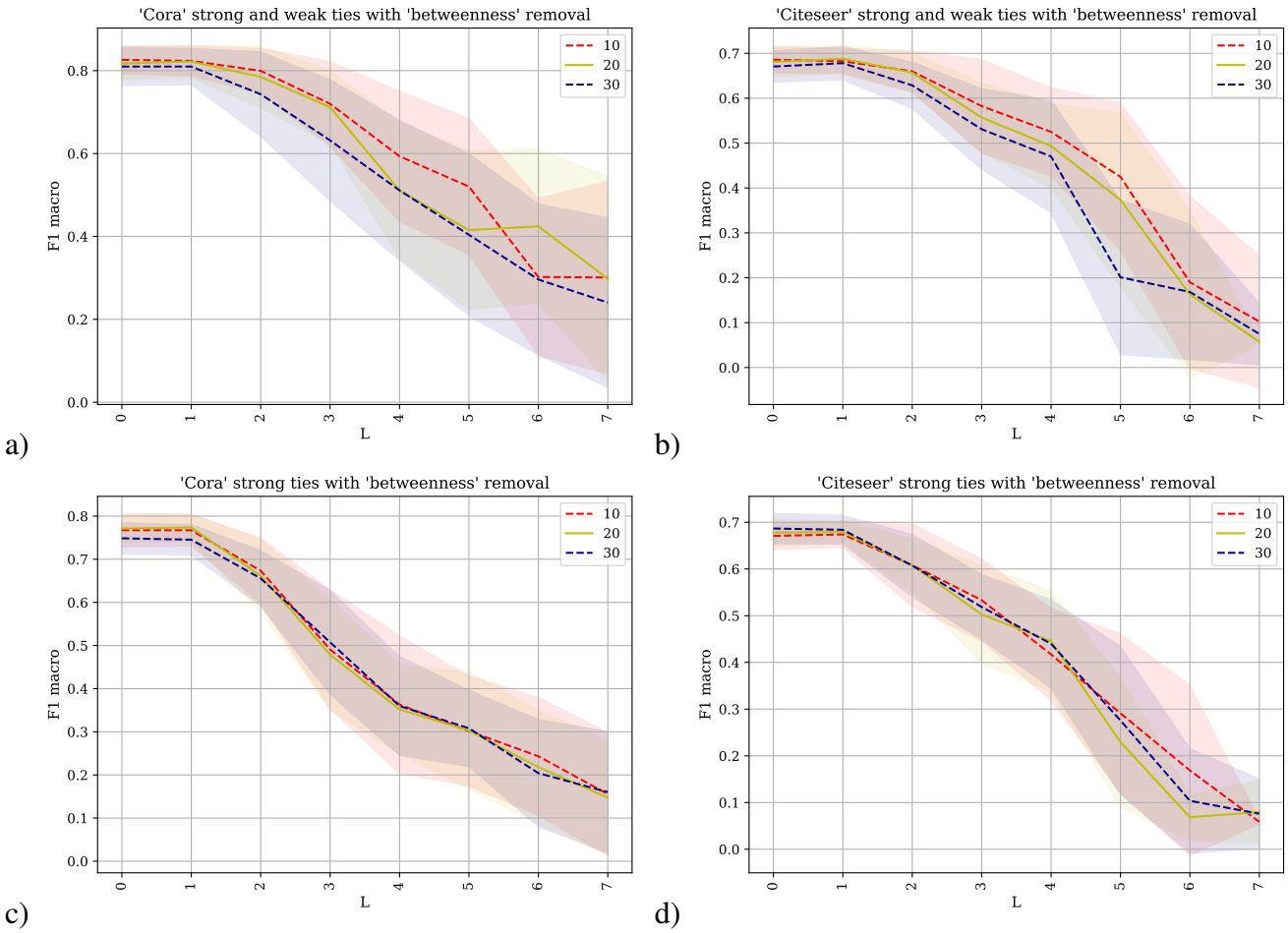


Figure A.13: The GCN methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the F1 macro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

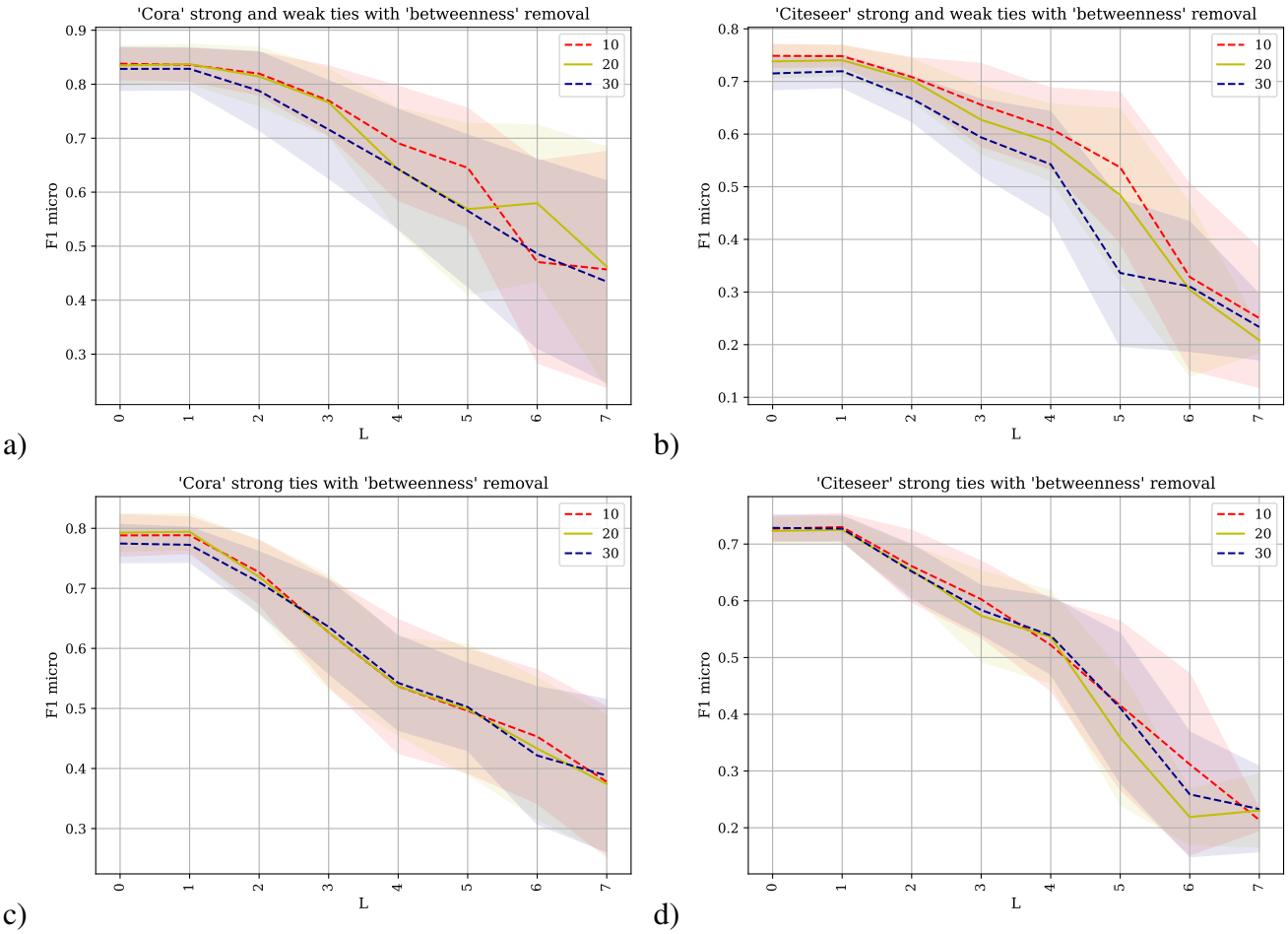


Figure A.14: The GCN methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the F1 micro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

