Finding Event-Specific Influencers in Dynamic Social Networks

by

Christopher Brendan Schenk

B.S. Computer Science, University of Colorado, 2004

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2010

This thesis entitled:
Finding Event-Specific Influencers in Dynamic Social Networks
written by Christopher Brendan Schenk
has been approved for the Department of Computer Science

Douglas Sicker (chair)

Qin Lv

Aaron Clauset

Date ____________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Schenk, Christopher Brendan (M.S., Computer Science)

Finding Event-Specific Influencers in Dynamic Social Networks

Thesis directed by Professor Douglas Sicker (chair)

Reputation models are widely in use today in commercial transaction (ebay), product review (amazon, epinions), and news commentary websites (slashdot). The purpose of these reputation models is to provide behavioral or informational data for future users to determine whether or not he or she will trust the data. These models are dependent on explicit feedback mechanisms where users rate product, other users, or information. However, for many popular social network information sources on the web, no such explicit feedback systems exist where users rate information in order for consumers of this information to be able to judge the trustworthiness of the data source or the data itself.

Here I describe the layers of the problem of determining reputation among users or data during events discussed on social networks, and evaluate data and network analysis methods from varying disciplines that may implicitly infer user or data reputation based on metadata, user relationships and user actions in social networks. I demonstrate that the HITS algorithm is not effective at finding influential users, and propose a new algorithm and demonstrate its effectiveness for finding influential users during an event.

Dedication

I want to dedicate this work to my parents who believe education is the foundation for success, as I would not be here without their sacrifices for my brothers and sister and myself throughout our lifetimes, putting us through good schools and teaching us to never give up and to try, and try again.

Acknowledgements

There are many people who have contributed to my success in this process, and I don't know if I'd be able to remember every single one. First I want to thank my advisor Doug Sicker for giving me unyielding support both psychologically and financially through this entire process, even when I didn't think I'd finish. I want to thank Aaron Clauset for giving me fantastic direction in the work as well, along with great conversations and wonderful new opportunities for work. Christine Lv is fabulous and kind, and has been a positive force through this process as well. I want to thank Ken Anderson, Leysia Palen, Jim Martin, Martha Palmer, Ban Al-Ani, Gloria Mark and all of their students Aaron Schram, Jo White, Kate Starbird, Sarah Vieweg, Sophia Liu, Casey McTaggart, Mossaab Bagdouri, Aleksandra Sarcevic, and Will Corvey. Their presence during this process has been invaluable for my own learning. I want to thank the rest of the faculty and staff of the Computer Science department and for my opportunity to work there as I would not have chosen to return to school otherwise. I want to thank my parents for their dedication to putting us through good schools so my brothers and sisters would have the opportunity to be successful in many different ways, and I also want to thank them for their love and moral support through this entire process as well. I want to thank Cassie Houtz for her moral and emotional support as well as some help in understanding some of the dense papers I had to read during the process. Your presence was also invaluable.

Thank you all.

Contents

Tables

Figures

Chapter 1

Introduction

Early commercial uses of the internet revolved around online retail sales with sites like Amazon[1] and the popular auctioneering site EBay.[2] The problem faced by early users of these sites was a *lack* of information about goods sold on the retail sites (Is this book or audio CD any good?) or behavioral history of users in auctions (Does this user pay and deliver quickly?). Explicit reputation systems were developed by these sites and others with similar needs in order to rate users' behavior and record transaction history.

On the internet today, nearly fifteen years after Amazon and EBay began their businesses, there is no end to the amount of information one is able to encounter. Google's page index count reached one trillion pages in July of 2008 [46] and continues to grow, and with the abundance of information online – whether that information is in news articles, blog posts, wikipedia entries, forums, or random messages on social networking sites like Twitter or Facebook – the task of sorting through that information to determine relevance or validity of claims requires significant effort on the part of the consumer of that information. In addition to the numerous amateur and professional news websites, many content management systems (CMS) such as Wordpress[3] are freely available to allow the average home internet user to create content to be posted publicly.

Some studies into user generated content (UGC) reveal an immense amount of information is both created and shared daily via different online services, including 65,000 new video uploads

[1] http://www.amazon.com
[2] http://www.ebay.com
[3] http://www.wordpress.com

per day on the YouTube[4] service [20], and over 11.1 million unique photos favorited over 34.7 million times on the Flickr[5] photo sharing service [21]. Other work discusses people's desire to share personal information or opinions in blogs [83], in small microupdates in Twitter [57] [68], and in wall posts on Facebook and other sites [79] [23].

One of the main issues with this abundance of information from both organizations and individuals with varying credentials is that users of the internet are inundated with too much information. Sorting through this large volume of information to verify or dispute user posts or news articles becomes a very difficult task, especially considering the variability and often conflicting nature of information online. Internet users will attempt to verify or dispute claims through their own searches, through their own experience [59], or make a decision based solely on content [35]. Similar to the need for explicit rating systems described by online commerce, a need now exists to determine the reputation for people and information online that are *not* explicitly rated.

I seek to answer the question, "are there tools available to determine influential users in social networks in the context of a specific event?" This question is the first sub-problem of the larger question, "are there tools available to infer reputation about users or content in social networks?" In this research, I investigate different tools and analysis methods gathered from varying science disciplines that may be employed to analyze social network content and relationships. Tools and analysis methods broadly include statistics, probability, graph theory, natural language processing, and others.

The outline of this research is as follows: In chapter 2 I outline the overall problem space of inferring reputation, explore the definitions of trust and reputation, summarize prior work in the area of reputation, and discuss goals for this research. In chapter 3 I review many statistical, graph, social network and natural language processing analysis tools and methods and discuss their potential applications for inferring reputation. In chapter 4 I describe the data I obtain from the Twitter[6] social network and limitations therein, as well as many statistics on the data returned.

[4] http://www.youtube.com
[5] http://www.flickr.com
[6] http://twitter.com

Chapters 5 and 6 describe the implementation and analysis of the Hyperlink-Induced Topic Search (HITS) algorithm and a new dynamic algorithm for determining influencers in the network during an event, respectively. In chapter 7 I discuss conclusions and future work to be pursued.

Chapter 2

Influencers, Reputation, Validation, and Security

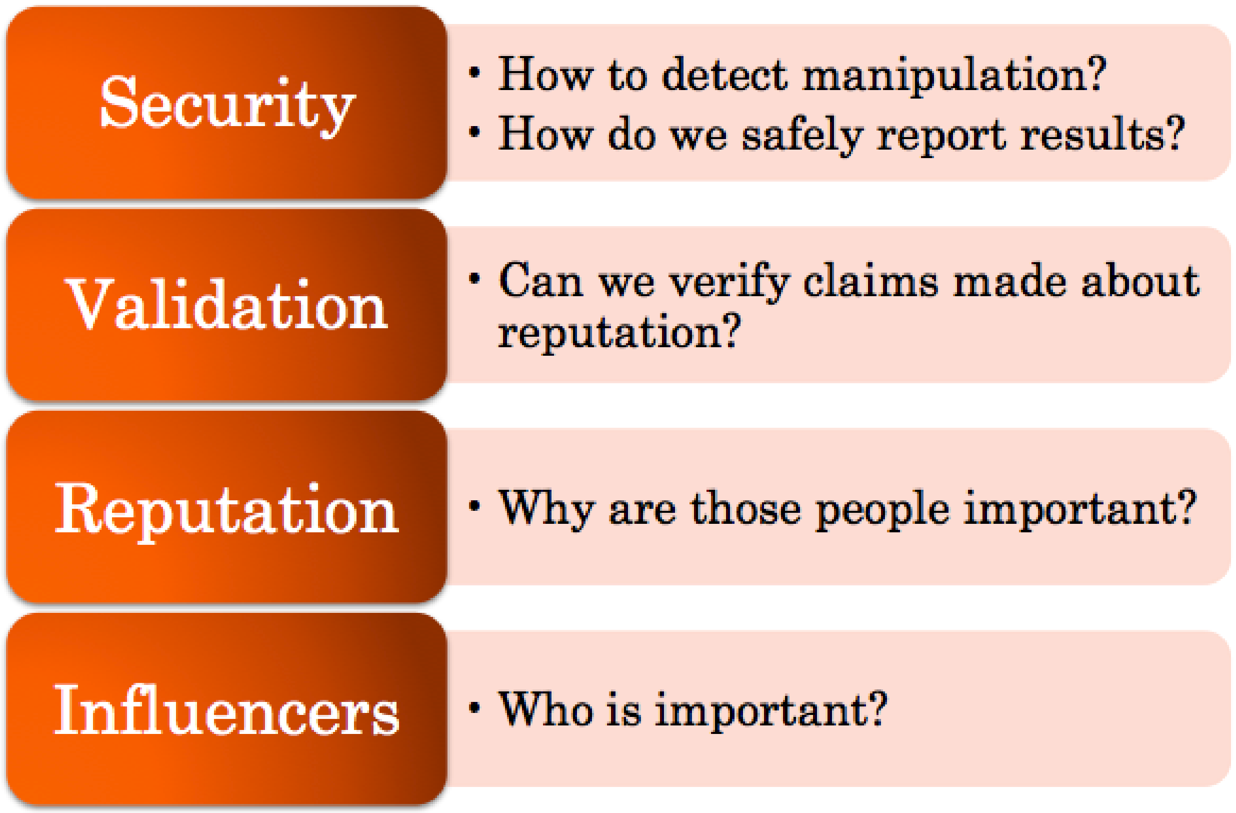
In this chapter I discuss the dependencies and layers of determining a user's reputation in the context of an event. Layers are not necessarily independent of each other; some methods employed to solve a problem at one layer may also inform problems at another layer. We review research in area of finding influencers in multiple social contexts and follow with a review of different definitions of Trust and Reputation that will inform my research direction. We then discuss validation of a user's reputation as the third layer in answering the high-level research question described in the introduction. Finally I discuss security implications of systems implemented to answer the research questions.

Figure 2.1 shows the four layers of the problem with associated questions within each layer. The layers are foundational, and must be approached from bottom-to-top, beginning with finding influencers and moving on to determining reputation, validating that reputation, and dealing with security implications of determining this information. These layers will now be described.

2.1 Influencers

Finding influencers in a network during an event is a first step in determining reputation. Before we can look at reputation, We must first ask the question, "who is important in an online social network during a specific event being discussed within that network?" Intuition leads us to the fact that influential people are most likely the people who are interacting and communicating in a network or social group *and* who are also being listened to by many others. Entire fields

Figure 2.1: The four layers of the problem of determining a user's reputation, including questions for each layer. The layers are foundational and must be approached bottom-to-top.


Security
• How to detect manipulation?
• How do we safely report results?
Validation
• Can we verify claims made about reputation?
Reputation
• Why are those people important?
Influencers
• Who is important?

of study exist around dynamics of social groups and many have been studied in an attempt to understand influencers. Some examples include adolescent smoking initiation by Arnett [3], moral hypocrisy in power dynamics by Lammers *et al.* [70], centrality dynamics and delinquent influence in game-theoretic networks by Ballester *et al.* [8] [9], interpersonal interactions in a research team by Klemm *et al.* [66], interpersonal interactions in group discussion dynamics by Pan *et al.* [95], and notably Wasserman and Faust [121] with various techniques. However, many of these studies and techniques are centered around social dynamics governed by interaction in-person.

We want to understand how influencers emerge in online social networks where the social dynamics are governed by the features of the particular online social network such as Facebook[1] or Twitter[2] where the user interface has very specific behaviors that do not necessarily (and sometimes intentionally) correspond to in-person interactions. Here I describe studies of online social networks that attempt to look for influencers in these types of networks.

Ghosh and Lerman [41] analyze data collected from the Digg[3] news network of 3553 news stories in June 2009. They rank Digg users who posted a news story using many different ranking measures to determine which ranking system best models the empirical data from the resulting top Digg stories. The ranking models and algorithms user are closeness centrality, graph centrality, betweenness centrality (discussed in section 3.7), PageRank/eigenvector centrality (discussed in section 3.6), Hubbel's model, in-degree centrality, α-centrality, normalized α-centrality, Katz score, and SenderRank.

Gomez-Rodriguez *et al.* [44] analyze 170 million blog entries and news articles collected over a 1-year period between September 1, 2008 through August 31, 2009. They attempt to infer the diffusion network of how users are inter-connected by looking at how the information spreads through the blogs and news websites during the year. They use natural language processing techniques and use two network models (forest fire and kronecker) to determine the diffusion network.

Kitsak *et al.* [64] use k-shell decomposition (a variant of k-core decomposition described in

[1] http://www.facebook.com
[2] http://www.twitter.com
[3] http://digg.com

section 3.11) to determine influential actors in a network and apply the algorithm to LiveJournal[4] communities, email contacts, contact network of inpatients in Swedish hospitals, and movie actor co-star networks. They compare their results to results obtained from running the Susceptible-Infectious-Recovered (SIR) and Susceptible-Infectious-Susceptible (SIS) epidemiology models on the same data looking for those who have the most influence on the overall network. This method does not look at any dynamics of the network and how it might change.

Ormerod and Glass [91] attempt to predict what songs will gain the most popularity in a toy online music sharing network created by Salganik *et al.* [103]. Since the sharing data contains temporal information about when songs were downloaded, Ormerod and Glass are able to look at the dynamics of the network to predict what songs will become popular.

Some of the above studies look at the dynamic nature of the networks, and some do not. For my needs, I specifically need to look at dynamic methods or ways that the static methods may be applied to dynamic data to yield interesting results for finding influencers in the network during a specific event. New events have initial actors and new actors may be added as the event evolves, so the analysis methods must adapt to the changes in the underlying network. A number of other methods techniques that operate on the type of data that exists in online social networks will be described further in chapter 3 along with analysis of what data these methods may yield.

2.2 Trust and Reputation

Once I am able to determine the "who" of who's influential in during an event, I then ask the question, "why is a user influential in the context of an event?" As I attempt to answer this question, I begin to search for the source of a user's *reputation* within this context which is the second layer of the overall problem. Building a reputation is a dynamic process that occurs over time, and techniques to determine a user's reputation must also be adaptive and dynamic. Here I will describe the work in the area of reputation systems, investigate definitions of Trust and Reputation, and discuss what information is available for determining reputation in order to refine

[4] http://www.livejournal.com

my research goals.

2.2.1 Definitions

Many attempts have been made by different researchers to define the very subjective notions of Trust and Reputation. Vu *et al.* [120], Grabner-Kräuter *et al.* [47], and Jøsang *et al.* [59] each survey different definitions of Trust and Reputation by people from varying disciplines including Morton Deutsch (psychology), Niklas Luhmann (sociology), Diego Gambetta (sociology, economics) and McKnight and Chervany (management, business) in order to make sense of the variable nature of people's perceptions of the concepts. Herzig *et al.* [50] offer a logic-based definition of Trust and Reputation. We first look at reputation as it is defined for systems that have ratings that are made explicitly, or actions that can be measured in discrete values to be considered as explicit ratings.

Vu *et al.* [120] provide a breakdown of trust types, values, properties and models. They describe two trust types: trust in action, which is trust in the behavior of another agent, and trust in recommendation, which incorporates the group aspect of social relationships. Trust values may take on four different values: single, binary, multiple, or continuous. **Single** values are considered complaints where a transaction goes badly and an agent sends a claim to the system but does nothing upon a good transaction. Single value systems cannot differentiate between trusted and unknown agents. **Binary** values do allow values for trust and non-trust and an agent can distinguish between unknown and trusted agents, but unknown agents are neither trusted or distrusted. **Multiple** values allow an agent to specify varying levels of trust, such as "very untrustworthy", "somewhat untrustworthy", "somewhat trustworthy", and "very trustworthy". **Continuous** values are percentages of trust between 0 and 1.

Vu *et al* describe four properties of trust: autonomy, asymmetry, transitivity and composability. **Autonomy** as a property describes that trust depends on the perspective of the individual, and two individuals may have very differing trust views on the same subject. **Asymmetry** is the property where two individuals may not trust each other equally, depending on the context. In some cases, trust may be only in a single direction. **Transitivity** is the property where trust is conferred to an individual who is not directly trusted, but is trusted *by* an individual who is directly trusted. Trust transitivity is not perfect, and decays very quickly upon only a few transitive hops. **Composability** is the property of trust that is necessary in order to combine multiple trust values into a final conclusion. This property is not only limited to trust values stated by individuals, but also includes attributes of the trustee. For example, if readers are asked to rate a news article on how much they trust the article, extra information about the news agency (whether or not it is CNN or Fox News) may affect the ratings. The extra attributes are composed into a final conclusion.

Two trust models described by Vu *et al.* include *credential-based* and *reputation-based* trust. Credential-based trust relies on the authentication of an identity, which is used in many different systems for user logins such as online banking and email accounts). Many enhancements to simple username and password credentials have been proposed and studied, such as RSA keys, X.509 certificates, PGP keys, and others.

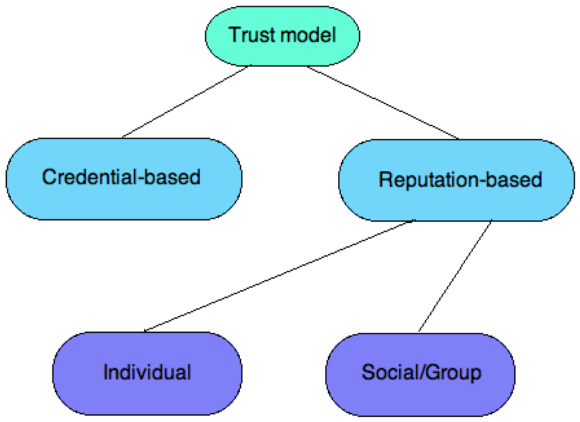
Reputation-based trust fills the gap where credentials are not enough to trust an individual and past actions are considered as part of the process of verifying trust. Reputation has both individual and social/group components. The individual component exists when an agent A only has knowledge of its own transactions with agent B, and the social/group component exists when agent A has knowledge of many if not all prior transactions of agent B. Figure 2.2 is a visualization of the trust model taxonomy.

Vu *et al.* propose a definition of Trust based on the reputation-based trust model described in the previous section: "*Trust* is the belief of an agent A about another agent B in the success of a transaction as a function of agent B's reputation based on B's history (either individual or social). If B's reputation is good, A trusts B. If B's reputation is bad, A should not trust B in the transaction."

2.2.2 Explicit Reputation Systems

Many systems currently exist to rate a user's reputation in specific contexts. When I describe *explicit* rating systems, I am speaking of systems that have well-defined trust attributes that can

Figure 2.2: A taxonomy of trust models, Vu *et al.* [120]



take on multiple values and be composed into a meaningful overall reputation score as described in section 2.2.1. As mentioned in the introduction, online commerce demands demonstrated the need for systems that track user behavior. That need has been studied numerous times, with solutions proposed to meet the need (either well or poorly) by a number of different systems. Below is a non-exhaustive list of reviews or proposals of varying types of explicit reputation computation models:

- Jøsang *et al.* [59] review definitions of trust and reputation, computation models (central or distributed systems using averages, bayesian, fuzzy calculations, etc), and commercial implementations of reputation systems such as Amazon, Epinions, Slashdot, Google PageRank and others,

- Vu *et al.* [120] also review definitions, types, properties and values of trust as well as survey multiple central and distributed computational reputation systems such as Regret, NodeRanking, P2PRep, and others,

- Abdul-Rahman and Hailes [1] propose a model with discrete categories of very trustworthy, trustworthy, untrustworthy, and very untrustworthy, using lookup tables to update reputation scores,

- Sabater and Sierra [102] propose a model to be used in transactions using fuzzy rules to obtain reputation scores, but do not attempt to detect malicious users,

- Guo and Kraines [48] also propose a model with discrete categories (high, moderate, low, unknown, dishonest), making probabilistic assumptions of how correctly users will recommend each other based on their rating, and calculate a transitive trust value for users with no prior history,

- Carbo *et al.* [17] and Sherchan *et al.* [107] propose methods based on fuzzy logic models for reasoning about user feedback,

- Kuter and Golbeck [69] propose a model that performs a probabilistic logic sampling over a Bayesian network to trust user recommendations only within high confidence intervals,
- Mislove *et al.* [80] propose a central token authentication system to combat spam from different sources (web, email, mislabeled content, and others) by leveraging trust relationships and degree via social links,
- Kamvar *et al.* [61] propose a distributed model for managing reputation in peer-to-peer networks, attempting to prevent malicious users from injecting false reputation values into the network, and weighting long-lived users more strongly than newcomers, and
- Caverlee *et al.* [19] propose the SocialTrust framework which is an explicit feedback mechanism for communities within MySpace.

2.2.3 Implicit Rating Systems (Client)

The above explicit reputation systems do serve specific purposes, but in the context of many online social networks or other information online, no means exist to explicitly rate people or information using defined trust values that can be composed into a meaningful rating, so we must look at implicit means of measuring and rating trust of people or information online.

Nichols [89] describes implicit data – data generated by users interacting with a system – as usable for determining implicit ratings. He continues to state that implicit ratings overall have less value than ratings generated from explicit systems (such as those mentioned in the previous section), but that implicit ratings may be much more numerous. He describes many examples of data that can be used for implicit ratings. Some of those examples are:

- Reference - User cites or refers to an item
- Saving - An item is favorited or saved for later use
- Examination - How long does a user look at an item
- Removal - What information is deleted or removed by a user

Research into implicit rating systems typically involves monitoring users at their source clients. Examples of monitoring users includes White *et al.* [125] who create a web search application (utilizing existing search engines at the time) to group search results based on user interaction with the search over time. They closely monitor how users interact with each search result and re-order pages and modify summary snippets accordingly. Two more examples of client-side user monitoring include Claypool *et al.* [26] who modify a web browser to monitor exactly how a user traverses web links from search results to determine relevance of pages based based on how far links are followed and Joachims *et al.* [58] combine an eye-tracking device to monitor how far down a user reads a document and an HTTP proxy to monitor the time between page requests in order to determine utility of web content to the user.

2.2.4 Implicit Rating Systems (Server)

The problem with applying implicit monitoring systems to social networks such as the ones described in the previous section is we rarely have access to client-side information about what links the user follows or how long a user remains reading a piece of information. Web server logs are also not accessible, and as anonymous researchers we only have access to the public Application Programming Interfaces (APIs) that have varying restrictions. As an example, the Twitter service does have a public API[5] which allows us to see only the final results of explicit actions taken by users, such as posting a message or creating a social link with other users. This significant limitation requires us to find tools or analysis methods that rely on observable user activity in social networks to potentially infer reputation on a user or information generated by that user.

We describe varying methods in chapter 3 that may be employed to determine implicit ratings of a user that may lead to eventual calculation of a reputation score. In chapter 4 I describe the data available as well as limitations accessing that data from the Twitter social network used in this research.

[5] http://dev.twitter.com

2.3 Validation

The third layer of the overall research problem of determining reputation of a user includes validation of claims made by users about information or other users in the network. While this is an important piece of the larger problem and arguably the most desirable for which to find a solution, this thesis does not focus on this part of the overall research question. We will briefly discuss the area of validation here.

The process of validating content is a very difficult one and typically left to humans for processing of information. Data must be cross-correlated between sources to verify identities, content, intent, and many other attributes. Humphreys *et al.* [53] perform a qualitative analysis of how often users in the social network are releasing private information in their general tweet streams and find that 12.1% of tweets have a location and 22.7% have proper names included in the text. Further, Mendoza *et al.* [78] also qualitatively analyze tweets for intent and self-correction during the Chile earthquakes in 2010. Additional qualitative studies of the network include Starbird *et al.* [109] who analyze use of the Twitter service during the Red River floods in North Dakota and Canada during the 2009 season and Palen *et al.* [92] who analyze use of the Facebook[6] service during the Virginia Tech shootings of 2007. While the information gained in each of these studies is important, it is also very difficult for a computer to determine on its own.

In light of the need for human intervention or review, some larger organizations have been created to facilitate human review of data of varying kinds. The concept is termed "crowdsourcing" where a crowd of people is employed, usually as a call for volunteers, to analyze data. Examples of systems that exist for crowdsourcing include Amazon's Mechanical Turk[7] and the disaster visualization and coding system of Ushahidi.[8] Mechanical Turk may be utilized to enabled semantic coding of general data for its relevance and correctness for varying applications. Ushahidi is more specialized for disaster events of varying kinds.

[6] http://www.facebook.com
[7] http://www.mturk.com
[8] http://www.ushahidi.com

For the purposes of validating the reasons discovered for why influencers are influential, I foresee the need to have humans perform the validation of those reasons, or to dynamically inform algorithms or processes created to find those influencers or reasons. And while computing systems may be implemented to assist in narrowing the scope of the information reviewed by humans, but the need will always exist to have people reviewing data. The groups of people may simply be public health and safety officials or those similar to Ushahidi or Mechanical Turk.

2.4 Security

The final layer of the overall research problem lies in reporting of any of the data gained from analysis from the previous three layers: finding influencers, understanding why those users are influential, and validating those reasons. The question of security arises when reporting any of that information publicly. Any information about users during a disaster event must be properly scrubbed of information that may identify actors involved in the disaster that may introduce risk of harm, whether that harm is physical, emotional, financial or otherwise. The problem enters the area of anonymizing data with means that guarantee a reduction or elimination of those risks.

An example of the risk of publicly reporting data is demonstrated by the de-anonymization research performed by Narayanan and Shmatikov [82] on the Netflix challenge dataset where a significant percentage of the population (in some cases more than 80%) can be uniquely identified in a supposedly "anonymized" dataset. Another example of potentially sensitive information that may appear in the data could be information about the deaths of individuals such as those involved in the Virginia Tech shootings of 2007. A study performed by Palen *et al.* [92] show that many groups of people had names of the deceased and were posting that information publicly before public health and safety officials had made any official announcement on the incident.

Another problem that arises when reporting any of the information gained in the data analysis is gaming of the system. This is the problem of dealing with delinquent actors in a network who are actively trying to subvert or break the system by injecting false or misleading information. These actors could simply be curious individuals who would attempt to inflate the influencer score of themselves or another user, or as a worst-case scenario, multiple actors would be involved an organized group of individuals performing a Sybil attack in order to misdirect the public or officials from a terrorist attack by giving misleading information about a possible attack in a different location than the one planned.

Research in the area of manipulation of networks and ranking systems includes Cheng and Friedman [22] who study how the PageRank algorithm may be manipulated with Sybil attacks, Yu *et al.* [128] who propose a technique for detecting nodes that are participating in a Sybil attack to infiltrate the trust network of users in a social network, Puttaswamy *et al.* [98] who propose a technique to protect user data in a network using k-anonymity, and Gayo-Avello [40] who analyze how different ranking methods (PageRank, HITS, NodeRanking, TunkRank, TwitterRank) may be manipulated by attackers.

While I do not go in depth in this area in the research in this thesis, these issues must be very carefully considered when building systems and algorithms operating on public data that may be public as they must be robust from attack.

2.5 Goals for Our Research

In this research I seek to answer the question described in section 2.1, “who is important in an online social network during a specific event being discussed within that network?” In chapter 3 I will investigate numerous methods and tools that may assist in answering this question. The subsequent layers of the problem as described in this chapter will be left for future work.

Chapter 3

Analysis Methods

In this chapter I broadly survey analysis methods and tools from many different science disciplines such as biology, linguistics, computer science, physics, economics, and others. We begin in section 3.1 by describing the different types of data I encounter in social networks and define attributes I will investigate. Sections 3.2 through 3.5 describe a number of different statistics, probability, and natural language processing methods used in social network data analysis. Sections 3.6 through 3.14 cover a number of graph-theoretic methods for determining community and user influence based on the social graph. We conclude the chapter with a summary of the methods and briefly mention other work not covered in the chapter. For most of the methods, I will provide the following:

- A high-level description of what the method does,
- Prior work related to the method being described,
- A description of the algorithm,
- An analysis of the complexity and running time of the algorithm, and
- An analysis of how this method may be applied to inferring reputation of users or information in social networks.

3.1 Social Network Data

Before I am able to investigate the data online produced by people, I must first discuss the motivations for choosing to investigate the analysis methods and tools described in this chapter. General data analysis includes a vast set of tools, methods and algorithms, so I need to understand the form and context of the data to be analyzed. In this section I discuss what data is available to us from different online sources that may be used to infer reputation.

3.1.1 Online "Social" Data

Many services exist for the publication of information online by the average home internet user. Aside from the long ago popular Bulletin Board Systems (BBS), one of the first methods for publication was simple website hosting by providers in the mid-to-late 1990s such as the GeoCities[1] service. Over the last eleven years, a number of new services have appeared and old services have matured with larger feature sets. These services include rating sites (sites that allow people to give feedback on products or people), forums (the evolution of bulletin board systems where people are able to create discussions around a topic), news and blogs[2] (places were news or opinions are posted and people may leave comments), and wikis.[3]

As capabilities evolved, another feature began emerging in these services to allow users to declare connections, or relationships, to one another. The common term *social network* is typically reserved for online services that include this relationship feature in addition to the ability share personal information, photos, videos or opinions about themselves or their experiences. Examples of popular services that fall into the *social network* category include Facebook, Twitter, MySpace,[4] and many others. Table 3.1 lists some examples of online services that fall into these different categories. These divisions are not strict; some services such as Ning include many features that span multiple categories. The category explored in this research is the *social networks* category

[1] GeoCities was acquired by Yahoo in 1999 [27] but no longer exists as an available hosting service [111].
[2] News and blog websites have a very similar feature set and therefore I group them together here.
[3] Wikis are usually site-specific and are not open to public modification with only few major exceptions such as http://wikipedia.org and http://wikibooks.org.
[4] http://www.myspace.com

(colored grey in the table) that includes the network graph connectivity information in addition to other content shared among users in messages, photos, videos, and other media.

Table 3.1: Examples of online services

Ratings	Forums/News/Blogs	Social Networks
amazon.com	slashdot.org	twitter.com
epinions.com	cnn.com	facebook.com
netflixx.com	wordpress.org	myspace.com
bizrate.com	blogspot.com	linkedin.com
reviews.cnet.com	thedailywtf.com	flickr.com
hotornot.com	ning.com	ning.com
angieslist.com	livejournal.com	orkut.com

3.1.2 Social Network Data Attributes

In order to determine what available analysis methods and tools would be most applicable for analyzing social network service data, I must understand the form of that data. The data for *social networks* falls into three general categories:

(1) **content** - The content of the messages, profiles, pictures, videos, or any other type of media shared between users,

(2) **metadata** - Extra data included with that content, and

(3) **social graph connectivity** - The declared relationship links between users.

Obviously the first two categories are common to all types of online services, but *social networks* differ from most forums, blogs and rating websites by the existence of the declared social graph. The content category is divided into four main types: *text*, *pictures*, *audio*, and *video*. The metadata associated with each type varies wildly depending on the social network, but nevertheless metadata exists. The social graph connectivity of the popular social network services is typically binary, meaning a link exists or does not exist. Rarely are social graph links given weights to represent different kinds of relationships (friend, acquaintance, business partner, etc). The social graph is also typically very visible by default for all to see. For the purposes of this research, I am only interested in the text and metadata of messages and profiles as well as the social graph.

3.1.2.1 Message and Profile Text

Analyzing message and profile text falls under the domain of Natural Language Processing (NLP), specifically in the area of Information Retrieval (IR). When this analysis is performed in the context of reputation, I am interested in methods and tools that assist us in understanding the following:

- **Keyword/topic detection** - Understanding what words co-occur among many users to help determine both topic groups and other words to monitor.
- **Phrase detection** - Determining what phrases are being used among a group of users
- **URLs** - Knowing what websites are referenced among a group of users.
- **Duplicates detection** - For both detection of spam or bots as well as detection of when information is forwarded between different users that is deemed interesting.
- **User mentions** - When users are mentioned by others in text, either in response to or forwarding of existing information.

3.1.2.2 Message and Profile Metadata

Analyzing metadata becomes largely a problem of statistics and probability in terms of counting, grouping, and predicting the likelihood of users having certain attributes or metadata values. The metadata that is most important to us is the following:

- **Timestamps** - We want to know when messages were sent, profiles were created or modified, or relationships in the social graph were created.

- **Data source** - In a number of social networks, the source interface (such as the website or phone application) is available. This information reveals user behavior as he or she interacts with the social network.

- **Location** - One of the most sought-after pieces of information is the physical location of a user in a social network as they are interacting with the site, or revealing where they may have been at the time posted photos or videos that were created.

- **Language** - Language settings may assist in determining other attributes of users in a social network, and occasionally this information is available.

3.1.2.3 Social Graph

The availability of the social graph representing declared relationships allows us to apply graph-theoretic methods and algorithms to determine a number of different attributes. Some of those attributes I may wish to know are as follows:

- **Community** - Users automatically form or seek out online communities, but I want to be able to detect the boundaries of those communities.

- **Similarity** - Combining similarity measures and the social graph may yield additional information about users based on their association with people in the social graph and what common interests or activities they may share.

- **Influence/Rank** - We want to understand who is the most influential in a network of people during a specific event, and the social graph may be able to inform us of which users have the farthest-reaching influence when sending messages.

- **Connectivity** - Other relationship patterns in the social graph may yield extra information about the interaction between users, for example in the friend-of-a-friend motif[5] in a network.

[5] One available tool for motif detection is FANMOD [124].

3.1.2.4 Investigating Analysis Methods and Tools

Now that I have an understanding of the data attributes I want to discover or further understand within social networks, I am now able to investigate different analysis methods and tools that may assist in determining the above information. If I am able to gain this information, I may then begin analysis to determine whether or not the information may confer reputation on users or information within social networks.

3.2 Similarity Measures

In this section I describe a number of different metrics used to determine similarity between different sets of data. The methods described here are not exhaustive but are merely some of the common methods used in different data analysis. We follow-up the descriptions of the methods with a summary of how each may be applied in inferring reputation.

3.2.1 Jaccard Similarity

The Jaccard Similarity Coefficient (also known as the Jaccard Index) was developed by French botany professor Paul Jaccard in 1901 [56]. The coefficient measures the similarity (or distance) between two sets of data. The calculation is defined as the size of the intersection of the sets divided by the size of the union of the sets as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

3.2.1.1 Complexity

Given pre-created sets A and B of size m and n respectively, the time complexity of the algorithm is $O(m+n)$ when using efficient data structures.