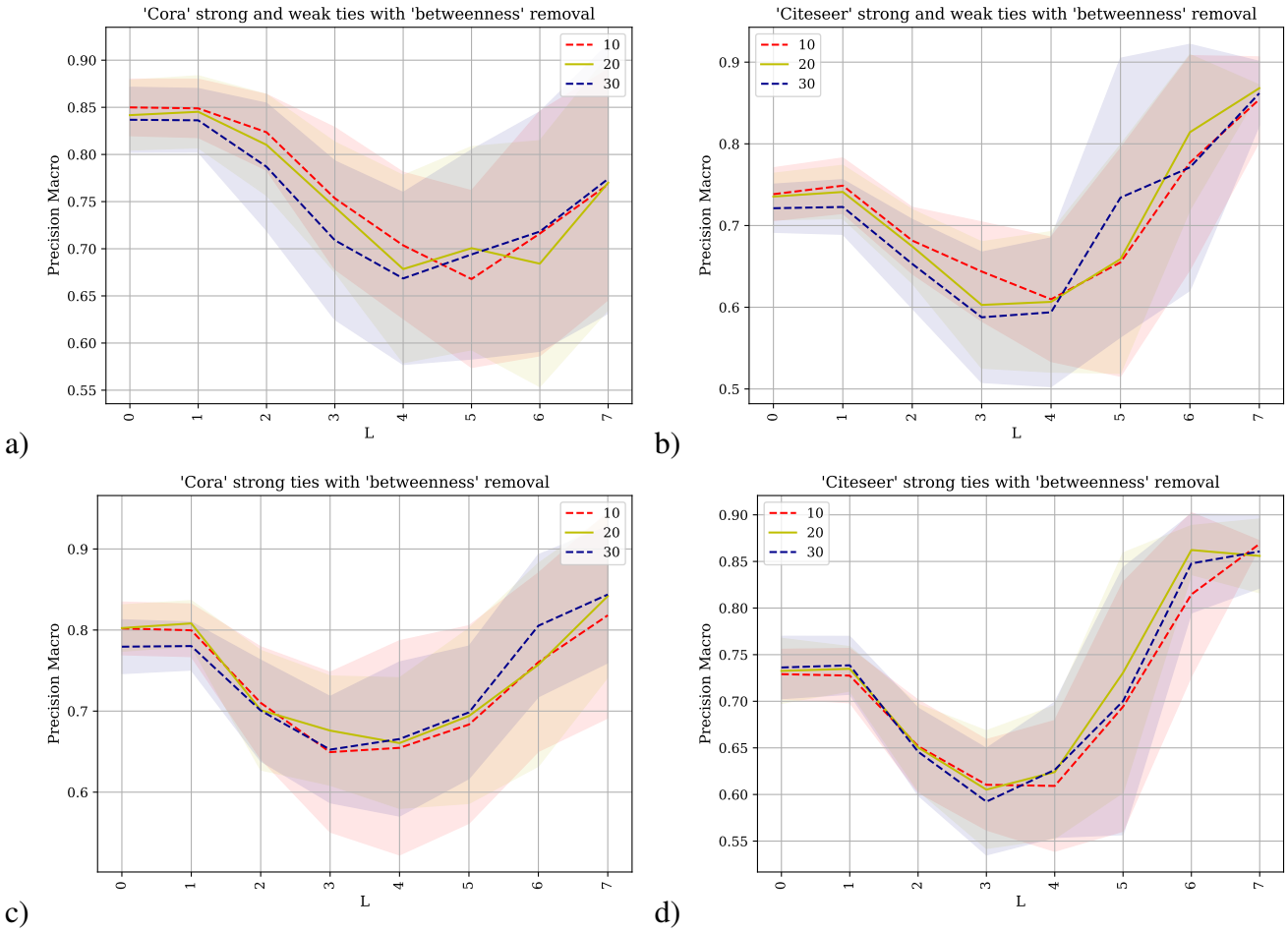


Figure A.15: The GCN methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the precision macro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

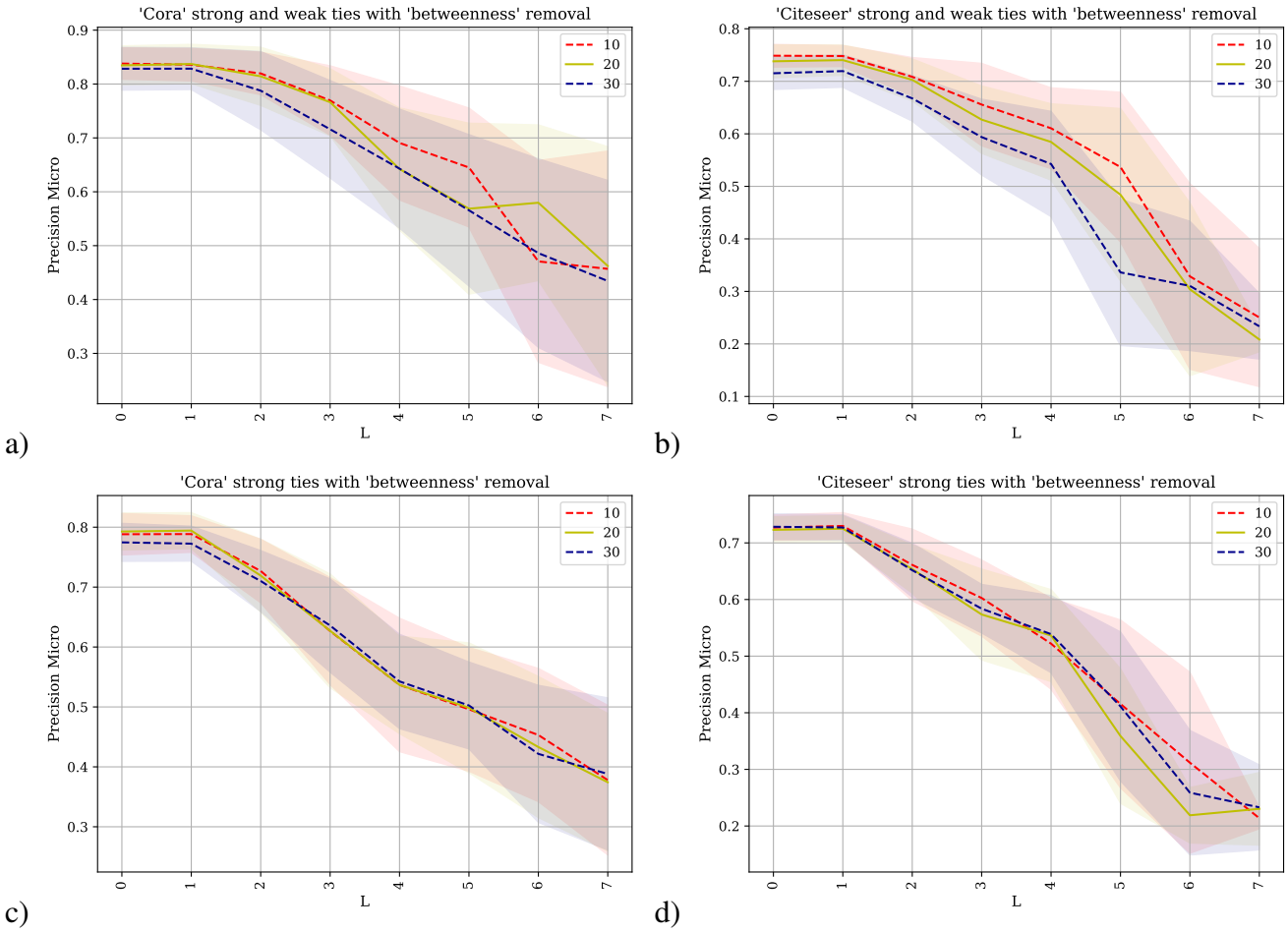


Figure A.16: The GCN methodology is applied to predicting the class labels of the datasets Cora and Citeseer where the precision micro is plotted against the parameter L. The betweenness metric is used to rank and remove different percentages of the network. Subfigures a) and b) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and subfigures c) and d) show the results when the original adjacency matrix containing only *strong-ties* is used.

Voterank Removal:

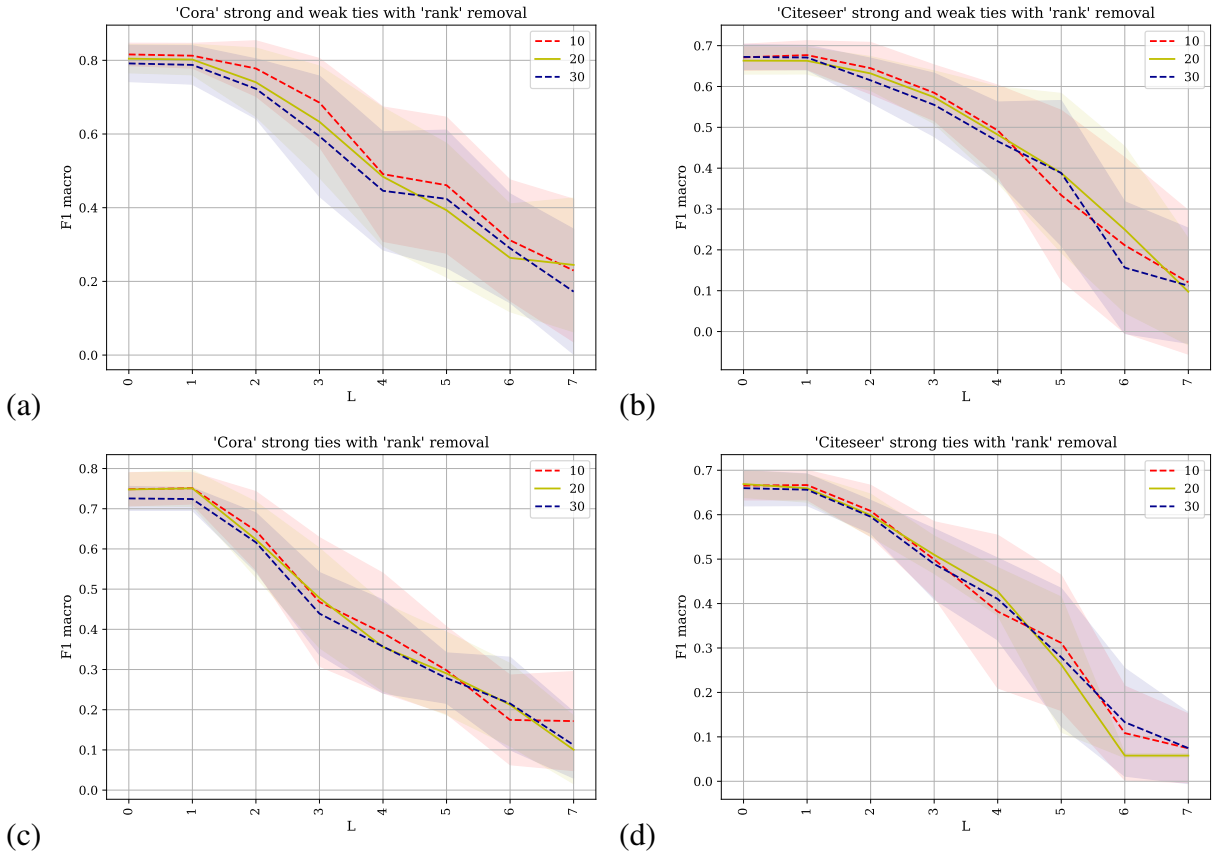


Figure A.17: The GCN methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 macro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

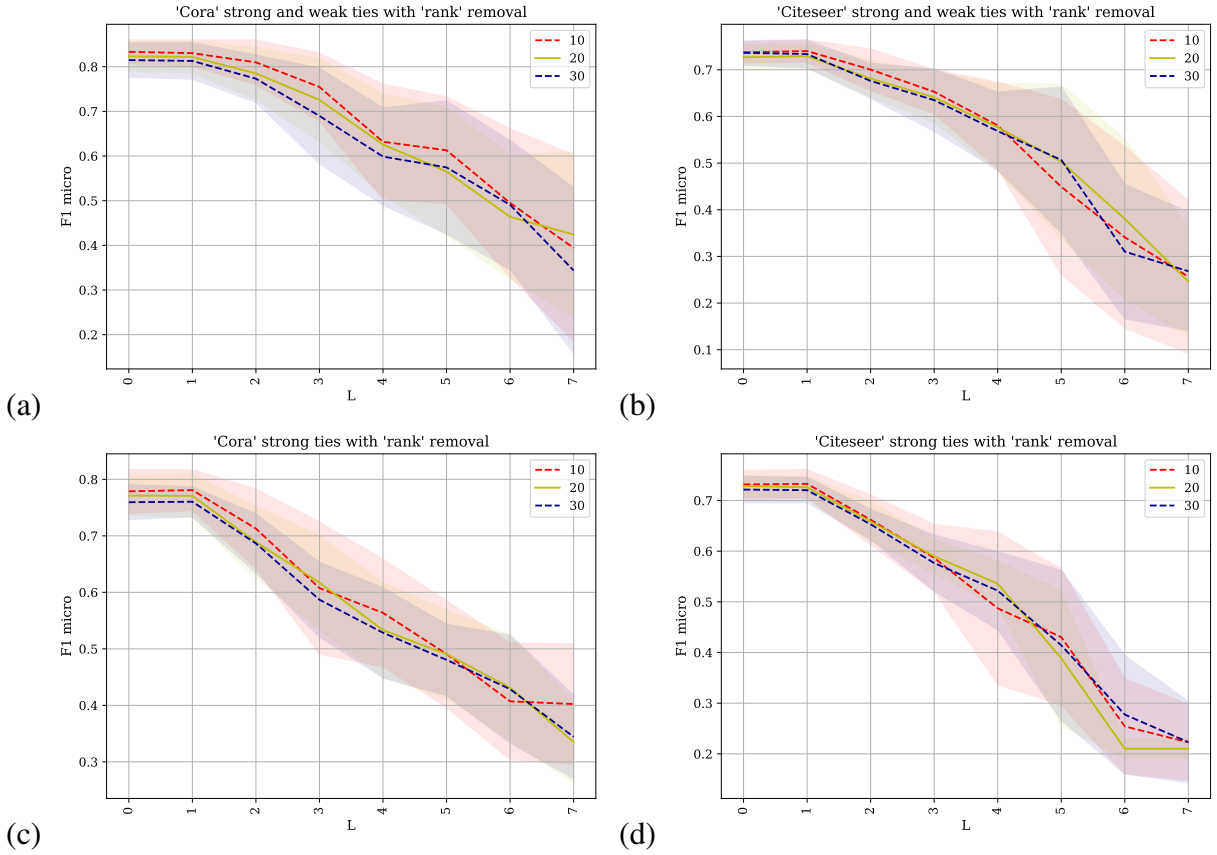


Figure A.18: The GCN methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 micro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

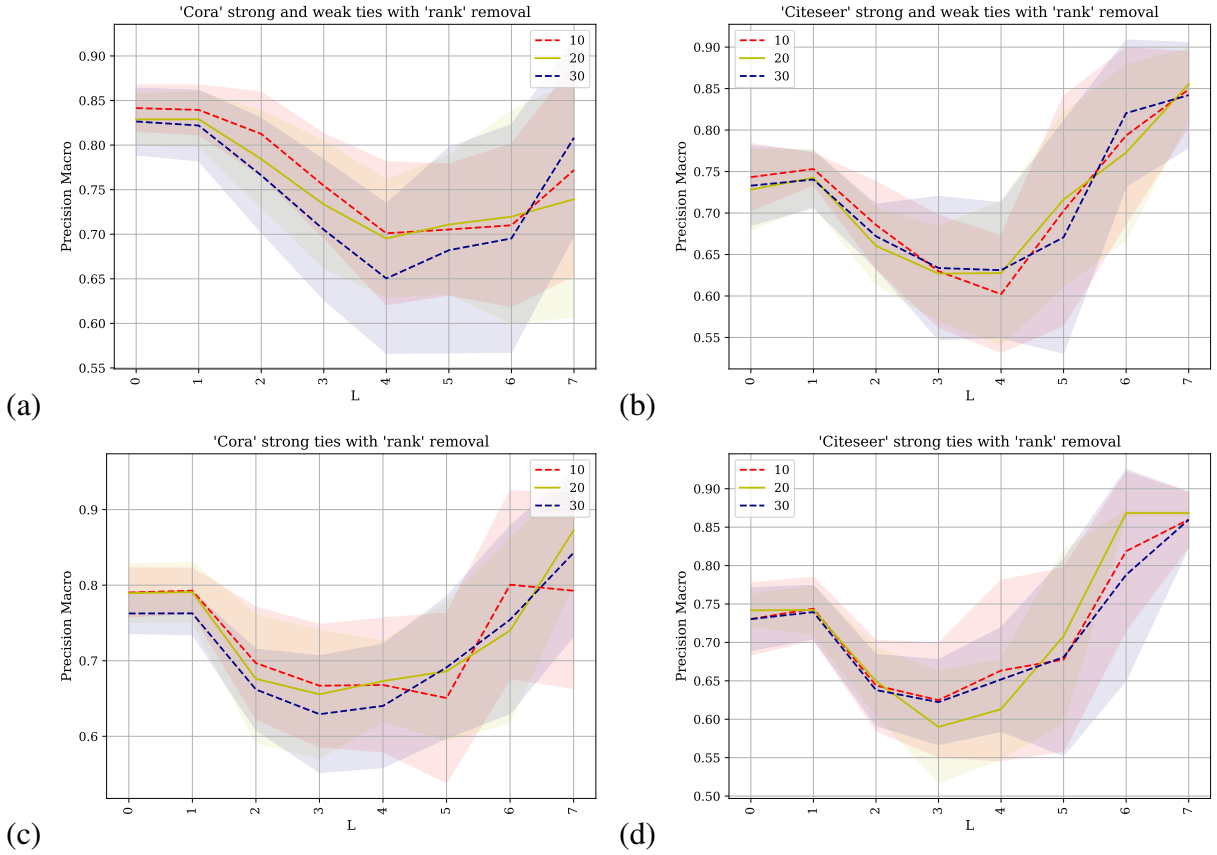


Figure A.19: The GCN methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision macro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