3.2.2 Pearson's Correlation Coefficient

Pearson's Correlation Coefficient measures the linearity between two variables. Values range from -1 (perfect negative relationship) and +1 (perfect positive relationship), where 0 is no correlation at all. N is size of the sample set of both variables.

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X^2)}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N}}} \tag{3.2}$$

3.2.2.1 Complexity

The calculation occurs in $O(n)$ time. All sums and squares can be calculated in one pass followed with a calculation in constant time.

3.2.3 Assortative Mixing

Assortative mixing, proposed by Newman [85], is a property of networks where vertices in a graph are connected to other vertices with the same or similar attributes of each other. To calculate the assortative mixing value r, I define M as the number of edges, N is the number of vertices, and j and k are the vertices at either end of an ith edge. The following calculation results in a value ranging from -1 (perfectly disassortative) to $+1$ (perfectly assortative):

$$r = \frac{M^{-1} \sum_i j_i k_i - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2}{M^{-1} \sum_i \frac{1}{2}(j_i^2 + k_i^2) - [M^{-1} \sum_i \frac{1}{2}(j_i + k_i)]^2} \tag{3.3}$$

3.2.3.1 Complexity

This calculation is run over all M edges in the network, and so runs in $O(M)$ time.

3.2.4 Spearman's Rank Correlation Coefficient

Spearman's Rank Correlation Coefficient, also known as Spearman's Rho, is also a measure of linearity but after the variables are converted to rankings (starting at 1 for the smallest value of each variable and increasing from there). The measure essentially determines whether or not the two variables have a monotonically increasing or decreasing relationship as the value of the independent variable increases. N is the number of data points and x_i, y_i are the ranking values for each variable.

$$\rho = 1 - \frac{6 \sum_i (x_i - y_i)^2}{N^3 - N} \tag{3.4}$$

3.2.4.1 Complexity

If one has values of variables that require conversion to ranking scores, a sorting method must be used on both variables in order to set the appropriate ranking scores. A fast sorting routine such as quicksort will on average require $O(n \log n)$ time which must be run twice. Once the ranking scores are calculated, the correlation score requires $O(n)$ time to sum the difference in ranking scores.

3.2.5 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors of size N by finding the cosine angle between the vectors by calculating the dot product of the two vectors. Since the measure computes the value for any pair of vectors of equal length, many different types of data can be converted into vector form to be compared using this method. Given two vectors A and B of size N:

$$cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} \tag{3.5}$$

3.2.5.1 Complexity

The time to compute a dot product is simply $O(n)$ which makes this a very useful and common similarity measure for large or multiple dataset comparisons.

3.2.6 Euclidean Distance

Euclidean distance is commonly used to measure similarity among data points, but does not necessarily have to represent real distance measures. The Pythagorean Theorem is used to determine the distance between two points in n-dimensional space. Given two points $\mathbf{p}$ and $\mathbf{q}$ in space with n dimensions, $\mathbf{p} = (p_1, p_2, ..., p_n)$ and $\mathbf{q} = (q_1, q_2, ..., q_n)$. We then calculate the length of the line segment $\overline{\mathbf{pq}}$ as follows:

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1} (p_i - q_i)^2} \tag{3.6}$$

3.2.6.1 Complexity

The calculation of euclidean distance takes time $O(n)$ where n is the dimension of the space of the points. Typically $n = 2$ or $n = 3$ in most real-world applications, so this calculation can be considered constant time. If S is the set of all points in a sample space, the time to calculate distance between all points is $O(\|S\| \log \|S\|)$.

3.2.7 Normalized Mutual Information

The Normalized Mutual Information metric was first proposed by Fred and Jain [38] as a means of comparing different clusterings of data with ground truth information to measure the consistency between different data clusterings. Mislove *et al.* [81] apply the metric to a global hierarchical clustering method (described in section 3.8) using profile data collected from Facebook.

Then the metric is calculated between the two clusterings to determine how well the clusterings match. The value ranges from 0 to 1, where 0 is no correlation between clusterings, and 1 is an exact match between clusterings.

- $\mathbf{x}$ - A square matrix whose rows (typically) correspond to the ground truth data clustering, and whose columns correspond to data clusterings from a second clustering method
- $\mathbf{x}_{ij}$ - The number of nodes in ground truth cluster i that appear in the detected cluster j

- $\mathbf{x}_{.i}$ - The sum over column i
- $\mathbf{x}_{i.}$ - The sum over row i
- N - The number of vertices in the graph

$$\frac{-2\sum_i \sum_j x_{ij} log(\frac{x_{ij}N}{x_{i.}x_{.j}})}{\sum_i x_{i.} log(\frac{x_{i.}}{N}) + \sum_j x_{.j} log(\frac{x_{.j}}{N})} \tag{3.7}$$

3.2.7.1 Complexity

Since the double summation exists in the top of the fraction, the worst-case running time of the algorithm is $O(mn)$ where m and n are the number of attribute-based and algorithm-based clusterings, respectively.

3.2.8 Naive Bayes Classifier

The Naive Bayes Classifier method is based on Bayes' theorem named after Thomas Bayes, an English mathematician in the 18th century who derived a special case of the theorem and was published posthumously. The method works to describe posterior probabilities of events occurring, given an existing set of evidence. Essentially, the method works to classify data into categories based on the similarity different samples of data compared to a known set. Blei *et al.* [14] and Ramage *et al.* use use Bayes Theorem in the Latent Dirichlet Allocation method to classify words in documents to different topics. Sankaranarayanan *et al.* [104] use a naive Bayes classifier to determine whether or not different Tweets in the Twitter service are considered "news" or "junk". To calculate a Bayesian probability, we have the following definitions:

- $P(H)$ - The probability that our hypothesis H may occur.
- $P(\sim H)$ - The value $1 - P(H)$.
- $P(E|H)$ - The conditional probability that the evidence will occur given the hypothesis has occurred.

- $P(E| \sim H)$ - The value $1 - P(E|H)$.

The basic calculation for a Bayesian probability is as follows:

$$P(H|E) = \frac{P(E|H) \times P(H)}{P(E|H) \times P(H) + P(E| \sim H) \times P(\sim H)} \tag{3.8}$$

The Naive Bayes Classifier then divides data as either matching or not matching based on a specific probabilistic threshold. The threshold value must be tuned for each application and is usually found empirically from tests.

3.2.8.1 Complexity

The complexity of the probability calculation is simply $O(n)$ to obtain the necessary values of $P(H)$ and $P(E|H)$, followed by a constant calculation.

3.2.9 Applying Similarity Measures to Reputation

Similarity measures are very flexible in how they may be applied to analyzing data. In the context of social networks, similarity measures are mainly used to determine how well-clustered users are in different ways. Some examples of the different ways these measures may be used are determining the similarity between the sets of words used in messages in a group of users, the similarity of attributes between users in a group, the frequency of postings, or the time of day of postings. There are many arrangements of the data to be compared withs similarity measures in this fashion.

Jaccard Similarity is useful for comparing sets of data without any need for ordering and results in a value between 0 and 1 where 0 is no similarity between sets at all, and 1 is fully identical. As an example, this method has been used by O'Connor *et al.* [90] for determining whether or not a Tweet is actually a re-tweet of a prior Tweet using this simple "bag of words" approach where spam is likewise grouped due to similarity.

Cosine Similarity is widely used in data analysis as a similarity measure due to its simplicity and fast calculation. It can be used to compare words in documents and normalize the comparison between documents of different word counts as well as compare vectors of profile attributes. Some examples include Narayanan and Shmatikov [82] who use cosine similarity in their popular analysis of user profiles in the Netflixx dataset, Sankaranarayanan *et al.* [104] who determine topic clustering of Tweets with cosine similarity, and Sayyadi *et al.* [105] use cosine similarity to cluster documents around topics.

Euclidean Distance is commonly used as the similarity metric for data values that have measurable delta values between each other. Obviously this metric is used when dealing with real-world distance values, but other measurements that can be plotted in a coordinate plane may use euclidean distance to determine similarity as well.

Pearson's Correlation Coefficient is a very common method for quickly determining relationships between different variables in a data set. It has been used by Cha *et al.* [21] and Tiovonen *et al.* [112] to measure in-degree and out-degree correlation of friends links in social networks, by Ziegler and Lausen [129] to measure the similarity between recommendations of users rating books on Amazon,[6] and by Newman [85] who uses Pearson's Correlation as the basis for the measure of assortativity in social networks which is described in section 3.2.3.

Assortative Mixing is known as "homophily" or "positive preferential attachment". Newman states, "Patterns of friendship between individuals for example are strongly affected by the language, race, and age of the individuals in question, among other things. If the people prefer to associate with others who are like them, we say that the network shows assortative mixing or assortative matching" [86]. Assortative mixing is based on Pearson's Correlation Coefficient [85] which is described above in section 3.2.2. Newman [86] studies assortative mixing in the context of ages of marriage partners, racial mixing in heterosexual relationships, websites on the internet, co-authorship networks, address books, software dependencies, and multiple biological networks such as protein interactions, neural networks, food webs, and more. Chun *et al.* [23] study assortative mixing by vertex degree in the Cyworld social network in South Korea and Bagler and Sinha

[6] http://www.amazon.com

[5] apply assortative mixing to protein contact networks. Assortative mixing is also the basis of the Girvan-Newman modularity measurement [42] for community detection which is described in section 3.7.2.3.

Using assortative mixing we may find interesting patterns in the data of users for different events. We may attempt to cluster users around different attributes or keywords and this method could be used to determine whether or not a specific attribute clustering lends to positive preferential attachment during that event.

Spearman's Rank Correlation Coefficient can be used to compare different reputation metrics to determine the efficacy of a new approach to determining reputation. If I find that an analysis method is particularly good at determining location, I could use Spearman's correlation to see how well a second location method performs when compared to the first method. One issue that Spearman's ranking has is when the independent variable x_i has two or more tied rankings, the calculation is no longer valid. Two data points that have exactly the same values do not have any linear correlation, so Pearson's correlation coefficient described in section 3.2.2 must be used and the tied rankings end up sharing an averaged ranking score.

Normalized Mutual Information is simply another way in which to compare clusterings of data, and in this case will be very useful for comparing the many different graph clusterings that are reviewed in this chapter. For example, we may use this measure to compare ground-truth data of human-analyzed datasets to k-shell decomposition which is a variation of k-core decomposition as described in section 3.11. This measure has been used in this way by Mislove *et al.* [81] when comparing ground-truth clusterings of users in Facebook to the global hierarchical clustering method as described in section 3.8.

A **Naive Bayes Classifier** may be applied to many different aspects of social networking data. We may have a set of users who share an attribute, and may determine the probability of whether or not a new user shares that attribute based on other data from user profiles. We may also apply the Latent Dirichlet Allocation methods to determine groupings of users based on the topics and keywords they share. These are only two examples of a great variety of applications for this specific measure.

3.3 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) is a very common method in information retrieval systems for determining the importance of a word in a group of documents. The Inverse Document Frequency (IDF) portion of the method was proposed by Spärck Jones [108] as a means to find words that are good discriminators between different documents. Intuitively, a word that occurs in many documents should be given a lower weight than one which occurs in few documents. When combined with Term Frequency (TF) which in the simplest manner is simply the count of the words in a document, Robertson [100] describes TF-IDF as a measure which is “proved extraordinarily robust and difficult to beat” for determining the importance of a word among documents.

3.3.1 Related Work

Bilenko and White [13] use a variant of TF-IDF to determine relevant web pages based on user search and browsing history based on query terms and how many times those query terms appear in documents that users have visited. Costache *et al.* [29] use global TF-IDF scores to personalize PageRank scorings for websites. Sankaranarayanan *et al.* [104] use TF-IDF scores combined with Cosine similarity to determine whether or not to add a Tweet post to a clustering of other news-related Tweets.

3.3.2 Algorithm

Robertson [100] presents equations and derivations of the measure. The probability of a term t_i occurring in a document d is given as the following where n_i is the number of documents in which t_i occurs and N is the count of all documents:

$$P(t_i \text{ occurs in } d) = \frac{n_i}{N} \tag{3.9}$$

Now the IDF is given as the following:

$$idf(t_i) = -\log P(t_i) \tag{3.10}$$

3.3.3 Complexity

The TF-IDF calculation requires a count of words and phrases in all documents considered, which takes $O(n)$ time where n is simply the total number of words in all documents. Any additional documents can be added to the total without the need to recalculate the values from the beginning.

3.3.4 Applying TF-IDF to Reputation

The TF-IDF method works well with a large collection of documents that have a reasonable length. One issue that arises with TF-IDF in the context of microblogged messages is that the messages are very small (the average Tweet is 11 words long) and words very rarely repeat so the term frequency is also essentially a count of the documents in which it occurs [90]. Further, new words or misspellings can lead to a large TF-IDF score when they may not actually be important [104].

However, this technique can still prove to be useful as one of the important issues in dealing with social networking information is to determine when new events are occurring or what new keywords are starting to be used. One could envision a new keyword initially having a large TF-IDF score and over time this value drops as more people adopt the keyword or a larger phrase (n-gram) and then the derivative of the TF-IDF score over time may become an important indicator of adoption of a keyword or phrase.

3.4 *N*-Gram Analysis

The *n*-gram analysis method is a word counting technique in the natural language processing field that computes the probability of a series of words occurring in a specific order. For example, I would use *n*-gram analysis if I want to compute the probability of the word “nine” occurring after the series of words, “I went to bed at”. The size of the grouping of words is the value of the n in n-gram, and *2*-grams are called *bigrams* and *3*-grams are called *trigrams.* The method offers insight into a set of documents as to the common set of phrases that appear in the corpus.

3.4.1 Related Work

N-gram analysis is a very common Natural Language Processing method and is taught in courses on the subject and appears in textbooks [60]. The usefulness of n-grams is demonstrated by Google’s release of their *5*-gram (and fewer) corpus of phrases that appear at least 40 times among all websites they crawl for the purposes of facilitating public research [45]. O’Connor *et al.* [90] use unigram, bigram, and trigram analysis to determine common phrases in tweets in the Twitter service. N-grams are also not limited to processing text, but has been applied to computing the distance between protein sequences [55].

3.4.2 Algorithm

Here I will describe the process for computing unsmoothed n-grams for a corpus of text. The process for computing the probability of a sequence of words of size n involves computing the conditional probabilities of the preceding $n - 1$ number of words in the sequence. For *bigrams* this means I only compute the probability of a word w_n occurring based only the previous word w_{n-1}. Using the example in the introduction to this section, instead of computing the following probability (which is read, “What is the probability of the word *nine* occurring given *I went to bed at*?”):

$$P(\text{nine}|\text{I went to bed at}) \tag{3.11}$$

We instead approximate the *bigram* using maximum likelihood estimation of bigrams from a specific corpus of data normalized to their relative frequencies in the corpus, and then multiply the relative frequencies together:

$$\begin{aligned} P(\text{nine}|\text{I went to bed at}) &= P(\text{went}|\text{I}) \times P(\text{to}|\text{went}) \times P(\text{bed}|\text{to}) \times \\ &\quad P(\text{at}|\text{bed}) \times P(\text{nine}|\text{at}) \end{aligned} \tag{3.12}$$

3.4.3 Complexity

The complexity of computing the phrase probabilities using n-grams is contained in counting the n-grams in the corpus and simultaneously computing the relative frequencies of those n-grams. The process of computing n-grams requires a single pass of $O(w - n + 1)$ where w is the count of all words in the corpus. Different n-gram sizes must be computed separately. The process of computing the actual probability of a word given a phrase using bigrams is a constant calculation with simple lookups to the computed relative frequencies.

3.4.4 Applying N-gram Analysis to Reputation

The majority of the data available in social networks is contained in text if I ignore the other media types of pictures, audio and video. N-gram analysis offers the potential for culling out useless information from bots or spammers, or from messages that are unrelated to the event being considered at that moment for determining reputation. This analysis will lend insight into the phrases that are popular among users during an event, and if combined with effective visualization tools, will offer researchers a means of quickly detecting phrases that are becoming important over the course of the event.

3.5 K-means Clustering

The k-means clustering method, formally introduced computationally by Lloyd [73], is a method for automatically partitioning data into k different sets. The value of k is either pre-determined or iteratively increased to determine clustering. The method is unsupervised and works to cluster data around k points (called *centroids*).

3.5.1 Related Work

The original idea was initially explored by MacQueen [75] in 1967 before Lloyd [73] offered a euclidean-space based computation. The method is very commonly used in data mining techniques with widely varying applications. The following are a small sample of applying k-means to information available online: Huttenhower *et al.* compare their Nearest Neighbor Networks clustering method (described in section 3.10) to k-means. Fred and Jain [38] combine k-means clustering with normalized mutual information (described in section 3.2.7) yield more robust data clustering across various data sets. Strehl *et al.* [110] evaluate k-means clustering among other methods in the context of web page clustering.

3.5.2 Algorithm

In order for data to be clustered using the k-means algorithm, the data must first be converted to comparable values (if it is not already in that form). For example, if one wants to look at attributes of fruit, such as weight and type, one first has to convert the type (apples, oranges, bananas, etc.) to numerical values. What values are used are not important, as long as the values are uniquely represented in a set. Once the data is in a comparable form, the k-means method operates in five major steps:

(1) Choose a value of k for the number of clusters to be discovered.