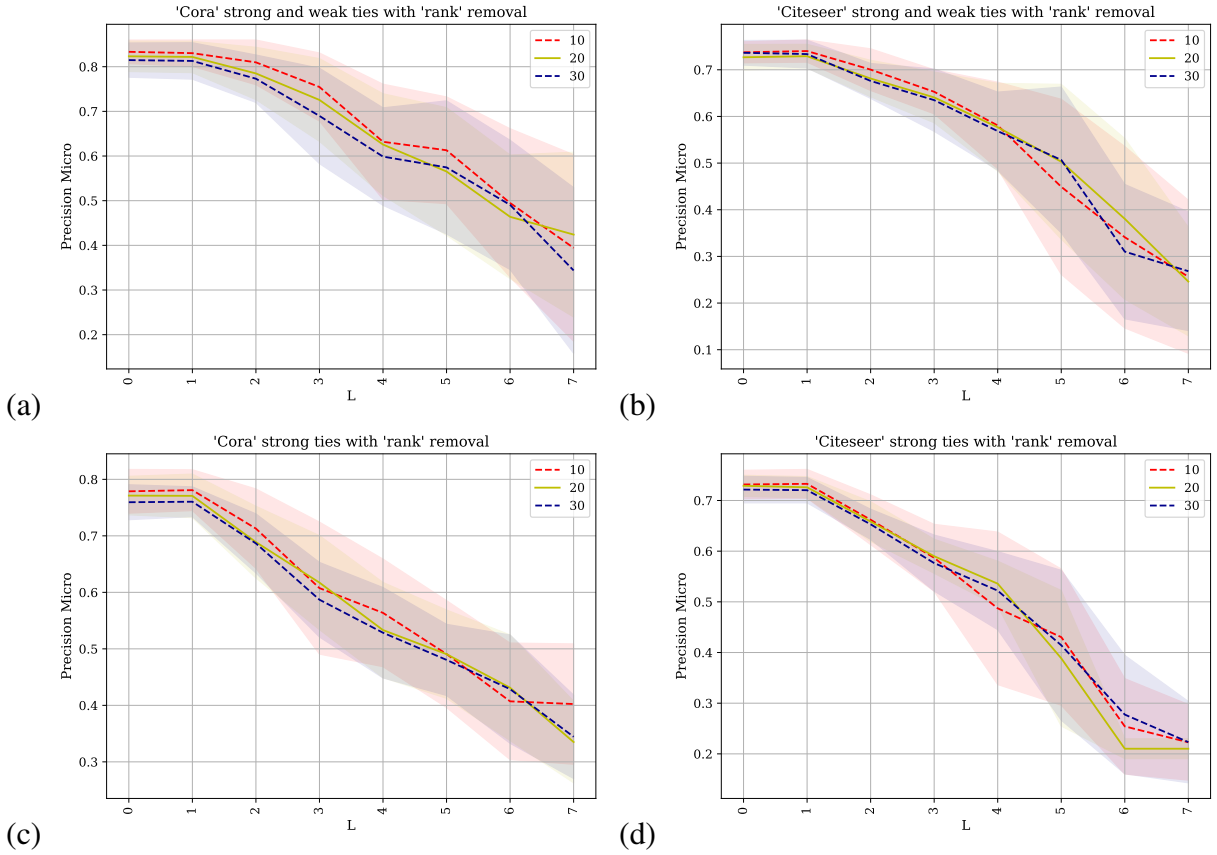


Figure A.20: The GCN methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision micro is plotted against the parameter l. The VoteRank metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

Closeness Removal:

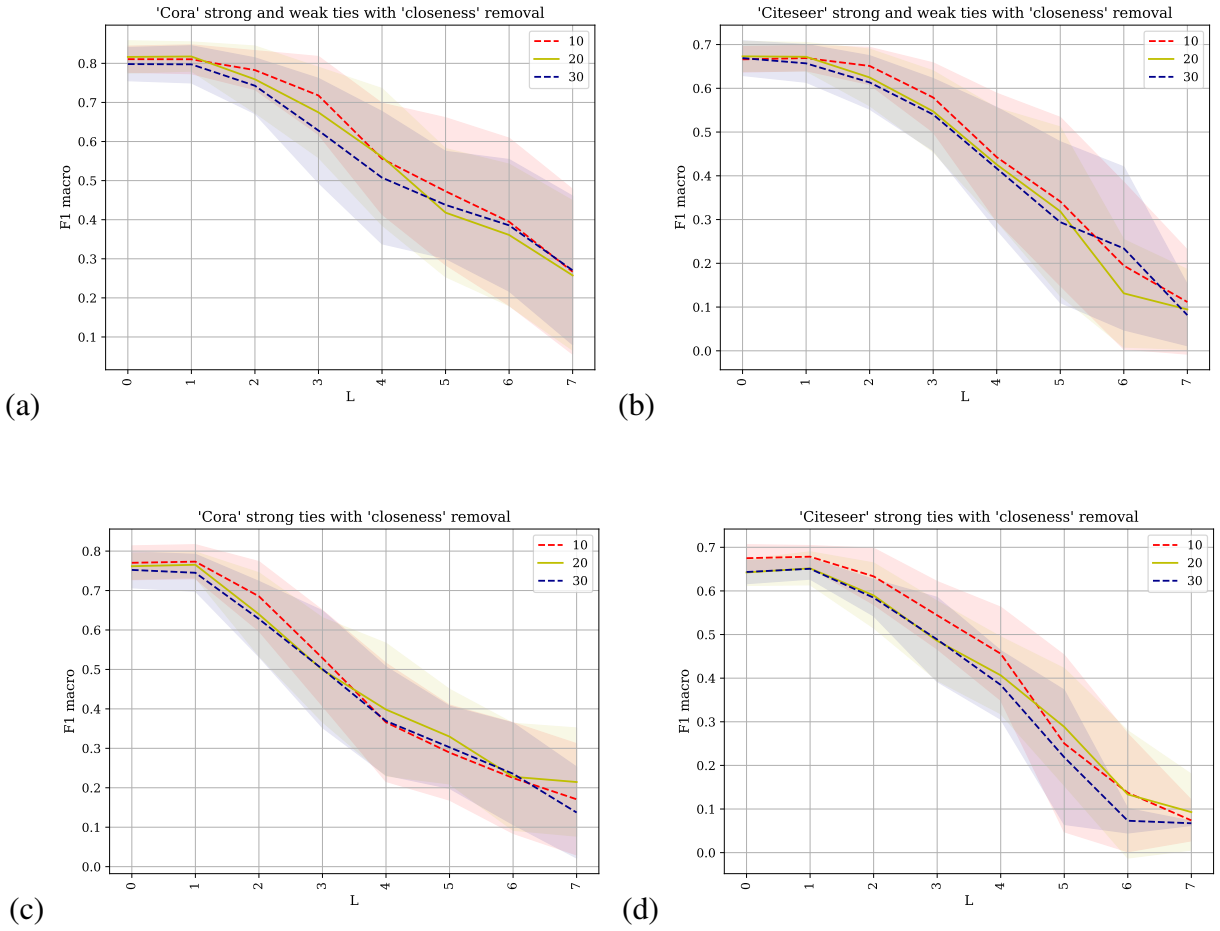


Figure A.21: The Graph Convolutional Networks (GCNs) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 macro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

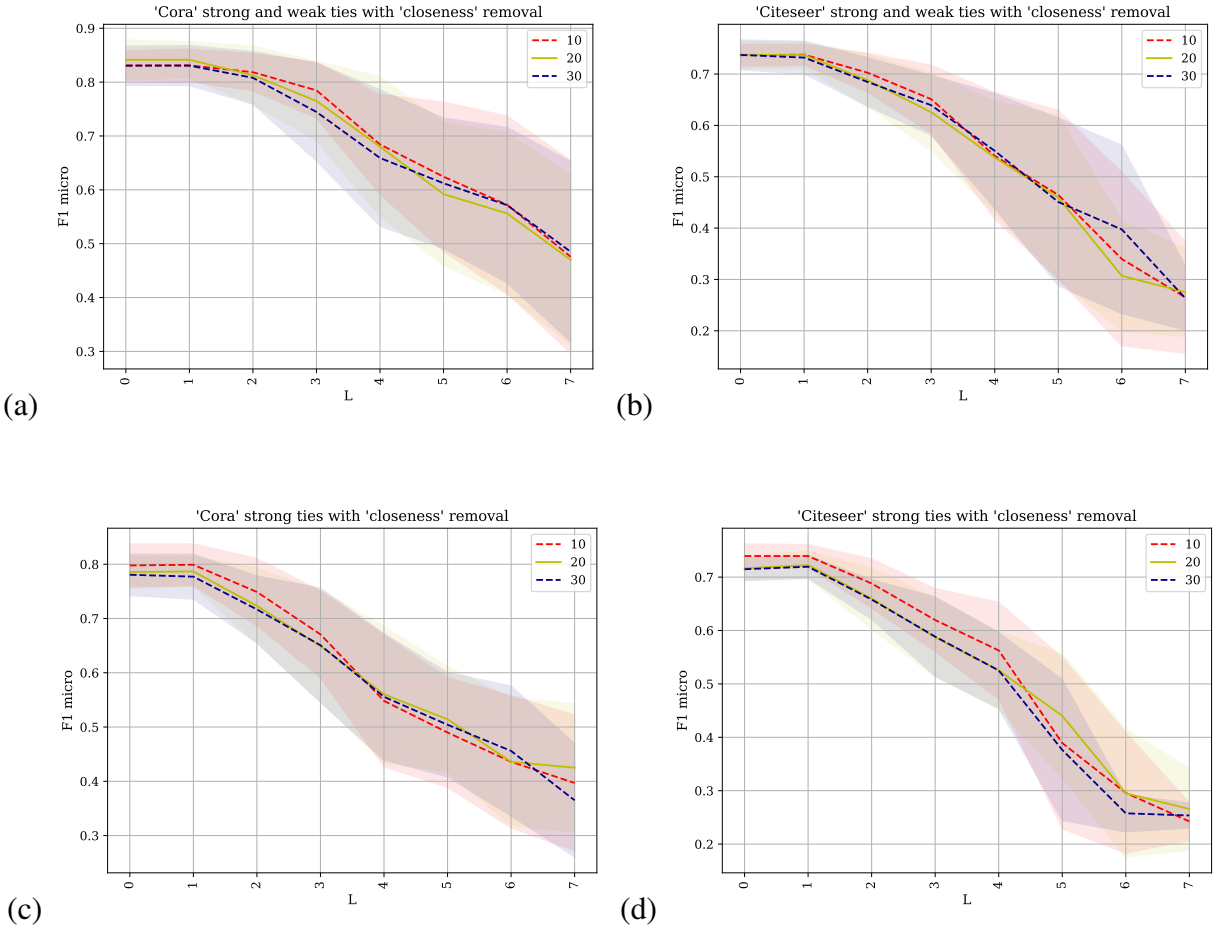


Figure A.22: The Graph Convolutional Networks (GCNs) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the F1 micro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

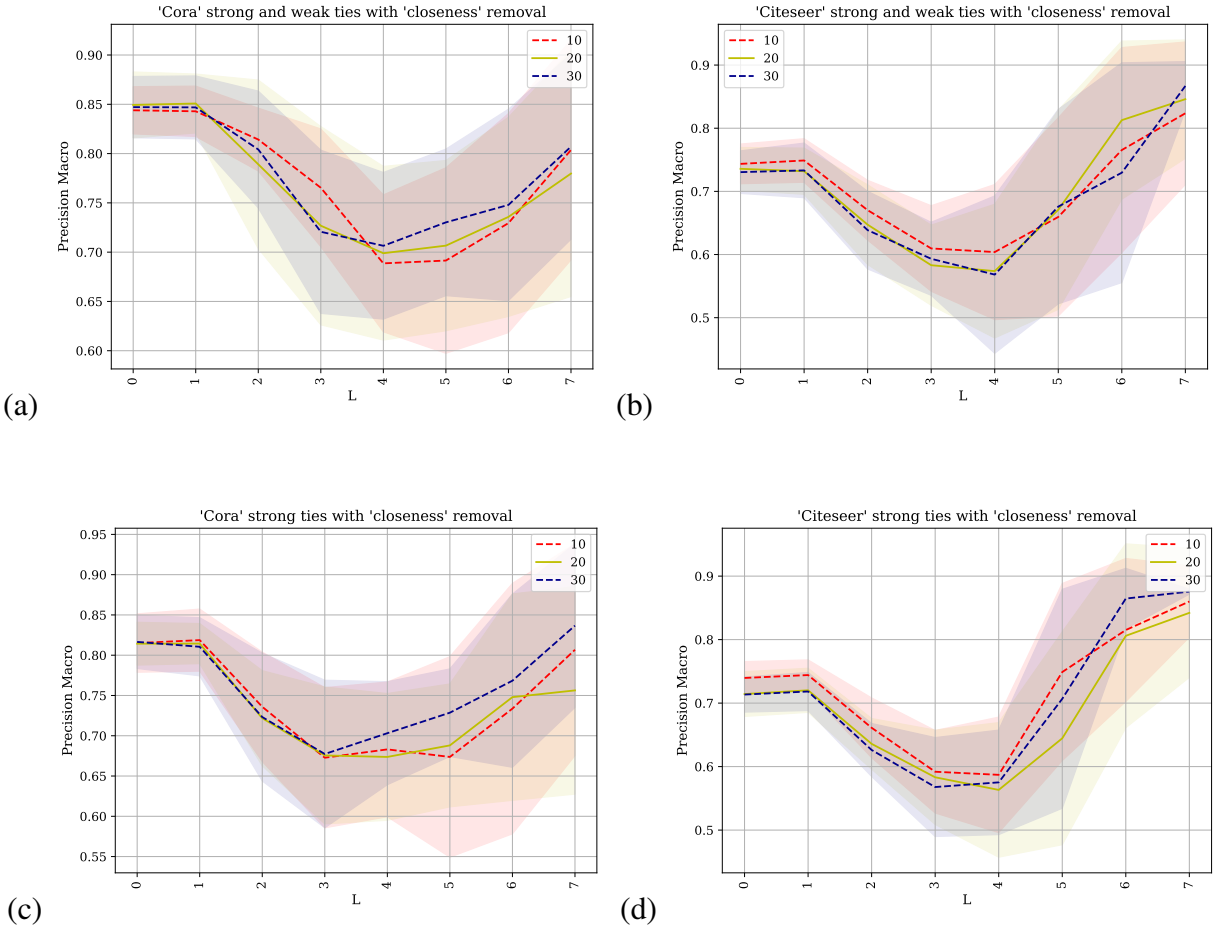


Figure A.23: The Graph Convolutional Networks (GCNs) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision macro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

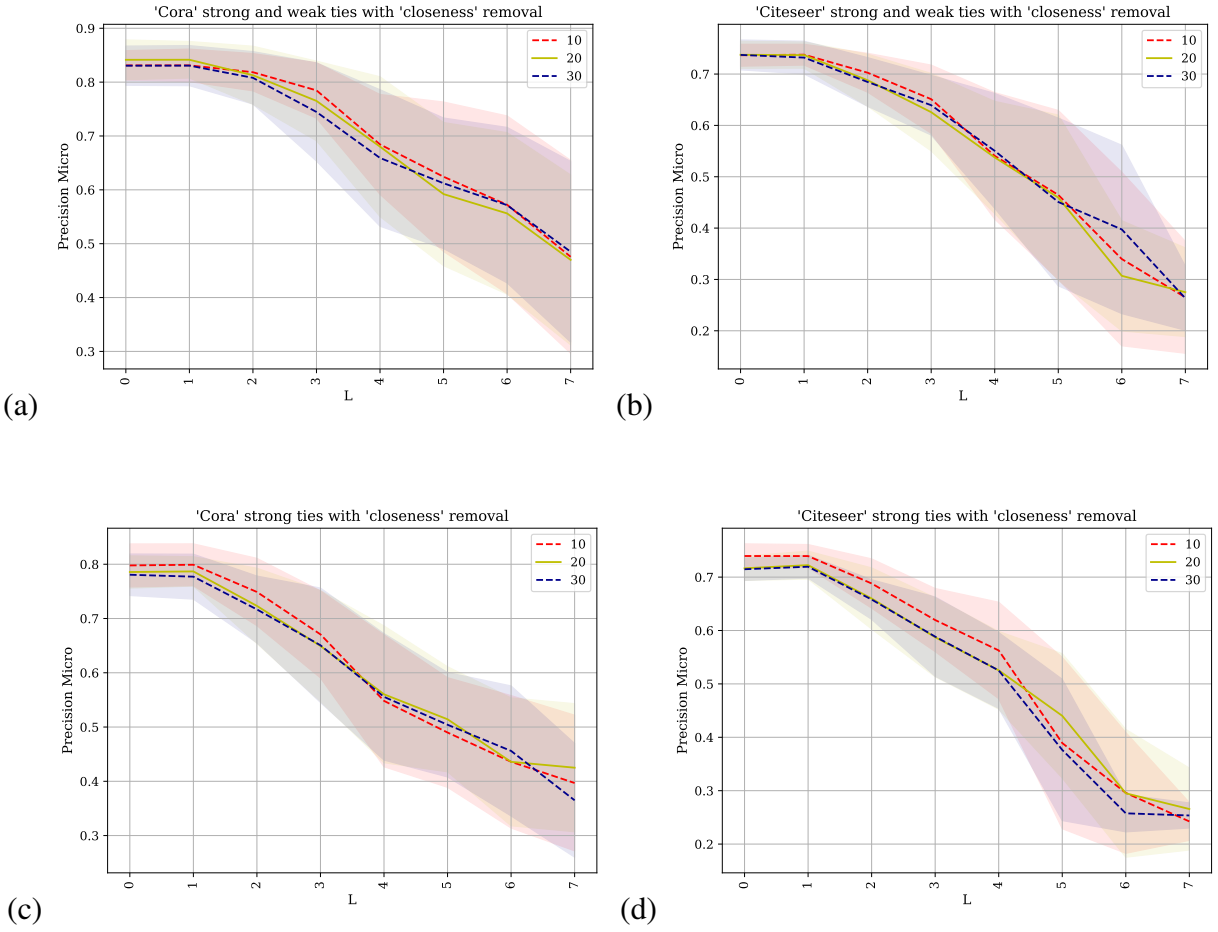


Figure A.24: The Graph Convolutional Networks (GCNs) methodology was applied to predicting the class labels of the Cora and Citeseer datasets where the precision micro is plotted against the parameter l. The closeness metric is used to rank and remove different percentages of the network: (**a**) and (**b**) show the prediction changes when the network consists of *strong-ties* and *weak-ties*, and (**c**) and (**d**) show the results when the original adjacency matrix containing only *strong-ties* is used.