(2) Choose starting position of each of the k centroids. Note that the centroids may be chosen specifically or randomly and that the starting positions may affect the final clusterings.

(3) Calculate the distances for each data point in the sample space to each centroid. This calculation is done using the Pythagorean Theorem described in section 3.2.6.

(4) Group each data point according to its closest centroid. If the groupings have not changed from the previous iteration, then stop.

(5) With the current groupings of points to their nearest centroids, calculate the new values of the k centroids based on the membership. This is done by calculating the average coordinate among all data points in each group.

(6) Repeat from step 3 until finished.

3.5.3 Complexity

Step 3 requires $O(kn)$ time to calculate the distance from k centroids to n points in space. Each distance calculation requires constant time which is of order of the dimension of the sample space. Depending on how the distances are sorted and stored, adding a data point to a grouping is dependent on the data structure. If a max-heap is used for each data point for its distance values to each centroid, grouping time is $O(\log k)$ for step 4. Step 5 requires $O(n)$ time for computing new centroids for all groups. Then I must iterate until centroids no longer change their values, which may take I times. This leaves us with a worst-case running time of $O(In^2 \log k)$. Many times the convergence occurs much more quickly than worst-case.

3.5.4 Applying K-means Clustering to Reputation

The k-means algorithm is applicable in attempting to infer reputation, especially in the early stages of data analysis. The method assists with understanding the basic patterns in the data which I am able to apply to word and n-grams, profile or message metadata, and any available location data. One of the problems with keyword search on social networks is that results may contain more than one search term. One of the ways I would be able to use k-means clustering would be to group users to keywords in search results in order to determine to which keyword users are closest, or said differently, which keyword users are using the most in their messages. For other general data analysis, I can compute clusterings of users based on varying profile attributes or friends.

3.6 PageRank

The PageRank method, proposed by Larry Page and Sergey Brin [16], is one which calculates the eigenvector centrality of pages which is the probability that a random surfer will follow a link to a specific web page on the internet based on the number of links to that page, as well as the number of links to parent pages as well. A page with very low in-degree may be linked from a very highly ranked page, and therefore confers a high pagerank onto the low in-degree page. They also account for the fact that a surfer may also randomly choose to go to a separate website and not follow the links on the current page (known as the damping factor). The algorithm works iteratively and converges relatively quickly to useful values.

3.6.1 Related Work

The PageRank algorithm is essentially a very large Markov chain as described by Ding *et al.* [31] since a user following links on a page is independent of the previous page that user visited. TunkRank, proposed by Tunkelang [113], is an implementation of PageRank on Twitter relationships.[7] The TwitterRank method developed by Weng *et al.* [123] uses a modified PageRank in addition to topic modeling to find influential users on Twitter. Costache *et al.* [29] use a variant of PageRank to personalize results based on individual and peer history. Gyöngyi *et al.* [49] propose the TrustRank algorithm which combines human-rated pages (spam or not spam) with PageRank and transitive trust values to compute "good" and "bad" pages. Other investigations into the efficacy and manipulability of PageRank include Bar-Yossef and Mashiach [10] who investigate local approximations of PageRank, Cheng and Friedman [22] who look at how PageRank may be manipulated using Sybil attacks, and Forsati and Meybodi [37] use learning automata with PageRank to recommend pages to users based on their current browsing patterns. PageRank has been extended in many other ways including dealing with different weights for outgoing page links and working on undirected graphs. Ding *et al.* [31] and Hopcroft and Sheldon [51] also offer reviews

[7] A web-based implementation of TunkRank is available at http://tunkrank.com

of other extensions to the original PageRank algorithm.

3.6.2 Algorithm (PageRank)

Here I will describe the original algorithm from Brin and Page [16]. Given the structure of the internet, I am able to represent the structure as a graph $G = (V, E)$ where pages are considered the vertices V and links to other pages are considered the edges E. We have the following definitions for the algorithm:

- $PR(p)$ - The rank value of page $p \in V$
- N - The number of total pages
- d - The damping factor
- I_A - The set of pages that link to page A
- $|O_p|$ - The count of the set of all outgoing links of page p

The algorithm begins in an initial state where all pages have an initial probability of $1/N$ and the damping factor is set to .85. We then are able to calculate the rank of a page A as follows:

$$PR(A) = \frac{1-d}{N} + d \sum_{p \in I_A} \frac{PR(p)}{|O_p|} \tag{3.13}$$

The rank calculation occurs for each page in the graph, and is run iteratively until the values converge. The higher the damping factor, the longer the values take to converge as the calculation relies more on the link structure of the web instead of the random behavior of surfers [31].

The running time of the original algorithm is $O(nD)$, where n is the number of pages being considered and D is the average in-degree of all pages. The more common power method runs in $O(n)$ time and only requires 50-100 iterations [71]. The space requirements of the PageRank algorithm requires the sparse adjacency matrix A which is implemented as an adjacency list, a vector to store dangling leaf pages that to not have links to other pages, and a vector of size n containing multiplication values for each iteration.

3.6.3 Algorithm (TunkRank)

TunkRank, proposed by Tunkelang [113], is a computation analogous to PageRank to determine the influence of users in Twitter based on how those users are connected in the social graph and how often users "re-tweet" messages. The intuition of this algorithm is that users who follow few users are able to give more attention to the messages of the users that they follow, and therefore confer a large ranking on those they follow.

The initial state of the influence rank of all users is $1/N$ where N is the number of users in the network. The damping factor d from PageRank has been replaced by the "Re-Tweet" factor p. To determine the influence of a user X, the following equation is evaluated:

$$Influence(X) = \sum_{Y \in Followers(X)} \frac{1 + p * Influence(Y)}{|Following(Y)|} \tag{3.14}$$

The calculation is done for all users and is performed a number of times until the values begin to converge. The algorithm can be implemented with the same power method as described in section 3.6.4 offering the same quick convergence.

3.6.4 Algorithm (TrustRank)

The TrustRank method proposed by Gyöngyi *et al.* [49] is a supervised transitive trust model combined with PageRank to do determine whether or not a web page is considered to be spam. The algorithm is based on the assumption that "good" pages rarely link to "bad" pages and therefore good pages transfer high trust values to downstream pages. A set of pages that are evaluated by humans to be good or bad are stored in the oracle L. The algorithm occurs in five major steps:

(1) Calculate seed value s for each page in an initial set of pages using an *inverse* PageRank score

(2) Order seed pages by their values from high to low

(3) Invoke the oracle L on a limited set of high-value seed pages to generate a score distribution vector d of all pages where good seed pages are set to 1 and all other pages are set to 0.

(4) Normalize the distribution vector d

(5) Iteratively calculate TrustRank scores from 1 to iteration limit M using a modified PageRank computation

Since this algorithm uses the PageRank algorithm, the running time is similar although slightly worse as the two versions of PageRank are run during the algorithm (one at step 1 and another at step 5). Time is also required by humans to rate a set of pages as spam or good that are then contained in the Oracle for lookups. The iteration limit M is simply the amount as described previously in section which will typically be 50-100.

3.6.5 Applying PageRank and Related Algorithms to Reputation

PageRank works to impart a reputation-based ranking on a page which essentially is how important the world believes a page to be by 'voting' using links to the page. This same method may be applied to social networks, but it has the major limitation of requiring "global knowledge" of the network in order to confer appropriate ranking values. In the cases of social networks, only the network provider (Twitter, Facebook, etc) has access to the entire network. Although public access may be available to some of this information via Application Programming Interfaces (APIs), a lot of time is required to collect the network data due to rate and authentication limits and not all the data may be available. Locally approximating PageRank has been studied [10] but is NP-hard which makes the measure infeasible for large graphs. Still, attempting to locally approximate by potentially collecting enough of the graph around a set of users is worthwhile to investigate. PageRank-related algorithms are popular versus other ranking methods due to the good speed and storage efficiency of the algorithm [71], and if I am able to obtain the social network graph data, these related algorithms plus potential variations on them may be useful.

TunkRank would be the algorithm of choice for a PageRank variant if the entire social graph is available. The intent of TunkRank is to find users who are "influencers", but influence is merely a way of describing reputation. But again, this is a global method therefore inferring a global reputation which isn't a useful metric. With reputation being context-dependent, I would need a way to deal with pruning out users that are not important to the context being considered. One possible way to prune unimportant users may be to determine the activity network as described in section 3.12.

TrustRank has an aspect of explicit trust systems as it relies on human rating feedback in order to perform its calculation. However, the requirement of an oracle for only a limited set of web pages makes the algorithm feasible for use in a larger system that is supervised by humans. The process does require many interconnected pages in order to assign reasonable trust values to individual pages, which could be a potential limitation due to access limitations of social graphs. When applied to social networks, this method could be used to determine "good" or "bad" users who are rated by the oracle based on the messages that they send to others around them. The algorithm would then assign transitive trust values to users who are linked to by the seed set. Although this situation may prove more difficult for humans as a user in a social network may have many messages that need to be read and considered in the oracle scoring. However, human supervision could potentially be augmented or replaced by smart natural language processing algorithms that are able to determine a "good" user based on the automatically analyzed messages from a user.

3.6.5.1 Security Implications for Reputation

Caverlee *et al.* [19] investigate the effect of distributed collusion by multiple malicious users in a social network that is ranked using both PageRank and TrustRank compared with their proposed SocialTrust framework. SocialTrust is an explicit rating system that incorporates user feedback to give weights to the relationship links in the social graph. They illustrate the significant issue of ranking systems that contain multiple malicious users going undetected without the incorporation of user feedback into the scoring. They specifically compare SocialTrust with PageRank and

TrustRank and demonstration that malicious users are able to establish themselves in trusting relationships before misbehaving, and the PageRank and TrustRank methods (as well as TunkRank since it is based on PageRank) will not be able to detect this malicious behavior. This issue is one of the major issues when attempting to determine reputation of a user or piece of information in these social networks.

3.7 Betweenness Centrality and Modularity

The Betweenness Centrality method, originally proposed by Girvan and Newman [42], is a divisive[8] community detection algorithm. The method eliminates edges that are the most "between" communities, resulting in distinct, densely connected subgraphs. In subsequent work, Newman and Girvan [88] add a calculation called "modularity" to measure the best-fit divisions of a graph into communities using the Betweenness Centrality method.

3.7.1 Related Work

The Betweenness Centrality method is a modification of vertex betweenness centrality first proposed by Freeman [39] for edges. The calculation of edge betweenness includes shortest-path calculations in a graph, and two fast algorithms were developed independently by both Newman [84] and Brandes [15]. The Brandes algorithm has been applied to publications of articles related to genes by Wilkinson and Huberman [126] in order to determine networks of genes based on how keywords are linked in the published articles. Everett and Borgotti [32] analyze how different clustering techniques overlap including betweenness centrality. Pollner *et al.* [97] compare users who appear in multiple k-cliques in the Clique Percolation Method (discussed in section 3.9) to betweenness centrality and find a correlation between many k-clique membership and high vertex betweenness. Radicchi *et al.* [99] compare a proposed local edge clustering coefficient community detection algorithm with edge betweenness in both accuracy and complexity. Kitsak *et al.* [64] compare k-shell

[8] "Divisive" algorithms begin with a fully-connected graph and then iteratively remove edges until all vertices are disconnected.

decomposition with betweenness centrality in terms of predicting the spread of information through the network. Gloor *et al.* [43] use betweenness centrality to track the importance of a concept in the web in news, blogs, and forums.

3.7.2 Algorithm

The full betweenness centrality method includes single-source shortest path, betweenness centrality, and modularity calculations. We will describe each calculation here separately and describe how all three interact to determine community divisions in a graph.

3.7.2.1 Single-Source Shortest Path

Newman [84] uses a modified depth-first search to calculate shortest paths from a single vertex j as follows:

(1) Choose a vertex j and assign it distance zero, indicating it is zero steps away from itself, and set distance $d \leftarrow 0$.

(2) For each vertex k whose assigned distance is d, follow each attached edge to the vertex l at its other end and, if l has not already been assigned a distance, and assign it distance $d+1$ and also declare node k to be a predecessor of l.

(3) If l has already been assigned a distance equal to $d+1$, there is no need to set the distance again, but k is still declared to be a predecessor of l, indicating more than one shortest path to the original vertex j.

(4) Set $d \leftarrow d+1$.

(5) Repeat from Step 2 until there are no unassigned nodes remaining.

3.7.2.2 Betweenness Centrality

Betweenness centrality for an edge e_{ij} from vertices v_i and v_j is a count of the number of shortest paths from a single chosen source vertex j to all other vertices in the network that pass over edge e_{ij}. In order to calculate the betweenness value for each vertex in the network. The process is as follows:

(1) A variable b_k, taking the initial value 1, is assigned to each vertex k.

(2) Going through vertices k in order of their distances from j, starting from the farthest, the value b_k is added to the corresponding variable on the predecessor vertex of k. If k has more than one predecessor, then b_k is divided equally between them. This means that, if there are two shortest paths between a pair of vertices, the nodes along those paths are given a betweenness of $\frac{1}{2}$ each.

(3) When I have gone through all vertices in this fashion, the resulting values of the variables b_k represent the number of geodesic paths to vertex j that run through each vertex in the network, with end points of each path being counted as part of the path. To calculate betweenness for all paths, the b_k are added to a running score maintained for each vertex and the entire calculation is repeated for each of the n possible values of j. The final running scores are precisely the betweenness of each of the n vertices.

3.7.2.3 Modularity

Modularity is the key component to this enhanced algorithm to determine the “best fit” of community divisions. The measure is based on “Assortative Mixing” proposed by Newman [86] (described in section 3.2.3). The basic algorithm for calculating the modularity Q for a division of a network (removal of an edge with the highest ”betweenness” value) is as follows:

(1) Choose a division of a network into k communities

(2) Create a $k \times k$ symmetric matrix $\mathbf{e}$ whose element e_{ij} is the fraction of all edges in the network that link edges in community i to community j. Note that an edge linking communities i and j must be split, half-and-half, between the ij and ji elements of $\mathbf{e}$ to maintain symmetry.

(3) Calculate the trace of the matrix $\mathbf{e}$, which is the fraction of all edges in the network that connect to nodes within the same community:

$$\mathrm{Tr}\mathbf{e} = \sum_i e_{ii} \tag{3.15}$$

(4) Calculate the row (or column) sums which represent the fraction of edges that connect to nodes in community i:

$$a_i = \sum_j e_{ij} \tag{3.16}$$

(5) Finally, calculate Q as the trace of the matrix $\mathbf{e}$ minus the square of the sum of the rows (or columns) a_i from above:

$$Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}\ \mathbf{e} - \parallel e^2 \parallel \tag{3.17}$$

As mentioned in [88], if the number of within-community edges is no better than random, $Q = 0$. Values approaching $Q = 1$ indicate strong community, however typically values fall in the range of 0.3 to 0.7.

3.7.2.4 Edge Betweenness and Modularity

All three of the above calculations work together in the following manner to determine communities within a network:

(1) Calculate the betweenness for all edges in the network,

(2) Remove the edge with the highest betweenness value,

(3) Recalculate the betweenness for all edges affected by the removal, and

(4) Repeat from step 2 until no edges remain.

3.7.3 Complexity

The Betweenness Centrality method is quite complex due to the multiple distinct calculations for shortest path, edge betweenness, and modularity. The shortest path calculation requires $O(m)$ time where m is the number of edges in the network, the edge betweenness calculation requires $O(n)$ time where n is the number of vertices in the network, and the modularity measure requires $O(n)$ time to build the community adjacency matrix. To determine communities for the entire graph until all vertices are fully disconnected, the running time is $O(mn^2)$.

3.7.4 Applying Betweenness Centrality and Modularity to Reputation

The Betweenness Centrality with Modularity measure yields very good community divisions in any network that is of relatively small size. However, this is the major limitation of the method since it does have a very bad running time of $O(mn^2)$. Newman and Girvan observe that this algorithm is only feasible for networks of about 10,000 vertices at the time of writing in 2004 [88]. The trade-off is that the algorithm does perform well when determining community. Others have also cited this limitation as a reason to avoid using the algorithm [28] [25] although Newman [87] concludes that the algorithm yields the best results compared to other algorithms and should be used where computationally feasible.

This method may still be useful for small collections or approximations of networks that expand far enough out from the target group of users in a network to determine community. We will summarize other community detection algorithms in sections 3.8 and 3.13 that use this method as a basis with significant speed improvements.

3.8 Global Hierarchical Clustering

The Global Hierarchical Clustering method, proposed by Newman [87], is a graph community detection method and differs from the Betweenness Centrality method described in section 3.7 in the fact that it is an agglomerative method.[9] Newman developed the method to overcome the poor performance of the divisive Betweenness Centrality algorithm. The method also uses a modularity calculation in such a way to greedily maximize the modularity value when deciding which vertex to agglomerate into a specific community. Said differently, the measure only agglomerates the vertex that yields the largest increase (or smallest decrease) in the modularity. This method also requires global knowledge of the graph as the algorithm depends on the interconnectivity of all nodes in the network.