LIST OF REFERENCES

[1] A. Mccallum, "Cora research paper classification dataset," *people. cs. umass. edu/mccallum/data. html. KDD*, 2001.

[2] J.-X. Zhang, D.-B. Chen, Q. Dong, and Z.-D. Zhao, "Identifying a set of influential spreaders in complex networks," *Scientific Reports*, vol. 6, p. 27823, 2016.

[3] C. Caragea, J. Wu, A. Ciobanu, K. Williams, J. Fernández-Ramírez, H.-H. Chen, Z. Wu, and L. Giles, "Citeseer x: A scholarly big dataset," in *European Conference on Information Retrieval*. Springer, 2014, pp. 311–322.

[4] "Twitter." [Online]. Available: https://en.wikipedia.org/wiki/Twitter

[5] D. Boyd, "Friends, friendsters, and myspace top 8: Writing community into being on social network sites," 2006.

[6] "Classmates classmates.com," http://www.classmates.com., accessed: 2019-06-15.

[7] "sixdegrees sixdegrees.com," http://www.sixdegrees.com, accessed: 2019-06-15.

[8] "Ryze,ryze.com," http://www.linkedin.com, accessed: 2019-06-15.

[9] "CyWorld cyworld.com," http://www.cyworld.com, accessed: 2019-06-15.

[10] "LinkedIn linkedin.com," http://www.linkedin.com, accessed: 2019-06-15.

[11] "livejournal livejournal.com," https://www.livejournal.com/site/about.bml, accessed: 2019-06-15.

[12] "zooomr zooomr.com," https:https://www.zooomr.com, accessed: 2019-06-15.

[13] "blogger blogger.com," https://www.blogger.com/about/?r=2, accessed: 2019-06-15.

[14] "digg digg.com," digg.com, accessed: 2019-06-15.

[15] A. Williams, "Github pours energies into enterprise–raises $100 million from power vc andreessen horowitz," *Tech Crunch. Andreessen Horowitz is investing an eye-popping $100 million into GitHub*, vol. 9, 2012.

[16] K. Davison, "Early modern social networks: Antecedents, opportunities, and challenges," *The American Historical Review*, vol. 124, no. 2, pp. 456–482, 2019.

[17] D. Boyd, "Friendster and publicly articulated social networks: Conference on human components and computing systems. vienna, austria," 2004.

[18] ——, "The role of networked publics in teenage social life," *Youth, identity, and digital media*, pp. 119–142, 2007.

[19] D. M. Boyd and N. B. Ellison, "Social network sites: Definition, history, and scholarship," *Journal of computer-mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[20] D. Boyd and J. Heer, "Profiles as conversation: Networked identity performance on friendster," in *Proceedings of the 39th annual Hawaii international conference on system sciences (HICSS'06)*, vol. 3. IEEE, 2006, pp. 59c–59c.

[21] K. M. Carley, *Dynamic network analysis*. na, 2003.

[22] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Social influence prediction with deep learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2110–2119.

[23] D. B. Kurka, A. Godoy, and F. J. Von Zuben, "Online social network analysis: A survey of research applications in computer science," *arXiv preprint arXiv:1504.05655*, 2015.

[24] J. Wang, C. Jiang, S. Guan, L. Xu, and Y. Ren, "Big data driven similarity based u-model for online social networks," in *IEEE Global Communications Conference*, 2017, pp. 1–6.

[25] L. A. Marascuilo and R. C. Serlin, *Statistical methods for the social and behavioral sciences.* WH Freeman/Times Books/Henry Holt & Co, 1988.

[26] T. A. Kohler and G. G. Gumerman, *Dynamics in human and primate societies: Agent-based modeling of social and spatial processes*. Oxford University Press, 2000.

[27] C. Castellano, S. Fortunato, and V. Loreto, "Statistical physics of social dynamics," *Reviews of modern physics*, vol. 81, no. 2, p. 591, 2009.

[28] A. Jedrzejewski and K. Sznajd-Weron, "Statistical physics of opinion formation: is it a spoof?" *Comptes Rendus Physique*, 2019.

[29] X. Yin, H. Wang, P. Yin, and H. Zhu, "Agent-based opinion formation modeling in social network: A perspective of social psychology," *Physica A: Statistical Mechanics and its Applications*, p. 121786, 2019.

[30] M. Lippe, M. Bithell, N. Gotts, D. Natalini, P. Barbrook-Johnson, C. Giupponi, M. Hallier, G. J. Hofstede, C. Le Page, R. B. Matthews *et al.*, "Using agent-based modelling to simulate social-ecological systems across scales," *GeoInformatica*, vol. 23, no. 2, pp. 269–298, 2019.

[31] X. Liu, D. He, and C. Liu, "Information diffusion nonlinear dynamics modeling and evolution analysis in online social network based on emergency events," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, pp. 8–19, 2019.

[32] L. A. Adamic, T. M. Lento, E. Adar, and P. C. Ng, "Information evolution in social networks," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016, pp. 473–482.

[33] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge & Data Engineering*, no. 6, pp. 734–749, 2005.

[34] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[35] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender systems handbook*. Springer, 2011, pp. 107–144.

[36] D. P. Heyman, A. Tabatabai, and T. Lakshman, "Statistical analysis and simulation study of video teleconference traffic in atm networks," *IEEE Transactions on circuits and systems for video technology*, vol. 2, no. 1, pp. 49–59, 1992.

[37] M. Šuvakov, D. Garcia, F. Schweitzer, and B. Tadić, "Agent-based simulations of emotion spreading in online social networks," *arXiv preprint arXiv:1205.6278*, 2012.

[38] U. Wilensky and W. Rand, *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*. MIT Press, 2015.

[39] M. Galesic and D. L. Stein, "Statistical physics models of belief dynamics: Theory and empirical tests," *Physica A: Statistical Mechanics and its Applications*, vol. 519, pp. 275–294, 2019.

[40] K. Jaidka, S. Ahmed, M. Skoric, and M. Hilbert, "Predicting elections from social media: a three-country, three-method comparative study," *Asian Journal of Communication*, vol. 29, no. 3, pp. 252–273, 2019.

[41] D. Li, Y. Wang, A. Madden, Y. Ding, J. Tang, G. G. Sun, N. Zhang, and E. Zhou, "Analyzing stock market trends using social media user moods and social influence," *Journal of the Association for Information Science and Technology*, 2019.

[42] C. W. Schmidt, "Trending now: using social media to predict and track disease outbreaks," 2012.

[43] Y. Zhou, L. Zhang, and Z. Yi, "Predicting movie box-office revenues using deep neural networks," *Neural Computing and Applications*, vol. 31, no. 6, pp. 1855–1865, 2019.

[44] H. Schoen, D. Gayo-Avello, P. Takis Metaxas, E. Mustafaraj, M. Strohmaier, and P. Gloor, "The power of prediction with social media," *Internet Research*, vol. 23, no. 5, pp. 528–543, 2013.

[45] F. Franch, "(wisdom of the crowds) 2: 2010 uk election prediction with social media," *Journal of Information Technology & Politics*, vol. 10, no. 1, pp. 57–71, 2013.

[46] H. Kaur and H. S. Pannu, "Blog response volume prediction using adaptive neuro fuzzy inference system," in *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2018, pp. 1–6.

[47] N. Kim, K. Lučivjanská, P. Molnár, and R. Villa, "Google searches and stock market activity: Evidence from norway," *Finance Research Letters*, vol. 28, pp. 208–220, 2019.

[48] I. Bordino, S. Battiston, G. Caldarelli, M. Cristelli, A. Ukkonen, and I. Weber, "Web search queries can predict stock market volumes," *PloS one*, vol. 7, no. 7, p. e40014, 2012.