3.8.1 Related Work

The agglomerative method is a general one used in many applications, but has been applied to football conference groupings, jazz musician collaborations, and co-authorship relationships by Newman [87]. Clauset *et al.* [25] further improve the running time of the algorithm by Newman and apply it to the Amazon purchasing network with over 400,000 vertices and over 2.4 million edges. Mislove *et al.* [81] use this method to attempt to globally infer attributes of users in Facebook.

3.8.2 Algorithm

Here I describe the undirected graph algorithm from Clauset *et al.* [25]. The algorithm improves on the one from Newman [87] in that it does not store the adjacency matrix, but instead only stores the change in modularity between two communities, ΔQ_{ij}. The data structures required for the algorithm are as follows:

- A sparse matrix of ΔQ_{ij} values for each pair i, j of communities that have at least one edge between them. This matrix is represented as a balanced binary tree for fast queries and insertions,
- A max-heap H containing the largest element of each row of the ΔQ_{ij} matrix, including the labels i, j to identify the pair of communities, and

[9] "Agglomerative" methods begin with a fully-disconnected graph and then iteratively add edges to re-connect the vertices.

- A vector array containing sum a_i of each row of the adjacency matrix

3.8.2.1 Initial State

As with all agglomerative graph algorithms, each vertex begins as a sole member of a community. The value k_i is the degree of community i (which is simply the degree of the single vertex initially in community i) and m is the number of edges in the network. The initial ΔQ_{ij} matrix and a_i vector are set to the following:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2m} - \frac{k_i k_j}{(2m)^2} & \text{if } i, j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

$$a_i = \frac{k_i}{2m} \tag{3.19}$$

The max-heap H is then initially populated with the largest element of each row of ΔQ.

3.8.2.2 Joining Communities

The algorithm then selects the largest ΔQ_{ij} value from H and joins the corresponding communities. The ΔQ matrix is then updated with new values, but only the communities that also touch communities i and j need to be updated. To join two communities i and j, the operation is a merge into one of the two communities i or j, and the other community is removed from the ΔQ matrix. To merge communities i and j into a new community j, all communities that connect to i and j must be updated with new ΔQ values. To update the matrix and a_j values when a community k is connected to one or both of the merging communities, the following calculations are used:

$$\Delta Q'_{jk} = \begin{cases} \Delta Q_{ik} + \Delta Q_{jk} & \text{if } k \text{ is connected to both } i \text{ and } j, \\ \Delta Q_{ik} - 2a_j a_k & \text{if } k \text{ is connected to } i \text{ but not } j, \\ \Delta Q_{jk} - 2a_i a_k & \text{if } k \text{ is connected to } j \text{ but not } i, \end{cases} \tag{3.20}$$

$$a'_j = a_i + a_j \tag{3.21}$$

The selected ΔQ_{ij} value is then added to the global modularity value Q. The max-heap H values must also be updated for each ΔQ_{ij} value that was affected by the merging of communities i and j. This merge step is repeated for all communities until only one final community remains.

3.8.3 Complexity

Merging two communities i and j requires $O((k_i + k_j)\log n)$ time. Recalculating the values for the max-heap H also requires $O((k_i + k_j)\log n)$ time. In the worst-case, the degree of a community i is the sum of the degrees of all vertices in the original network that comprise i, meaning the community connects to many other communities. This means that in the dendrogram that represents the network, the number of joins is depended on the depth d of the dendrogram as well as the total degree of all vertices in the network which is $2m$ where m is the number of edges. This means the running time is $O(md \log n)$.

3.8.4 Applying Global Hierarchical Clustering to Reputation

Much like the Edge Betweenness method described in section 3.7, this method requires full knowledge of the network graph. One is able to run this algorithm on a subset of a network, but there's no way to determine if the modularity values will correspond to any meaningful division of communities without all of the vertices and edges. For this reason, follow-up work for discovering local communities lacking global knowledge of the network have been developed, and are described in section 3.13. However, if I am able to obtain the full network topology, or at least a subset of the network that spreads far enough from the vertices being considered, the speed of this algorithm makes it an attractive candidate for determining communities in a network.

3.9 Clique Percolation Method

The Clique Percolation Method (CPM), proposed by Derényi *et al.* [30], is a community detection algorithm with a distinct characteristic in that is allows a vertex in a network to belong to multiple communities rather than limiting a vertex to only one community. In most real-world networks, vertices (or more appropriately, people) may belong to multiple overlapping communities with different characteristics that define each community.

3.9.1 Related Work

Deréyi *et al.* [30] and Palla *et al.* [93] propose the clique percolation method for detection of overlapping communities initially on undirected graphs. In subsequent work Farkas *et al.* extend the method for weighted networks [34] and Palla *et al.* extend the method for directed networks [94]. Implementations of all of the above variations of the algorithm are available online.[10] The algorithm was initially tested against Erdős-Rényi (ER) uncorrelated random graphs [30] [34], and later applied to word associations [93] [94], protein networks [93], Google's website structure [94], paper co-authorships [93], correlation graphs of NYSE stocks [34], student email interactions [94], and also to Twitter relationships [57].

3.9.2 Algorithm

Here I will describe the algorithm from the supplementary information of Palla *et al.* [93] for undirected graphs. Before describing the operations, I must first define structures and relationships for this method. These definitions are given in [30]:

- **clique** - A set of vertices that are fully-interconnected with one another.
- ***k*-clique** - A clique that contains k vertices.
- **maximal clique** or **complete subgraph** - The largest fully-interconnected subgraph from a specific vertex in the graph

[10] http://CFinder.org

- **maximum clique** - The largest fully-interconnected subgraph of the entire graph. Note that a graph may have one or more maximum cliques, and may have multiple, smaller maximal cliques.

- ***k*-clique adjacency** - Two k-cliques are adjacent if they share $k-1$ nodes. We see this illustrated in the left portion of figure 3.1(a) where the highlighted black triangles share two vertices (or one edge).

- ***k*-clique chain** - A subgraph which is the union of a sequence of adjacent k-cliques as shown in the both portions of figure 3.1(a). The black highlighted clique in the right portion demonstrates a large k-clique chain. Note that a k-clique chain is not the same as a maximal clique.

- ***k*-clique connectedness** - Two k-cliques are connected if they belong to any part of the same ***k*-clique chain**.

- ***k*-clique percolation cluster** - A maximal k-clique connected subgraph, i.e. the largest ***k*-clique chain** discovered in the graph.

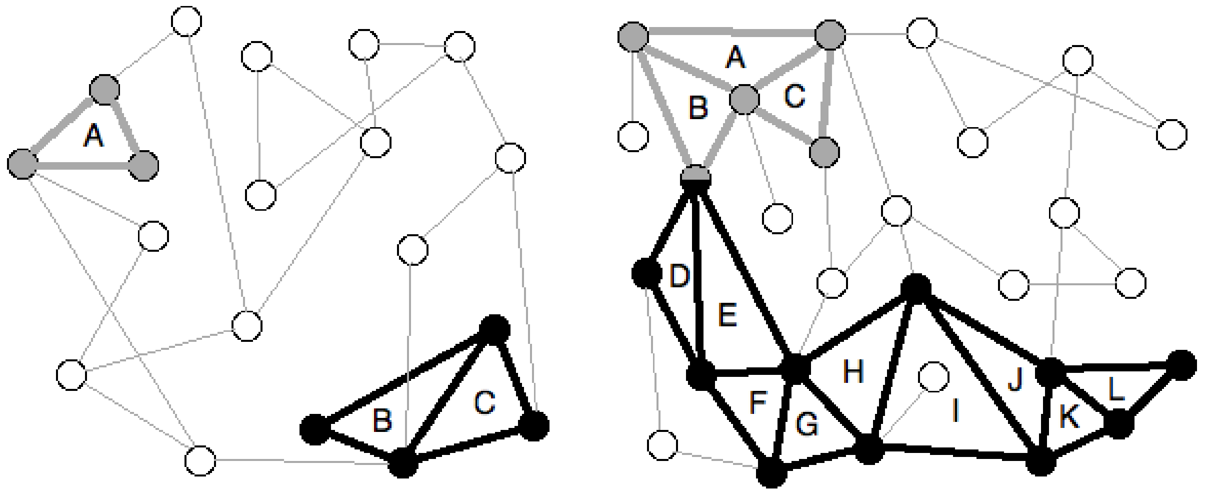
The CPM algorithm occurs in three high-level steps:

(1) Search the graph for all maximal cliques (as k-cliques are subgraphs of maximal cliques),

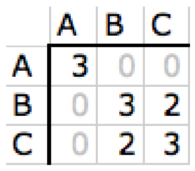
(2) Create an adjacency matrix of each of the discovered maximal cliques where the adjacency value is the number of shared nodes between each clique, and

(3) Reduce the matrix to evaluate a specific value of k by removing each diagonal entry with degree less than k and every off-diagonal entry with degree less than $k-1$.

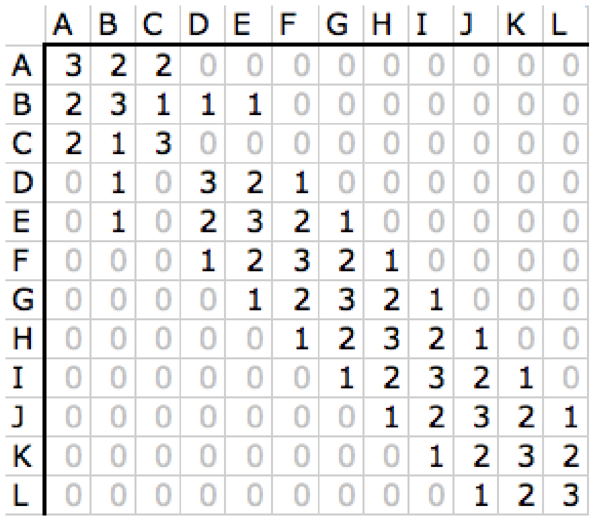
To locate the maximal cliques, one must exhaustively search through the graph one node at a time looking for surrounding nodes that share the same neighbors as the node in question. The search is a recursive one and consists of finding the intersection of the neighbors of nodes in the current clique and adding a node in the intersection and continuing the search for subsequent



(a) Two networks each with k-cliques of size $k = 3$. The left network has two small separated 3-clique percolation clusters. The right network contains one giant 3-clique percolation cluster (in black) overlapping with one small 3-clique percolation cluster (grey).



	A	B	C
A	3	0	0
B	0	3	2
C	0	2	3



	A	B	C	D	E	F	G	H	I	J	K	L
A	3	2	2	0	0	0	0	0	0	0	0	0
B	2	3	1	1	1	0	0	0	0	0	0	0
C	2	1	3	0	0	0	0	0	0	0	0	0
D	0	1	0	3	2	1	0	0	0	0	0	0
E	0	1	0	2	3	2	1	0	0	0	0	0
F	0	0	0	1	2	3	2	1	0	0	0	0
G	0	0	0	0	1	2	3	2	1	0	0	0
H	0	0	0	0	0	1	2	3	2	1	0	0
I	0	0	0	0	0	0	1	2	3	2	1	0
J	0	0	0	0	0	0	0	1	2	3	2	1
K	0	0	0	0	0	0	0	0	1	2	3	2
L	0	0	0	0	0	0	0	0	0	1	2	3

(b) The k-clique overlap matrices for the networks in figure 3.1(a). k-clique chains are found by removing diagonal elements of size less than k and off-diagonal elements of size less than $k - 1$.

Figure 3.1: Visualization of the clique percolation method

neighbor intersections. Cliques may overlap, so once a maximal clique is discovered, one must back up to a previously unconsidered node to continue the search. The clique adjacency matrix may be created as each maximal clique is discovered by iterating through the matrix and counting the overlapping nodes for each existing clique with the newly discovered clique.

3.9.3 Complexity

The first step of the CPM algorithm works to find all maximal cliques for a particular graph. Finding a clique of a specific size k has been shown to be NP-Complete by Karp [62]. Further, finding maximal cliques is also proven to be unapproximable by Zuckerman [130]. However, once the maximal clique adjacency matrix is created, it is trivial to analyze different clique sizes as the k-clique information for all sizes is contained in the matrix. Palla *et al.* offer a running time of under 2 hours to analyze the Arxiv[11] co-authorship data for around 127,000 links using a commodity desktop PC at the time of writing in 2005 of their supplemental information for [93].

3.9.4 Applying the Clique Percolation Method to Reputation

As mentioned previously, Java *et al.* [57] apply the undirected algorithm as described in this section to Twitter relationships, and follow up with analyzing the top keywords that appear in the different communities. This technique of performing statistical analysis to analyze the trends within the community is one of assessing group reputation, and therefore the reputation of a user as it is inherited from the group with which that user associates. As with all other community detection methods, the statistical analysis may be applied to any attribute or combination of attributes within the community to determine other information such as common language, location, and many other details.

However, the CPM algorithm suffers the same issue as the Betweenness Centrality method described in section 3.7 as the running time is too long to analyze large networks. The algorithm will still be useful if the network is pruned to a sub-network to perform the analysis. The largest

[11] http://arxiv.org

benefit of CPM is the fact that users may belong to multiple k-cliques (communities) which allows for a more accurate analysis of attribute data for users within a community compared to the other agglomerative and divisive algorithms such as those described in sections 3.7, 3.8, and 3.13 that completely segregate nodes into distinct, non-overlapping communities.

3.10 Nearest Neighbor Networks

The Nearest Neighbor Networks (NNN) method, proposed by Huttenhower *et al.* [54], is a divisive graph algorithm which clusters genes into neighborhoods of genes where they are the most similar and fully interconnected to each other (in k-cliques). This method differs from other clustering methods in that it combines a similarity measure of a graph vertex (in this case genes) in addition to network connectivity to determine communities.

3.10.1 Related Work

The NNN method is from the world of biology and genetics which employ many clustering techniques to analyze different gene relationships. Huttenhower *et al.* compare their algorithm to others that are typically used in the field including Aerie, CAST, CLICK, GenClust, Quality Threshold Clustering, SAMBA, and k-means[12] [54]. In their analysis, Huttenhower *et al.* use Pearson's Correlation Coefficient (as described in section 3.2.2) as well as euclidian distance for the similarity measure calculation.

3.10.2 Algorithm

Here I describe the algorithm by Huttenhower *et al.* [54] written in the context of social network analysis. The algorithm works on a graph $G = (V, E)$ with the following elements:

- $d(v_i, v_j)$ - A similarity measure which returns the similarity value of two vertices v_i and v_j,
- g - The size of k-cliques[13] to be found, and

[12] K-means clustering is described in section 3.5.
[13] For a description of k-cliques, review section 3.9.2.

- n - The size of the neighborhood for a vertex.

The algorithm then occurs in five major steps:

(1) Calculate the similarity measure of a vertex and all its neighbors and remove all links except for the top-n nearest (most similar) neighbors,

(2) Remove all links from the graph that are not bi-directional (and remove all nodes that no longer have any links),

(3) Find all of the k-cliques of size g in the undirected graph

(4) Overlapping k-cliques are merged (defined as ***k*-clique chains** in section 3.9.2) to produce initial networks, and

(5) Initial networks containing cut-vertices (vertices that are shared between overlapping k-clique chains) are split into final networks with the cut-vertices occupying both networks.

3.10.3 Complexity

As mentioned in section 3.9.3, finding cliques in a graph is either NP-Complete or unapproximable. Huttenhower *et al.* mention that their algorithm runs with a lower-bound of 2.5 minutes (with values of $g = 3$ and $n = 10$) and an upper-bound of 11.5 minutes (with values of $g = 5$ and $n = 40$) on a single-threaded 2GHz machine processing their largest data set of 6153 genes across 300 genetic conditions at the time of writing in 2007 [54].

3.10.4 Applying Nearest Neighbor Networks to Reputation

This method has many features which are of interest when analyzing social networks. First, this method combines a similarity measure with network connectivity which gives us the freedom to define "similarity" for my analysis. Since a vertex is a user with a profile and a history of messages, I can define similarity as profile attributes or topics about which people are discussing. Second, this method specifically allows a single vertex (or user) to belong to multiple communities, unlike many other methods such as Betweenness Centrality (described in section 3.7), Global Hierarchical Clustering (described in section 3.8) and Local Hierarchical Clustering (described in section 3.13) which only allow a vertex to belong to one community.

However, even given the benefits of the algorithm, it also suffers the same issue as the Clique Percolation Method (described in section 3.9) as it uses k-cliques for discovering communities which is very slow for small graphs and practically intractable for large graphs. Again, I would need to approximate a smaller portion of the social network being analyzed to be able to use this method.

3.11 K-core Decomposition

The k-core decomposition method, proposed by Seidman [106], is a clustering method which ranks vertices based on how closely they are connected to the *core* of the network. A *k-core* subgraph of a larger graph G contains the set of vertices that have at least degree k after all other vertices with degree less than $k - 1$ have been recursively removed from the graph G.

3.11.1 Related Work

Batagelj and Zaveršnik [11] propose a fast algorithm for computing the k-cores of a graph. Chun *et al.* [23] calculate the k-cores of the collected graph of the Cyworld social network in South Korea to observe how strongly connected are people of similar degree. Alvarez-Hamelin *et al.* [2] analyze Internet autonomous system (AS) cross-connectivity data from the skitter project at CAIDA and the Distributed Internet Measurements and Simulations (DIMES) project using k-core compared to the edge betweenness method (discussed in section 3.7). Carmi *et al.* [18] modify the k-core decomposition to create *k-shells* which are groupings of multiple k-cores to determine the functional roles of the autonomous systems from the DIMES data set. Kitsak *et al.* [64] compare k-shell decomposition with the Susceptible-Infectious-Recovered (SIR) epidemiological model [63] as a means to reasonably determine the influence a specific vertex has in a network and applies the algorithm to the LiveJournal.com community, a network of email contacts, the contact network of inpatients (CNI) from hospitals in Sweden, and the actor network from the Internet Movie Database

(IMDB).

3.11.2 Algorithm

Here I will describe at a high-level the algorithm from Batagelj and Zaveršnik, and the full description can be found in [11]. Algorithm 1 describes the process for computing the cores of a network.

Algorithm 1 $O(m)$ cores decomposition of networks [11]

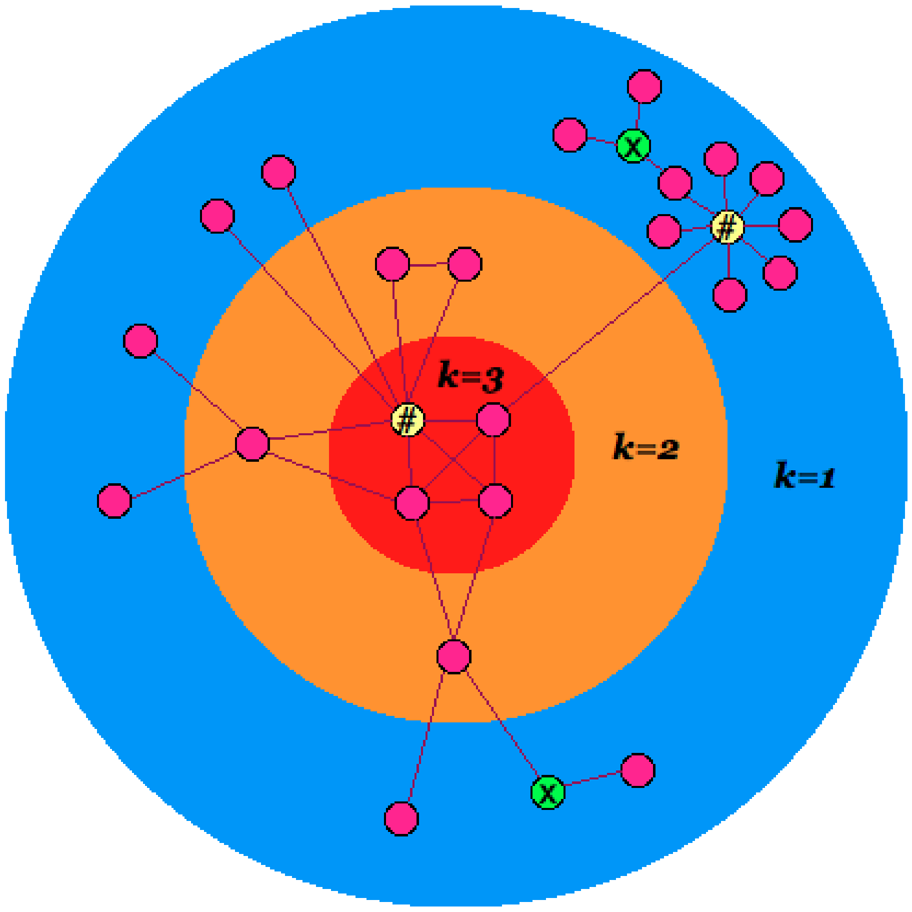
```
Given a graph $G = (V, E)$ as a list of neighbors
Map $core = \{\}$
Compute the degree $d$ for each vertex $v \in V$
Order set of vertices $V$ in increasing order of their degrees
for each $v \in V$ in order, do
    $core[v] = d(v)$
    for $u \in Neighbors(v)$ do
        if $d(u) > d(v)$ then
            $d(u) = d(u) - 1$
            Reorder vertices $V$
        end if
    end for
end for
```

Figure 3.2 is a visualization of an example network that is split into three k-cores. Note that the vertices marked with an **x** both have degree $k = 8$ but belong to different k-cores. This is due to the fact that as vertices with degree of $k = 1$ are removed from the graph, the **x**-labeled vertex at the top-right only has a single connection (degree $k = 1$) to the core of the network, and therefore belongs to **k**-core $k = 1$. Other interesting vertices are denoted with a pound (#) which are vertices that also have a degree greater than 1, but belong to the k-core $k = 1$ due to their neighbors being removed and leaving them with only one connection to other k-cores.

3.11.3 Complexity

The running time of the Batagelj and Zaveršnik algorithm in [11] is $O(m)$ which is the number of edges in the graph due to time to sort the vertices by their degrees. The storage requirements of the algorithm are steep as two arrays of size n must be kept, as well as an array of the starting

Figure 3.2: An illustration of the k-cores decomposition of a network [64]. Note that vertices marked with # have the same degree $k = 8$ but are not in the same core. Further, vertices marked with **x** have degree $k > 1$ but are still considered to be in core $k = 1$.



position of each vertex of increasing degree. If this algorithm is run on a large graph, a significant amount memory is required.

3.11.4 Applying K-Core Decomposition to Reputation

The speed of this algorithm makes it very appealing for social graph analysis. As previously mentioned, Kitsak *et al.* [64] compare k-shell decomposition (a variant of k-cores) to the SIR epidemiological model and argue that k-cores are a good means to determine the influence a user has in the network. More specifically, k-cores are a good measure of how likely a piece of information spread by a user in a high-value k-core will reach all other users in the network. This measure will be very useful when identifying who is at the center of a group during an event.

Further, Batagelj and Zaveršnik [11] mention that this technique may be used to prune vertices from consideration when attempting to find k-cliques of a specific size. For example, if a vertex belongs to a k-core of size 4, it may never be a part of a k-clique of size 5. This heuristic may speed up the running time of finding cliques as used by the Clique Percolation Method (described in section 3.9) and the Nearest Neighbor Networks method (described in section 3.10).

3.12 Activity Network

The Activity Network[14] method is a clustering technique that prunes edges from a social graph based on user interactions with one another. The result is a subgraph (usually very weakly connected and disjoint) that represents only those relationships that involve some sort of direction activity between users.

3.12.1 Related Work

The idea of the Activity Network has been proposed in multiple ways. Huberman *et al.* [52] create a directed activity network of users in the Twitter service by considering a user A to be a "friend" of user B by user B sending at least two directed messages to user A. Wilson *et al.* [127]

[14] An alternate name proposed by Wilson *et al.* [127] is *interaction graph*, but is functionally equivalent.

combine an activity rate as well as the number of directed messages between two users (both with varying values) to create an undirected activity network of users in the Facebook service. Chun *et al.* [23] analyze the Cyworld service and create an activity network using weighted vertices where weights are calculated as the sum of all outgoing messages to other users. Unlike the work by Huberman *et al.* and Wilson *et al.*, this analysis does not involve determining which users are friends by interaction, but instead uses the weights to do general analysis on the overall network.

3.12.2 Algorithm

Here I will describe the process for all three of the methods mentioned above. The first I will describe is from Huberman *et al.* [52]. They create the activity network based on messages sent directly from one user to another. However, the Twitter service did not originally have support for directed messages, so users began messages with what is known as a *mention*[15] among Twitter users. These *mentions* are placed at the beginning of a message and denote a directed message to another user, although every follower of the broadcaster is still able to see the message. Huberman *et al.* simply prune all of the "friend" or "follower" links of users that do not have at least two directed messages to one another.

Wilson *et al.* [127] perform a similar pruning of links as Huberman *et al.*, however there are two important differences. First, Facebook relationships are undirected, so two users are only considered to be "friends" in the activity network if they both send at least one message to each other. Twitter relationships are directed, meaning they do not have to be reciprocated. Second, an interaction rate is added to the pruning of links. Two parameters n and t are considered which are the minimum number of interactions and a window of time, respectively. With both metrics, a link is in the activity network is only retained if both users have sent at least one message to each other and have done so within a specific window of time.

The creation of the activity network by Chun *et al.* [23] is based on weighting each vertex in the social graph by their "strength". The strength is the count the number of outgoing messages

[15] A *mention* is when a username is placed in a message prepended by the "@" symbol.

to declared friends in the social graph. Once node strength is determined, links may be pruned at different thresholds to create a subgraph of users who use the network with a specific frequency.

3.12.3 Complexity

The method proposed by Huberman *et al.* requires three major steps: downloading of the social graph, downloading of the tweets by each user in the graph, and parsing of each tweet for those that begin with the *mention* format. If I disregard the time needed to collect the data from Twitter, the process of pruning edges requires $O(n)$ time where n is the total number of tweets downloaded.

The method proposed by Wilson *et al.* also requires collection of the posts from Facebook before analysis is possible.[16] If I also ignore the time needed to collect the information, determining the interaction rate can be done in the same pass as determining messages between users which only takes $O(n)$ time.

The method proposed by Chun *et al.* simply counts the number of messages sent by users in the network. This process, like the other two, simply requires a single pass through all messages by users which requires $O(n)$ time.

3.12.4 Applying Activity Networks to Reputation

The activity network is the basis of potential reputation between users. Since I am using implicit means of inferring reputation, user actions that directly involve other users, I may have an indicator that the user who sends the message demonstrates the importance of the target user. During a specific event, one way to determine clusterings of users will be to observe the messages sent between users. This method may be modified an extended in a number of ways, including using all *mentions* in the Twitter service (not just mentions that begin the messages) to determine which users are becoming important to that event.

[16] The data analyzed by Wilson *et al.* in [127] was collected by accessing regional networks which no longer exist in Facebook [33].

3.13 Local Hierarchical Clustering

The Local Hierarchical Clustering method, proposed by Clauset [24], is a graph community detection method. Clauset describes the major limitation of other community detection algorithms, which is the requirement of having global knowledge of the network structure in order to detect communities. This requirement is impractical when analyzing networks where the full structure cannot be known (such as the web), networks that may be too large for even the fastest agglomerative algorithms, or networks that require a lot of time to collect data (due to bandwidth or API limitations). The method includes a local modularity calculation in order to determine good clusterings without the need for global knowledge of the network. The method attempts to detect communities from the perspective of a single vertex in the entire network, radiating outward, and is in the class of "agglomerative" algorithms (such as Global Hierarchical Clustering described in section 3.8) as it adds vertices and edges to a graph to determine community structure rather than removing vertices or edges. The local nature of this algorithm implies the detection of concentric communities, or communities that are subsets of larger and larger single encompassing communities as the network is explored.

3.13.1 Related Work

Other researchers have investigated means of detecting local community structure to overcome the global knowledge requirements as previously described. Luo *et al.* [74] attempt to search a graph for a local module by both adding and removing vertices to maximize internal links over external links. Clauset [24] uses a similar modularity metric as Luo, but does not remove vertices to maximize the modularity value. Bagrow and Bollt [7] use a breadth-first search method to expand a community shell outward until ratio of in-degree vs. out-degree has reached a user-defined threshold. Mislove *et al.* [81] propose a metric called Normalized Conductance, which is almost identical to the local modularity measure proposed by Clauset in [24] but differs only in that the Normalized Conductance value is weighted against a random graph.

3.13.2 Algorithm (Clauset)

Clauset [24] proposes an algorithm that attempts to minimize arbitrary expansion by visiting local neighbors that have the lowest degree, thus helping to find the best local community before moving outward to other unvisited nodes. The algorithm includes a "local" modularity calculation similar to that of the modularity metric Q from the Global Hierarchical Clustering method (described in section 3.8) as it is a measure of how optimal a particular division is when segregating nodes into communities. The main difference is Clauset's modularity calculation only depends on the immediate neghbors of known vertices instead of the entire graph.

3.13.2.1 Local Communities and Neighborhoods

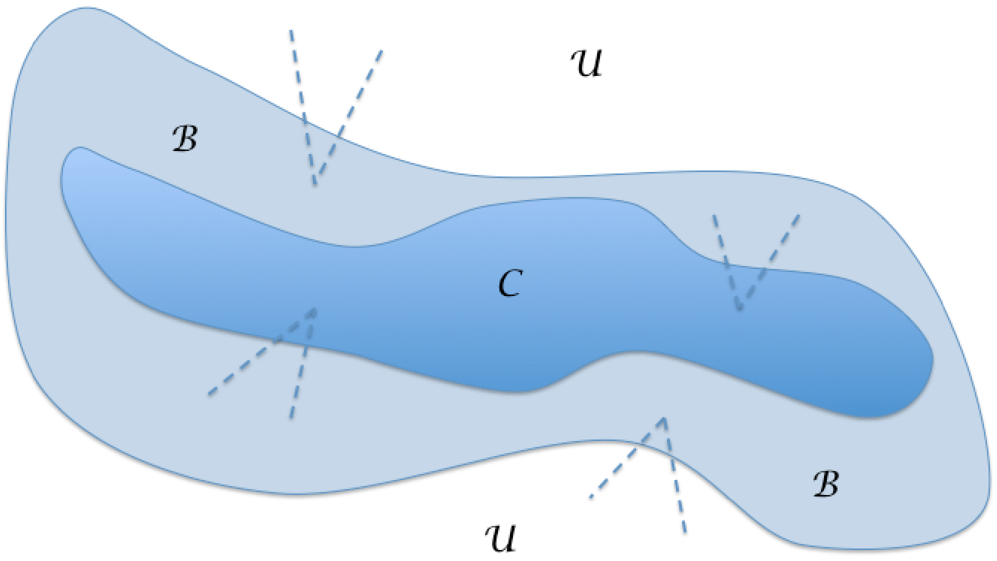
The algorithm assumes that I have a set of known vertices $\mathcal{C}$ of which I have perfect knowledge in the graph $\mathcal{G}$. The implication is there exists a set of vertices $\mathcal{U}$ about which I only know are adjacent to vertices in $\mathcal{C}$. Clauset further subdivides the vertices in community $\mathcal{C}$ to the following groups, which are all illustrated in figure 3.3:

- The *boundary* vertices $\mathcal{B}$ which are the vertices that are connected to both the known vertices in $\mathcal{C}$ and the unknown portion of the network $\mathcal{U}$, and
- The *internally connected* vertices $\mathcal{C}$ that are fully connected only within the known group of vertices (vertices that do not have neighbors in the unknown portion $\mathcal{U}$)

3.13.2.2 Local Modularity

The existence of boundary $\mathcal{B}$ of the community as described above allows us to calculate a "sharpness" of the boundary of the known community $\mathcal{C}$. The idea is that a "sharp" boundary will have few connections from the neighborhood $\mathcal{B}$ to the unknown portion $\mathcal{U}$ of the network. The adjacency matrix for the boundary $\mathcal{B}$ is given as the following:

Figure 3.3: An illustration of the groupings of a network into the local community $\mathcal{C}$, its boundary $\mathcal{B}$, and the edges which connect $\mathcal{B}$ to the unknown neighbors $\mathcal{U}$ [24].



$$B_{ij} = \begin{cases} 1 & \text{if nodes } i \text{ and } j \text{ are connected, and either node is in } \mathcal{B} \\ 0 & \text{otherwise.} \end{cases} \tag{3.22}$$

The local modularity value R is now defined to be the following:

$$R = \frac{\sum_{ij} B_{ij}\delta(i,j)}{\sum_{ij} B_{ij}} = \frac{I}{T} \tag{3.23}$$

The above function $\delta(i,j)$ is 1 when either $v_i \in \mathcal{B}$ and $v_j \in \mathcal{C}$ or vice versa, and is 0 otherwise. Said differently, $\delta(i,j)$ is 1 when an *edge* in the network crosses the boundary of $\mathcal{B}$ into $\mathcal{U}$. Here T is the number of edges with one or more endpoints in $\mathcal{B}$, while I is the number of edges with neither endpoint in $\mathcal{U}$ (the edges that are fully contained in $\mathcal{C}$).

3.13.2.3 Exploring the Unknown Nodes in $\mathcal{U}$

The only way in which I gain additional knowledge of the graph $\mathcal{G}$ is by visiting a neighboring vertex $v_i \in \mathcal{U}$, which yields a list of adjacencies. Since I only want to add vertices that give the largest increase (or smallest decrease) in the modularity R, I will need to calculate the change in modularity, ΔR, for a vertex $v_i \in \mathcal{U}$. The computation for ΔR is derived from equation 3.23 is given as the following:

$$\Delta R_j = \frac{x - Ry - z(1-R)}{T - z + y} \tag{3.24}$$

Once a vertex v_i yielding the largest increase or smallest decrease in R is discovered, v_i is added a member of $\mathcal{C}$ and additional unknown adjacent vertices of v_i may be added to $\mathcal{U}$. The pseudo-code for the algorithm is described in algorithm 2. Figure 3.4 also shows the modularity values for the Amazon recommender network for two books and a music CD.