[49] M. Eirinaki, J. Gao, I. Varlamis, and K. Tserpes, "Recommender systems for large-scale social networks: A review of challenges and solutions," 2018.

[50] R. H. Gálvez and A. Gravano, "Assessing the usefulness of online message board mining in automatic stock prediction systems," *Journal of Computational Science*, vol. 19, pp. 43–56, 2017.

[51] T. H. Nguyen, K. Shirai, and J. Velcin, "Sentiment analysis on social media for stock movement prediction," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9603–9611, 2015.

[52] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in GitHub teams," in *Proceedings of the Annual Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 3789–3798.

[53] K. Blincoe, F. Harrison, and D. Damian, "Ecosystems in GitHub and a method for ecosystem identification using reference coupling," in *Proceedings of the Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 202–207.

[54] L. Singer, F. Figueira Filho, B. Cleary, C. Treude, M.-A. Storey, and K. Schneider, "Mutual assessment in the social programmer ecosystem: an empirical investigation of developer profile aggregators," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 2013, pp. 103–116.

[55] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stack overflow and github: Associations between software development and crowdsourced knowledge," in *International Conference on Social Computing*. IEEE, 2013, pp. 188–195.

[56] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the International Conference on Software Engineering*. ACM, 2014, pp. 345–355.

[57] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in GitHub," in *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 155–165.

[58] B. Vasilescu, V. Filkov, and A. Serebrenik, "Perceptions of diversity on GitHub: A user survey," in *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press, 2015, pp. 50–56.

[59] K. Aggarwal, A. Hindle, and E. Stroulia, "Co-evolution of project documentation and popularity within GitHub," in *Proceedings of the Working Conference on Mining Software Repositories*. ACM, 2014, pp. 360–363.

[60] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in GitHub: the role of prior social links and language experience," in *Proceedings of the Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 817–828.

[61] D. M. Soares, M. L. de Lima Júnior, A. Plastino, and L. Murta, "What factors influence the reviewer assignment to pull requests?" *Information and Software Technology*, vol. 98, pp. 32–43, 2018.

[62] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 2012, pp. 1277–1286.

[63] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: activity traces and personal profiles in Github," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 2013, pp. 117–128.

[64] B. Vasilescu, A. Serebrenik, and V. Filkov, "A data set for social diversity studies of GitHub teams," in *Proceedings of the Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 514–517.

[65] A. Al-Rubaye and G. Sukthankar, "A popularity-based model of the diffusion of innovation on GitHub," in *Proceedings of the Computational Social Science Society of the Americas*, Santa Fe, NM, 2018.

[66] B. Wu, W.-H. Cheng, Y. Zhang, Q. Huang, J. Li, and T. Mei, "Sequential prediction of social media popularity with deep temporal context networks," in *Proceedings of the International Joint Conference on Artificial Intelligence*, July 2017.

[67] Q. Zhao, M. Erdogdu, H. He, A. Rajaraman, and J. Leskovec, "Seismic, a self-exciting point process model for predicting tweet popularity," in *Proceedings of KDD*, 2015.

[68] T. L. Cheng-yu, M.-m. Poo, and Y. Dan, "Burst spiking of a single cortical neuron modifies global brain state," *Science*, vol. 324, no. 5927, pp. 643–646, 2009.

[69] A. Friggeri, L. Adamic, D. Eckles, and J. Cheng, "Rumor cascades," in *International AAAI Conference on Weblogs and Social Media*, 2014.

[70] S. A. Myers and J. Leskovec, "The bursty dynamics of the Twitter information network," in *Proceedings of the International Conference on World Wide Web*. ACM, 2014, pp. 913–924.

[71] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini, "A peek into the future: Predicting the evolution of popularity in user generated content," in *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 2013, pp. 607–616.

[72] C. Bauckhage, K. Kersting, and F. Hadiji, "Mathematical models of fads explain the temporal dynamics of internet memes," in *International AAAI Conference on Weblogs and Social Media*, 2013.

[73] Y. Matsubara, Y. Sakurai, B. A. Prakash, L. Li, and C. Faloutsos, "Rise and fall patterns of information diffusion: model and implications," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2012, pp. 6–14.

[74] J. Yang and J. Leskovec, "Modeling information diffusion in implicit networks," in *IEEE International Conference on Data Mining*. IEEE, 2010, pp. 599–608.

[75] J. Cheng, L. A. Adamic, J. M. Kleinberg, and J. Leskovec, "Do cascades recur?" in *Proceedings of the International Conference on World Wide Web*, 2016, pp. 671–681.

[76] S. Kwon, M. Cha, K. Jung, W. Chen, and Y. Wang, "Prominent features of rumor propagation in online social media," in *International Conference on Data Mining*. IEEE, 2013, pp. 1103–1108.

[77] L. Wu, P. Sun, R. Hong, Y. Fu, X. Wang, and M. Wang, "Socialgcn: An efficient graph convolutional network based model for social recommendation," *arXiv preprint arXiv:1811.02815*, 2018.

[78] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[79] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[80] J. Amin, M. Sharif, M. Yasmin, and S. L. Fernandes, "Big data analysis for brain tumor detection: Deep convolutional neural networks," *Future Generation Computer Systems*, vol. 87, pp. 290–297, 2018.

[81] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[82] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks

and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[83] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[84] F. P. Such, S. Sah, M. A. Dominguez, S. Pillai, C. Zhang, A. Michael, N. D. Cahill, and R. Ptucha, "Robust spatial filtering with graph convolutional neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 884–896, 2017.

[85] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[86] M. S. Granovetter, "The strength of weak ties," in *Social Networks*. Elsevier, 1977, pp. 347–367.

[87] E. Patacchini and Y. Zenou, "The strength of weak ties in crime," *European Economic Review*, vol. 52, no. 2, pp. 209–236, 2008.

[88] M. Ruef, "Strong ties, weak ties and islands: structural and cultural predictors of organizational innovation," *Industrial and Corporate Change*, vol. 11, no. 3, pp. 427–449, 2002.

[89] J. D. Montgomery, "Job search and network composition: Implications of the strength-of-weak-ties hypothesis," *American Sociological Review*, pp. 586–596, 1992.

[90] L. Ryan, "Looking for weak ties: using a mixed methods approach to capture elusive connections," *The Sociological Review*, vol. 64, no. 4, pp. 951–969, 2016.

[91] M. D. Conover, J. Ratkiewicz, M. Francisco, B. Gonçalves, F. Menczer, and A. Flammini, "Political polarization on twitter," in *International Conference on Weblogs and Social Media*, 2011.

[92] M. P. Fiorina and S. J. Abrams, "Political polarization in the american public," *Annual Review Political Science*, vol. 11, pp. 563–588, 2008.

[93] R. Fan, K. Xu, and J. Zhao, "Weak ties strengthen anger contagion in social media," *arXiv preprint arXiv:2005.01924*, 2020.

[94] G. M. Sandstrom and E. W. Dunn, "Social interactions and well-being: The surprising power of weak ties," *Personality and Social Psychology Bulletin*, vol. 40, no. 7, pp. 910–922, 2014.

[95] I. Dagan, L. Lee, and F. Pereira, "Similarity-based methods for word sense disambiguation," *arXiv preprint cmp-lg/9708010*, 1997.

[96] G. Marsaglia, W. W. Tsang, J. Wang *et al.*, "Evaluating kolmogorov's distribution," *Journal of Statistical Software*, vol. 8, no. 18, pp. 1–4, 2003.

[97] R. Carpenter, "Principles and procedures of statistics, with special reference to the biological sciences," *The Eugenics Review*, vol. 52, no. 3, p. 172, 1960.

[98] N. H. Bidoki, M. Schiappa, G. Sukthankar, and I. Garibay, "Modeling social coding dynamics with sampled historical data," *Online Social Networks and Media*, vol. 16, p. 100070, 2020.

[99] I. Garibay, T. A. Oghaz, N. Yousefi, E. C. Mutlu, M. Schiappa, S. Scheinert, G. C. Anagnostopoulos, C. Bouwens, S. M. Fiore, A. Mantzaris *et al.*, "Deep agent: Studying the dynamics of information spread and evolution in social networks," *arXiv preprint arXiv:2003.11611*, 2020.

[100] I. Garibay, M. Schiappa, G. C. Anagnostopoulos, C. Bouwens, S. Fiore, H. Keathley, A. Mantzaris, J. T. Murphy, W. Rand, A. Salter, M. Stanfill, G. Sukthankar, N. Baral, A. Berea, C. Gunaratne, N. B. Hajiakhoond, G. Fair, J. Jasser, C. Jayalath, E. Mutlu, O. Newton, A. Rajabi, S. Saadat, S. R. Scheinert, C. Senevirathna, and X. Winter,

Rachel Zhang, "Deep agent: Computational social science centered simulation of online information environments," in *Proceedings of International Conference on Computational Social Science*, 2019.

[101] Q. F. Ying, D. M. Chiu, S. Venkatramanan, and X. Zhang, "Profiling OSN users based on temporal posting patterns," in *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18, 2018, pp. 1451–1456.

[102] Y. Yu, G. Yin, H. Wang, and T. Wang, "Exploring the patterns of social behavior in GitHub," in *Crowdsoft*, 11 2014, pp. 31–36.

[103] C. Wang, W. Tang, B. Sun, J. Fang, and Y. Wang, "Review on community detection algorithms in social networks," in *IEEE International Conference on Progress in Informatics and Computing (PIC)*, 2015, pp. 551–555.

[104] F. Thung, T. F. Bissyande, D. Lo, and L. Jiang, "Network structure of social coding in github," in *2013 17th European Conference on Software Maintenance and Reengineering*. IEEE, 2013, pp. 323–326.

[105] J. Jiang, L. Zhang, and L. Li, "Understanding project dissemination on a social coding site," in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 132–141.

[106] S. Harenberg, G. Bello, L. Gjeltema, S. Ranshous, J. Harlalka, R. Seay, K. Padmanabhan, and N. Samatova, "Community detection in large-scale networks: a survey and empirical evaluation," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 426–439, 2014.

[107] M. Wang, C. Wang, J. X. Yu, and J. Zhang, "Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 998–1009, 2015.

[108] A.-L. Barabási, *Bursts: the hidden patterns behind everything we do, from your e-mail to bloody crusades*. Penguin, 2010.

[109] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, p. 066111, 2004.

[110] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, pp. 2065–2073, 2014.

[111] D. Sun, T. He, and F. Zhang, "Survey of cold-start problems in collaborative filtering recommender system," *Computer and Modernization*, vol. 5, pp. 59–63, 2012.

[112] N. Hajiakhoond Bidoki, S. Madeline, S. Gita, and I. Garibay, "Predicting social network evolution from community data partitions," in *International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction and Behavior Representation in Modeling and Simulation*, 2019.

[113] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing letters*, vol. 12, no. 3, pp. 211–223, 2001.

[114] I. Garibay, M. Schiappa, G. C. Anagnostopoulos, C. Bouwens, S. Fiore, H. Keathley, A. Mantzaris, J. T. Murphy, W. Rand, A. Salter, M. Stanfill, G. Sukthankar, N. Baral, A. Berea, C. Gunaratne, N. B. Hajiakhoond, G. Fair, J. Jasser, C. Jayalath, E. Mutlu, O. Newton, A. Rajabi, S. Saadat, S. R. Scheinert, C. Senevirathna, and X. Winter,

Rachel Zhang, "Deep agent: Computational social science centered simulation of online information environments," 2019.

[115] H. U. Sheikh and L. Bölöni, "Multi-agent reinforcement learning for problems with combined individual and team reward," *arXiv preprint arXiv:2003.10598*, 2020.

[116] ——, "Emergence of scenario-appropriate collaborative behaviors for teams of robotic bodyguards," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019, pp. 2189–2191.

[117] J. Yang and S. Counts, "Predicting the speed, scale, and range of information diffusion in twitter," in *AAAI Conference on Weblogs and Social Media*, 2010.

[118] N. Friedkin, "A test of structural features of granovetter's strength of weak ties theory," *Social Networks*, vol. 2, no. 4, pp. 411–422, 1980.

[119] J. Zhao, J. Wu, and K. Xu, "Weak ties: Subtle role of information diffusion in online social networks," *Physical Review E*, vol. 82, no. 1, p. 016105, 2010.

[120] P. Mooney, A. C. Winstanley, and P. Corcoran, "Evaluating twitter for use in environmental awareness campaigns," in *Winstanley, Adam (Hg.): Proceedings of the China-Ireland Information and Communications Technologies Conference (CIICT 2009)*, 2009, pp. 83–86.

[121] M. Del Vicario, A. Bessi, F. Zollo, F. Petroni, A. Scala, G. Caldarelli, H. E. Stanley, and W. Quattrociocchi, "The spreading of misinformation online," *Proceedings of the National Academy of Sciences*, vol. 113, no. 3, pp. 554–559, 2016.

[122] C. Taylor, A. Mantzaris, and I. Garibay, "Exploring how homophily and accessibility can facilitate polarization in social networks," *Information*, vol. 9, no. 12, p. 325, 2018.

[123] K. Starbird and L. Palen, "(how) will the revolution be retweeted?: Information diffusion and the 2011 egyptian uprising," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM, 2012, pp. 7–16.

[124] A.-L. Barabasi, "The origin of bursts and heavy tails in human dynamics," *Nature*, vol. 435, no. 7039, p. 207, 2005.

[125] A. V. Mantzaris, "Uncovering nodes that spread information between communities in social networks," *EPJ Data Science*, vol. 3, no. 1, p. 26, 2014.

[126] N. Hajiakhoond Bidoki, A. V. Mantzaris, and G. Sukthankar, "An lstm model for predicting cross-platform bursts of social media activity," *Information*, vol. 10, no. 12, p. 394, 2019.

[127] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[128] S. Wang, Z. Yan, X. Hu, S. Y. Philip, and Z. Li, "Burst time prediction in cascades," in *AAAI Conference on Artificial Intelligence*, 2015.

[129] D. Higham, A. V. Mantzaris, P. Grindrod, A. Otley, and P. Laflin, "Anticipating activity in social media spikes," in *International AAAI Conference on Web and Social Media*, 2015.

[130] N. Alsaedi, P. Burnap, and O. Rana, "Automatic summarization of real world events using twitter," in *Tenth International AAAI Conference on Web and Social Media*, 2016.

[131] J. Xu and B. Livshits, "The anatomy of a cryptocurrency pump-and-dump scheme," in *USENIX Security Symposium*, 2019, pp. 1609–1625.

[132] J. Kleinberg, "Bursty and hierarchical structure in streams," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 373–397, 2003.

[133] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[134] M. Schiappa, G. Chantry, and I. Garibay, "Cyber security in a complex community: A social media analysis on common vulnerabilities and exposures," 2019.

[135] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *Proceedings of the International Conference on World Wide Web*, 2010, pp. 591–600.

[136] V. Wadhwa, E. Latimer, K. Chatterjee, J. McCarty, and R. Fitzgerald, "Maximizing the tweet engagement rate in academia: analysis of the ajnr twitter feed," *American Journal of Neuroradiology*, vol. 38, no. 10, pp. 1866–1868, 2017.

[137] X. Shuai, A. Pepe, and J. Bollen, "How the scientific community reacts to newly submitted preprints: Article downloads, Twitter mentions, and citations," *PloS One*, vol. 7, no. 11, 2012.

[138] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[139] Y. Ding, E. Yan, A. Frazho, and J. Caverlee, "Pagerank for ranking authors in co-citation networks," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 11, pp. 2229–2243, 2009.

[140] Y. Ding, "Scientific collaboration and endorsement: Network analysis of coauthorship and citation networks," *Journal of Informetrics*, vol. 5, no. 1, pp. 187–203, 2011.

[141] Y. Bian, "Bringing strong ties back in: Indirect ties, network bridges, and job searches in china," *American Sociological Review*, pp. 366–385, 1997.

[142] A. V. Mantzaris and D. J. Higham, "Infering and calibrating triadic closure in a dynamic network," in *Temporal Networks*. Springer, 2013, pp. 265–282.

[143] V. Roux, B. Bril, and A. Karasik, "Weak ties and expertise: Crossing technological boundaries," *Journal of Archaeological Method and Theory*, vol. 25, no. 4, pp. 1024–1050, 2018.

[144] F. Ghaffar, T. S. Buda, H. Assem, A. Afsharinejad, and N. Hurley, "A framework for enterprise social network assessment and weak ties recommendation," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2018, pp. 678–685.

[145] X. Wang and G. Sukthankar, "Multi-label relational neighbor classification using social context features," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Chicago, IL, August 2013, pp. 464–472.

[146] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[147] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[148] W. Samek, *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer Nature, 2019, vol. 11700.

[149] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[150] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.

[151] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *arXiv preprint arXiv:1812.04202*, 2018.

[152] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma *et al.*, "Deep graph library: Towards efficient and scalable deep learning on graphs," *arXiv preprint arXiv:1909.01315*, 2019.

[153] M. T. Angulo, G. Lippner, Y.-Y. Liu, and A.-L. Barabási, "Sensitivity of complex networks," *arXiv preprint arXiv:1610.05264*, 2016.

[154] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[155] P. Grindrod, M. C. Parsons, D. J. Higham, and E. Estrada, "Communicability across evolving networks," *Physical Review E*, vol. 83, no. 4, p. 046120, 2011.