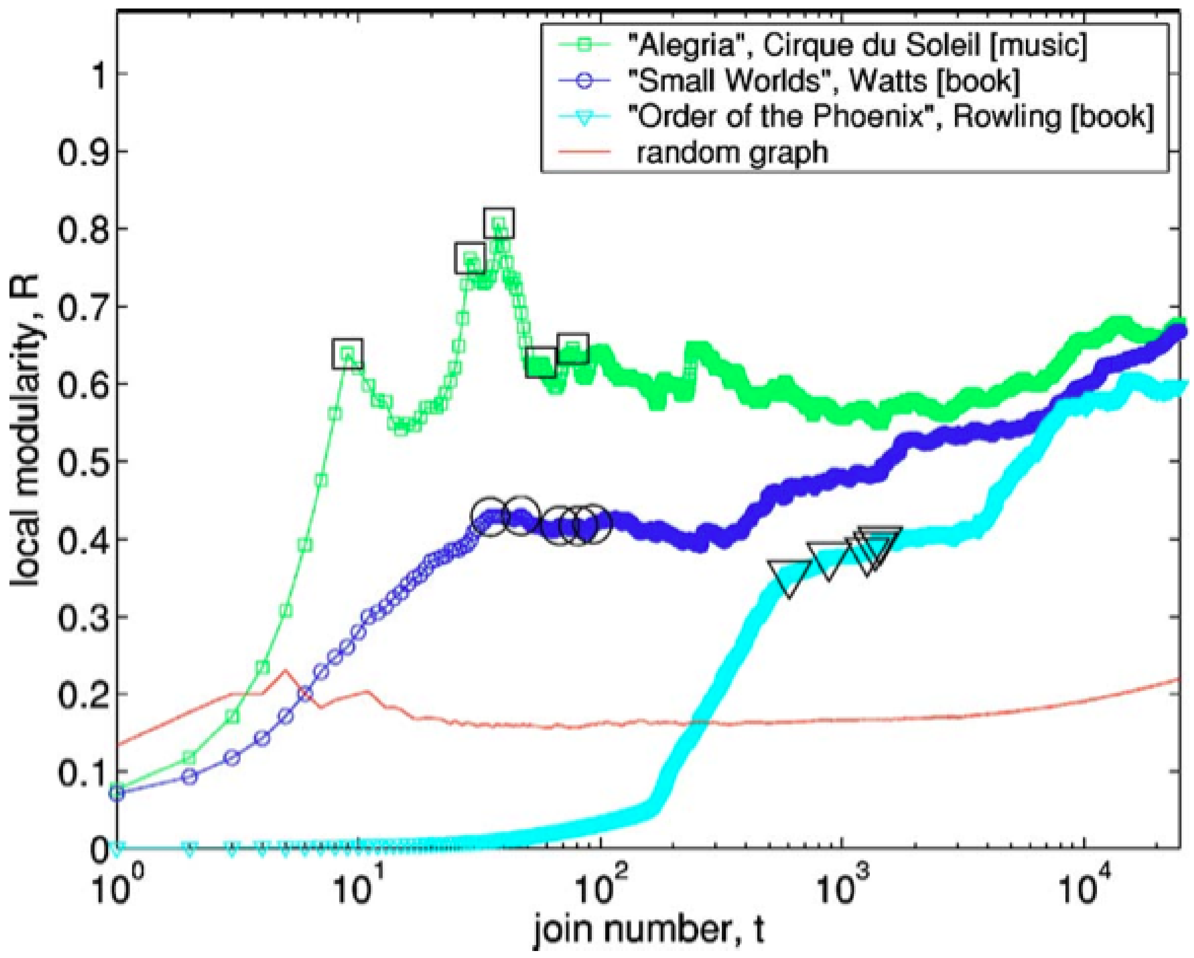
3.13.3 Complexity

Each calculation of ΔR requires iterating through all nodes in the unknown portion $\mathcal{U}$ of the network. If I choose a maximum size k for community $\mathcal{C}$ with an average degree of d, then

Algorithm 2 Local Hierarchical Clustering (Clauset [24])

```
add $v_0$ to $\mathcal{C}$
add all neighbors of $v_0$ to $\mathcal{U}$
set $\mathcal{B} = v_0$
**while** $|\mathcal{C}| < k$ **do**
   **for** each $v_j \in \mathcal{U}$ **do**
      compute $\Delta R_j$
   **end for**
   find $v_j$ such that $\Delta R_j$ is maximum
   add that $v_j$ to $\mathcal{C}$
   add all new neighbors of that $v_j$ to $\mathcal{U}$
   update $R$ and $\mathcal{B}$
**end while**
```

Figure 3.4: Local modularity R for three items in the Amazon recommender network, shown on log-linear axes. For comparison, the time series for a random graph with the same degree distribution is shown. The large open symbols indicate the locations of the five strongest enclosing communities [24].



the calculation of ΔR is roughly $O(kd)$ if the network data exists locally through prior collection. Therefore the overall running time for the algorithm for all nodes in the network in general is $O(k^2d)$ or $O(k^2)$ for a sparse network. However, as observed by Clauset, for networks that exist on the web, the running time is dominated by time to retrieve the data from the online source, resulting in a linear running time of $O(k)$. The value of k is not the entire size of the network, but only that portion of the network that is considered when detecting a community.

3.13.4 Applying Local Modularity to Reputation

As a community detection algorithm that performs with limited knowledge of the overall network, local modularity clustering is well-suited for use when operating under constraints for data collection from a social network. As previously mentioned, the running time $O(k^2d)$ is bound by the number of vertices that are evaluated for community division for the subgraph that has previously been collected. Thus I am able to control how long the algorithm runs and still obtain useful output.

However, one major limitation of the local method for hierarchical clustering is that I cannot infer any local information for users that are discovered in super-set communities far from the originating vertex. This implies that determining the immediate local community for an arbitrary user will require a separate run of this algorithm for that user. However, users of interest determined from other methods may be used to seed the collection for this algorithm, which subsequently could be used to see whether or not the users of interest belong to the same or neighboring communities in the network graph.

3.14 Hyperlink-Induced Topic Search (HITS)

The hyperlink-induced topic search (HITS) algorithm, proposed by Kleinberg [65], was developed to solve the problem of many early web search engines where "relevant" sites were ranked simply by the frequency of the specific keyword on which the search was performed. As the amount of websites grew (and still grow today), the number of search results have also increased to the point where there are too many results for the human reader to consume. Further, many websites began injecting multiple keywords in their HTML documents in order to inflate their rank in these original search engines and appear high on the search result list.

The HITS algorithm improves the search results for these search engines by also considering the link structure of the websites returned by these search engines. The goal is to determine the set of pages that are the "authorities" (the most relevant pages) for the keyword, as well as the pages that are "hubs" (the pages that link to many related "authorities").

3.14.1 Related Work

Kleinberg [65] first proposed the HITS algorithm and modifications to the algorithm were developed by Bharat and Henzinger [12] to solve the problem of "mutually reinforcing relationships between hosts" (one page on site A links to many pages on site B, or many pages on site A link to one page on site B) and the problem of "topic drift" (highly-ranked authorities and hubs are not about the original search query). Further improvements on the Bharat and Henzinger algorithm were developed by Li *et al.* [72] and Asano *et al.* [4]. HITS and its improvements have been applied many times to the web link structure, but has also been applied to social networks to determine the "authorities" within the users of the Twitter social network by Java *et al.* [57] (although without any keyword search context for the users). Romero *et al.* [101] apply the HITS algorithm to predict the popularity of links shared and re-tweeted in the Twitter social network.

3.14.2 Algorithm

Here I will describe the original algorithm by Kleinberg in [65]. The algorithm fundamentally solves for the principle eigenvector of two matrices using the power method after initial data collection. The algorithm occurs in three major steps:

- **Construct a focused subgraph** - Collect the t highest-ranked pages (typically $t = 200$) for a query σ from an existing web search engine. This set of pages is the root set R_σ.

- **Expand the root set of search results** - The expanded base set S_σ is created by adding any page pointed *to* by a page in R_σ and any page that points to a page *in* R_σ. Note: a restriction is that a single page in R_σ is only allowed to bring in at most d pages pointing to it (typically $d = 50$) into S_σ. Algorithm 3 describes the process for this step.

- **Iteratively compute hub and authority values** - For each page $p \in S_\sigma$, compute the hub and authority values k times (typically $k = 20$) until the values converge. Algorithm 4 describes the process for this step.

Algorithm 3 HITS Base Set Collection from Kleinberg [65]

```
Given the following:
Search results $R_\sigma$
In-degree limit $d$
Set $S_\sigma := \{\}$
**for** each page $p \in R_\sigma$ **do**
   Let $\Gamma^+(p)$ denote the set of all pages $p$ points to.
   Let $\Gamma^-(p)$ denote the set of all pages pointing to $p$.
   Add all pages $\Gamma^+(p)$ to $S_\sigma$.
   **if** $|\Gamma^-(p)| \leq d$ **then**
      Add all pages in $\Gamma^-(p)$ to $S_\sigma$
   **else**
      Add an arbitrary set of $d$ pages from $\Gamma^-(p)$ to $S_\sigma$
   **end if**
**end for**
Return $S_\sigma$
```

3.14.3 Complexity

The first step of the algorithm requires data collection from a separate source (in the case of the original algorithm, the data were web pages). The time complexity for this step is bound by the time to collect the data from the web. The space complexity for this step is simply the storage of each web page as an identifier, as well as the list of links on the page itself. Due to the variable nature of web pages, the storage needs may only be guessed initially.

The second step of the algorithm requires querying the search engine used to find the initial root set to obtain the list of pages that point to all the pages in the root set R_σ. Further, another

Algorithm 4 HITS Hub and Authority Computation from Kleinberg [65]

```
Given the following:
Base set $S_\sigma$ of $n$ pages
Iteration limit $k$
Let vector $z = (1_0, 1_1, ..., 1_{n-1})$.
Set initial authority values vector $a := z$.
Set initial hub values vector $h := z$.
**for** $i = 1, 2, ..., k$ **do**
  **for** each page $p \in S_\sigma$ **do**
    Calculate authority score as follows:
    **for** each page $p'$ that points to $p$ **do**
      Set $a_p = \sum_{p'} h_{p'}$
    **end for**
    Calculate hub score as follows:
    **for** each page $p'$ which is pointed to by $p$ **do**
      Set $h_p = \sum_{p'} a_{p'}$
    **end for**
  **end for**
  Normalize $a$ and $h$ values so that $\sum_j a_j^2 = \sum_j h_j^2 = 1$.
**end for**
Return $(a, h)$.
```

query must be made to collect all the pages to which the root set R_σ point. This can be done by either querying the search engine or following the outgoing links that exist in the pages in the root set.

The third step of the algorithm iterates through the links to and from each page in the base set S_σ and calculates hub and authority values, which means the running time for this step is $O(nd)$ where n is the number of pages in the base set S_σ and d is the average total degree (in and out) of all pages in the result set S_σ. The hub and authority values are then normalized for all pages which results in a running time of $O(n)$ for both values. Finally, this step iterates k times through the pages, so the overall average case running time for the algorithm is $O(knd)$.

3.14.4 Applying HITS to Reputation

The hubs and authorities calculation of HITS for web pages is a direct calculation of eigenvector centrality based on how neighbors link to the page in question, or how the page in question links out to neighbors in order to determine the importance of a single page relative to all other pages in the base set S_σ. However, when considering applying the algorithm to a social network, the situation is not directly analogous. First, the algorithm attempts to remove intrinsic links from web pages that point to themselves in order to prevent accidental inflation of a hub or authority score. If I consider the explicit link graph (the "friends" links) in online social networks, users are not allowed to make links to themselves, and are only allowed to make a single connection to another user, which negates the need to worry about intrinsic links or multiple transverse links. This eases the work done by the algorithm by skipping this pruning process in the second step of the algorithm where the root set R_σ is expanded to the base set S_σ. Second, the original HITS algorithm relies on an existing search engine which only exists in limited form from specific social networks, an example being Twitter. As mentioned previously, Java *et al.* [57] applies the HITS algorithm to Twitter, but does not mention any use of keyword search to obtain the seed set of users to begin the hubs and authorities calculation.

The Twitter service, for example, does allow keyword search on the service via an application programming interface (API), but the results are returned in no particular order as a sampling of the overall posts that contain the keyword in question. Therefore any hub or authority calculation for search results may produce very disjoint hubs and authorities that are not related to the keyword in any way. Instead, this method may be more applicable to determine the influencers within a particular community or within multiple communities.

Another property of the algorithm is that all pages which are computed to be of high important are assumed to be all related to each other. However, social networks differ from web pages considerably in this regard where as web pages typically only speak to specific topics, users in a social network have very diverse interests and connections to others that equates to having many topics belonging to a single user. This is an important distinction when analyzing nodes in a social network using algorithms used to analyze the web.

A potential modification to this method that may yield more interesting results for determining influencers is to observe the *change* in hub and authority values over time for a particular user or set of users. A change in hub or authority values may indicate a change in reputation which may or may not be related to an event (disaster or otherwise). Further, different input graphs may be created based on the interaction between users instead of simply their declared social network in order to cull out extra users who may not be important in the network.

3.15 Summary

In this chapter I have investigated a number of analysis methods and tools that may better inform us as to the form and function of user behavior as it pertains to inferring reputation. Table 3.2 is a matrix of analysis methods and tools versus data attributes on which they operate.

3.15.1 Analysis Methods and Tools Not Described

Many analysis methods and tools exist for investigating the data available to us in social networks, and I have covered a number of tools that demonstrate the most promise for inferring reputation. However, other tools are available that I did not cover but warrant further review.

Pizzuti [96] proposes a method to detect overlapping communities, the same goal as that of the Clique Percolation Method described in section 3.9. Van Dongen [119] proposes a method to cluster vertices in a graph based on stochastic flow simulation. Bagrow [6] proposes a local community detection method in the same family as the local method described in section 3.13. Many of the clustering methods that have been inspired from biology and genetics compared by Huttenhower *et al.* in have not been included in this analysis. Flake *et al.* [36] propose a community detection algorithm that is based on max-flow of vertices in the graph. Chun *et al.* apply the *disparity* metric to the social graph in Cyworld to show the spread of activity of a user over all friends. Blei *et al.* [14] propose an unsupervised topic modeling method to determine groupings of words as topics.

Table 3.2: Matrix of analysis methods and data attributes

Methods / Attributes	Keywords/topics	Phrases	URLs	Duplicates	Mentions	Timestamps	Data source	Location	Language	Community	Relationship similarity	Influence/Rank	Connectivity
Statistics/probability	X	X	X	X	X	X	X	X	X				
Network motifs										X	X		X
Similarity measures	X	X	X	X	X	X	X	X	X	X	X	X	X
TF-IDF	X	X	X	X	X								
LDA	X	X											
N-grams	X	X	X	X	X								
K-means	X	X	X	X	X	X	X	X	X	X	X	X	X
PageRank										X		X	
Betweenness Centrality										X		X	X
Global Hier. Clust.										X		X	X
Clique Percolation										X			X
Nearest Neighbor										X			X
K-core Decomp.												X	X
Activity Network											X	X	X
Local Hier. Clust.										X			X
HITS	X	X								X	X	X	X

Chapter 4

Twitter Data

The research performed in this thesis uses public data available from the Twitter social network. In this chapter I give an overview of the data that exists in Twitter and describe the limitations of the data as well as the limitations of collecting the data. We will also show some general statistics of a sample of the data.

4.1 Twitter API and Data Formats

The Twitter social network is available as a web interface in which people are able to login and post messages and read the messages of those users they follow. However, for the purposes of this research, I need access to large amounts of data that are cumbersome to collect via a web crawler. Twitter does provide an application programming interface (API) that allows for direct access to the data via a collection program.[1]

The API available from Twitter uses a RESTful[2] interface that is easily accessible through all major programming languages and some command-line unix tools. In section 3.1 I described the different types of data available in social networks. The types of data available from the Twitter service include the three major categories of message and profile text, associated metadata, and the declared social graph between users.

The data from each API call may be returned in varying formats specified by the developer.

[1] http://dev.twitter.com

[2] http://en.wikipedia.org/wiki/Representational_State_Transfer

Those formats include JavaScript Object Notation[3] (JSON), Atom Syndication Format[4] (ATOM), and Extensible Markup Language[5] (XML). Each of these formats may be parsed to extract the data from the query for use in my analysis.

The Twitter API consists of three distinct types:

(1) Search API - Allows for searching for tweets that match a set of keywords back in history

(2) Streaming API - Allows for searching for tweets that match a set of keywords as the tweets are created by users real-time

(3) REST API - Allows for querying for tweets, user profiles, and the social graph matching specific values

The search API has undergone many changes during its existence, originally starting as a company acquisition by Twitter, and most recently it has been migrated to using a full Lucene backend [117]. Tweet results from search are always returned in decreasing order based on time which has effects on what information users are most likely to see when they search for information on the network. The streaming API was developed as a means to obtain tweets matching keywords, user IDs, and location polygons [116]. The REST API allows access to data using queries that are not based on keyword. There are numerous methods that exist that allow access to all data using user IDs, usernames , timestamps, and others. Examples include queries to gather a user's full tweet stream history or friends and followers.

4.2 Collection Limitations

There exist a number of limitations when attempting to collect data from any of the three above API interfaces. Here I describe each of these limitations.

First and foremost, Twitter sets rate limits based on both authenticated users accessing the service as well as un-authenticated requests from any public IP address [114]. These limits have

[3] http://tools.ietf.org/html/rfc4627, http://www.json.org
[4] http://en.wikipedia.org/wiki/Atom_%28standard%29
[5] http://en.wikipedia.org/wiki/XML

varied over time, but typically default values are 350 requests *per hour* for an authenticated user and 150 requests per hour for un-authenticated requests from a specific IP address. These limits may be raised to "whitelisted" status by contacting Twitter for such a status. These requests require justification, and in fact for this research, Twitter has granted five IP addresses whitelisted status. Whitelisted status may apply to both authenticated requests and IP addresses, effectively raising the rate limits for both to 20,000 requests per hour. Further, many requests for information (such as tweets or friends/followers) require multiple individual API calls in order to gather all the desired information. For example, Twitter will return a maximum of 5000 user IDs per friend or follower requests as a "page" of data. This implies that if a user has say, 13,000 followers, three total requests must be made to collect all of this data, and each individual page request deducts from the overall allowed limit per hour. Additionally, Twitter currently has a limitation of only exposing the most recent 3,200 tweets posted by a user [118].

Another limitation in accessing data from Twitter appears in the network latency per query. Query response times vary depending on the type of query and the time of day the query is made. For example, a query for a user's latest tweet may be modified to include or exclude the user profile in the response. If the profile is included then the response time is greater due to the extra database query on the Twitter servers to obtain this extra information to be returned in the response. The time of day of the query also affects the overall response time due to standard internet latency times that vary with peak usage during the day versus the night. Our observed query response times have ranged from 200-500ms due to the described factors, and to fully exhaust 20,000 requests per hour requires one request every 180ms. In order to fully utilize these resources, I must incorporate multi-threading to reduce the amount of time waiting due to network latency.

The final limitation I encounter includes errors returned by Twitter for valid queries. Twitter defines many of their own HTTP status codes for errors [115] and many are encountered when attempting to query all types of data in Twitter. The streaming API nominally will stay open until the client closes the connection, but occasionally Twitter will return an error, or there will be other local networking issues that cause the connection to drop. We must write in logic to re-connect to the streaming API when a connection drops. Also in the search and REST APIs, Twitter will occasionally throw one of it's 500-level response errors, and in some rare cases, will completely hang the connection. Further, there are other factors which may cause other 400-level requests, such as a user changing their username (the user ID is the only attribute that is immutable), or by a user account being suspended due to terms of service violations. I must account for all of these errors when trying to collect the data.

4.3 Twitter Data and Attributes

Assuming I have accounted for all of the above described limitations in collecting the data, I must now understand what data I have exactly, as well as the limitations of that data and what I can do to infer missing information using additional collection techniques. The three major types of data obtained from Twitter are the user profiles, tweets generated by users, and the social graph friends and followers connections for users. Here I give real-world examples of each of these data.

Table 4.1 contains all of the key/value pairs of attributes for user "schenkmanus". The bolded attributes are the ones used in this research to determine influence and reputation. The user ID is one of few immutable attributes of a user and is what I use for the primary key when tracking users. We are also interested in the friends and followers counts for various reasons will will be described in section 4.5.2. The screen name attribute I also care about since that is how most average users find others, even though the name can be changed at any time. The time zone/UTC offset are also an interesting attributes as they may tell us a general area in which a user may reside. The location field may also tell us where a user may be, depending on whether or not that user has entered in a valid location.

Table 4.2 contains all the key/value pairs of attributes for the latest tweet by user "schenkmanus". The bolded attributes again are the ones used in this research. Three similar attributes appear in the tweet which are "coordinates", "place", and "geo". These three attributes are various ways in which Twitter includes geographic information such as lattitude/longitude for tweets if the user who posted that tweet enabled GPS or a related feature on their device used at that time. We also see the actual tweet text included, as well as the timestamp of that tweet. Additionally we also have the source application (such as the web or various phone applications) that people use to post messages.

Table 4.3 contains two lists representing the social graph of user “schenkmanus”, one of friends (out-going links made to other users) and one of followers (in-coming links made by other users to this one). Two separate REST API calls must be made to gather each of these lists. Note that the information returned for these requests is minimal, only including a list of users. This has implications which I will describe in section 4.4. In this case user “schenkmanus” has very few friends and followers which is not the average [57].

Already one can see that there are a number of attributes that are not necessary but are included in all responses anyway. Some attributes are even duplicated across multiple queries. In section 4.5.2 I look at the attributes more in depth.

4.4 Data Limitations

A number of limitations appear when analyzing the data returned from Twitter. At a first observation, Twitter stores very little information about users in general, with only giving fields for time zone, location, name, and description as identifiers of the user. This is in comparison with, say, Facebook that contains numerous fields for users to store attributes describing their activities or interests. With the data available to us, a number of issues arise, and I will describe some of the major issues encountered in the data analysis. This is not an exhaustive list, but a sample to illustrate the issues.

First, the data returned from the same API call may vary depending on the format requested. For example, the ATOM format follows a specific schema which forces Twitter to modify data, specifically the tweet text when a tweet is a re-tweet. This forces researchers such as myself to accommodate for these variations by trial and error, and Twitter may make changes with these subtle variations without notification.

Second, profile fields are not validated by Twitter in any way. For example, the “location”

Table 4.1: Table of Twitter profile values returned from the REST API for user "schenkmanus". Keys and values in bold are those used in data analysis for this research. Values of "None" are equivalent to *null* or *empty*.

Key	Value
follow_request_sent	None
profile_use_background_image	True
id	**59859467**
verified	False
profile_sidebar_fill_color	252429
profile_text_color	666666
followers_count	**6**
protected	**False**
location	**earth**
profile_background_color	1A1B1F
id_str	**59859467**
status	***<See table 4.2 for the format of a tweet>***
utc_offset	**-25200**
statuses_count	11
description	CS geek, programming, sysadmin, music, composing, colorado, Hiking, biking, totorcycling, travel
friends_count	**4**
profile_link_color	2FC2EF
profile_image_url	http://a1.twimg.com/profile_images /330294609/evilherbert_normal.jpg
notifications	None
show_all_inline_media	False
geo_enabled	**False**
profile_background_image_url	http://s.twimg.com/a/1288305442/images /themes/theme9/bg.gif
name	**Chris Schenk**
lang	**en**
profile_background_tile	False
favourites_count	0
screen_name	**schenkmanus**
url	None
created_at	**Fri Jul 24 19:06:11 +0000 2009**
contributors_enabled	False
time_zone	**Mountain Time (US & Canada)**
profile_sidebar_border_color	181A1E
following	None
listed_count	**0**

Table 4.2: Table of attributes of the latest Tweet by user "schenkmanus" from the REST API. Keys and values in bold are those used in data analysis for this research. Values of "None" are equivalent to *null* or *empty*.

Key	Value
favorited	False
retweeted_status	*<The original tweet that may or may not have been retweeted will be embedded here and has the same format of the tweet represented in this table with the original author's attributes listed.>*
contributors	None
truncated	False
text	**RT @extralife: It occurs to me that today's comic means zero to anyone not familiar with Gauntlet. http://bit.ly/9aizuo**
created_at	**Thu Sep 30 00:21:16 +0000 2010**
retweeted	False
in_reply_to_status_id	None
coordinates	**None**
id	**25931435478**
source	**web**
in_reply_to_status_id_str	None
place	**None**
in_reply_to_user_id	None
in_reply_to_screen_name	None
retweet_count	None
geo	**None**
in_reply_to_user_id_str	None
id_str	**25931435478**

Table 4.3: Table of friends and followers of user "schenkmanus" as they are returned from Twitter via the REST API. This user is barely active in the network and as such, these lists are very small. Typical users have more friends and followers on average [57].

schenkmanus	
Friends	Followers
1374921	1374921
2989151	2989151
6274662	72343273
	12753892
	23838490
	627466

field is simply a text box in which a user may list his or her location. As you can see in table 4.1, user “schenkmanus” has listed his location to be “earth”.

Third, the notion of a “re-tweet” is a home-grown idea from the community and not a feature initially implemented by Twitter. Over time Twitter included support for re-tweets, but they are unable to track all the ways in which a message is duplicated on the network. As such, I must search for the multiple ways in which a tweet may be re-tweeted which may change over time based on the user population.

Fourth, no timestamp data is included with the friend/follower lists so I have no knowledge of when a user was friended or followed by another. Interestingly, Twitter does return the lists of users in relative order in which they were friended or followed, but that is not granular enough for my goals. As such, I must come up with interesting collection techniques such as collecting a user’s friend/follower lists periodically to see how those lists change. We use this technique in the algorithm described in chapter 6.

4.5 Statistics for Labor Day 2010 Boulder Fires Data

Before I am able to start searching for influential users in any data set, I first want to explore the data and see what properties appear using different metrics. I compute numerous statistics on the attributes and text available from the data collection. I use a dataset collected from the Boulder Four Mile Canyon fire that started around 10:00am on Monday, September 6th, 2010. During this time the search and streaming APIs were utilized to search for users using the keywords and phrases **boulderfire**, **boulder**, **4milefire**, **fourmilefire**, and **fourmilecanyon**. The implication is that this list is additive, increasing as more people talk about the fire. Subsequently once every 30 minutes I collected the friend/follower lists from the REST API of the unique users that appeared from the search and streaming API results at that time, however I analyze the social graph data in chapters 5 and 6.

The statistics generated for the data are as follows. All counts generated are sorted descending.

- Counts of the total number of tweets generated by a user. Note that this number will always cap at 3200 tweets for users limited by Twitter as mentioned in section 4.1.
- Counts of hashtag occurrences
- Counts of username mentions
- Counts of addressed messages from user A to user B
- Counts of the number of times a user was re-tweeted within the data set
- Counts of the number of times user A re-tweeted user B, a more granular statistic than the previous statistic
- Counts of different URLs that are posted by users
- Counts of the number of times an application was used to post a message, such as the Twitter web interface or other third-party applications
- Counts and plots of the latitude/longitude information included in tweets
- Counts of the number of users and edges in the social graph as well as top-ranked in-degree users

Additionally, any geo-location information is pulled from the tweet metadata and frequency and graphs of how many tweets are posted each hour are also generated. Each of these statistics may be computed on a group of users or for individuals. I am also able to limit the statistics to a specific time window given starting and ending dates and times in order to understand behavior during the event as it compares to general behavior.

4.5.1 Qualitative Influencers

Given that the Boulder fire occurred within the city in which I live, my fellow researchers were able to interact directly with people who were speaking actively about the fire. As such, one of

Table 4.4: A qualitative list of influencers during the Boulder fire of Labor Day 2010 as given by fellow researcher Jo White.

Users			
epiccolorado	laurasrecipies	HumaneBoulder	fishnette
suzanbond	CampSteve	ConnectColorado	Org9
metroseen	palen	sophiabliu	Mediamum
Tanukun	eadvocate	kate30_CU	BoulderChannel1

my fellow researchers, Jo White, gave me a qualitative list of users in Twitter who were considered influential during this fire. Table 4.4 is that list of users. Note that user “BoulderChannel1” is considered an influencer but is also known as putting forth mis-information about the fire and not interacting in a helpful manner during the fire. However, many of the posts by this user propagated through the network and therefore is considered influential, even if also delinquent. I use this list to see where these influential users are appearing in the different statistics. I will return to this list in the social graph analysis in chapters 5 and 6.

4.5.2 Statistics of the First 24 Hours and First Week

Here I compare statistics from the first 24 hours of the fire as well as the first week of the fire. The specific time windows begin with Monday, September 6th, 2010 at 10:00:00am MDT and end with Tuesday, September 7th, 2010 at 10:00:00am MDT and Monday, September 13th, 2010 for the 24-hour and one week windows, respectively. The 24-hour data set contains 398 users and the week-long data set contains 13,955 users, demonstrating the additional involvement of new users over time. The number of tweets seen during both time periods are 12,147 and 2,314,700, respectively.

I begin with table 4.5 which is a simple count of tweets for each user in the data set and the associated rate of posting. This information tells us who is most active in posting in this selected group. As you can see from the table, four of the sixteen qualitatively influential users are listed and are highlighted in bold in the 24-hour column. As the event continues during the week, many more people are speaking about the fire and very active users appear at the top of the list, about many of whom I have no information and are most likely not involved at all, but merely mentioning the fire using a re-tweet or one of the keywords used in the initial tweet search. We do see a well-known Boulder resident and technology entrepreneur, “andrewhyde”, appear in the data, and others appear in other statistics as well. One user, “364news”, hits the 3200 tweet limit imposed by Twitter as described in section 4.4. The average and median number of tweets per person during the first day are 30.52 and 9.00, respectively. The average and median number of tweets per person during the first week are 165.87 and 38.00, respectively.

In table 4.6 I show the counts for all hashtags found within the data set. A *hashtag* is an alpha-numeric-underscore string prepended with the pound sign. Examples include **#boulderfire**, **#4milefire**, and **#real_estate**. As can be seen in the 24-hour column, a number of boulder and fire-related hashtags appear, including **#boulderfire**, **#boulder**, **#fourmilefire**, **#fourmilecanyonfire**, **#4milefire**, **#wildfire**, and possibly **#colorado**. However, one can see that the only hashtag that survives among the noisy week-long data is **#boulderfire**. We see the surge of the “friend friday” hashtag in the one week data (denoted by **#ff**) and a mention of the San Bruno, California gas pipeline explosion that occurred in the same week (denoted by **#sanbrunofire**). The total numbers of hashtags seen for the first day and first week are 7,422 and 756,785, respectively. The numbers of unique hashtags seen for the first day and first week are 895 and 66,765, respectively. The average and median appearance of a hashtag seen during the first day are 8.29 and 1.00, respectively. The average and median appearance of a hashtag seen during the first week are 11.34 and 1.00, respectively.

Table 4.7 shows the top username mentions in all the tweet streams of the users confined to the Boulder fire data sets. A username *mention* is simply the appearance of a username in a tweet, such as “@schenkmanus” or “@fishnette”. As can be seen in the table, we see more of the qualitative users in the set during the 24-hour period, and as more users enter the event network, a lot of noise is introduced and we see many other usernames appear that have no relevance. Some qualitative influencers still appear in the one week column, but we see celebrities such as “@justinbieber” and “@barackobama” and local news outlets such as “@cbs4denver” and “@denverpost”. The

Table 4.5: A list of the top 25 users with the most tweet counts and associated post rate of each user in *tweets per hour* in the 24-hour and one week data sets. Users highlighted in bold are part of the qualitative set listed in table 4.4.

	24-Hour			One Week		
Rank	User	Count	Rate	User	Count	Rate
1	**fishnette**	**394**	**16.4**	364news	3200	19.0
2	wind4me	343	14.3	markayitea	3073	18.3
3	puregemstore	335	14.0	carlosmethelly	2919	17.4
4	shravanp	323	13.5	jonchan02	2696	16.0
5	smartmarketing1	302	12.6	batin_is	2687	16.0
6	brendanloy	285	11.9	norskeaviser	2614	15.6
7	tuckertown	267	11.1	hitsbyzuk	2609	15.5
8	world_policy	259	10.8	kshypptl	2594	15.4
9	**tanukun**	**251**	**10.5**	dropshipperssa	2567	15.3
10	stiftioree4	241	10.0	indianews247	2561	15.2
11	_mattters_	240	10.0	djilanihr	2534	15.1
12	karoli	237	9.9	carlosgil83	2523	15.0
13	alltop_science	225	9.4	keruffworldnews	2520	15.0
14	sitfu	216	9.0	kpjobs	2506	14.9
15	newsfeeding	211	8.8	vehixcar	2500	14.9
16	hlane	209	8.7	dhayes1098	2500	14.9
17	nicolew247	207	8.6	mikes_web_page	2486	14.8
18	**epiccolorado**	**183**	**7.6**	cosmonet_news	2482	14.8
19	andrewhyde	175	7.3	selvan_tengy	2478	14.8
20	twnstar2	155	6.5	cronaca24	2470	14.7
21	theroseinbloom	154	6.4	spotifyuri	2469	14.7
22	sarajuliet	148	6.2	bigmikepromo	2469	14.7
23	notperfume	143	6.0	thenewsblotter	2466	14.7
24	highnicole	138	5.8	phayes4342	2460	14.6
25	**mediamum**	**136**	**5.7**	0341marktplaats	2459	14.6

Table 4.6: A list of the top 25 hashtags in use by all users who mentioned at least one keyword related to the Boulder fire. Note that many hashtags appear that are completely unrelated to the event. The counts for the one week data have been normalized to per-day averages in column 5 for comparison with the 24-hour window.

	24-Hour		One Week		
Rank	Hashtag	Count	Hashtag	Count	Daily Avg.
1	#boulderfire	2767	#boulderfire	27982	3997.4
2	#boulder	513	#jobs	20991	2998.7
3	#news	432	#news	19948	2849.7
4	#fire	255	#ff	14368	2052.6
5	#src	151	#tcot	13113	1873.3
6	#loc	135	#p2	8692	1241.7
7	#evac	134	#health	5542	791.7
8	#business	91	#quote	5076	725.1
9	#info	83	#fb	4963	709.0
10	#fourmilefire	61	#socialmedia	4737	676.7
11	#fourmilecanyonfire	59	#sanbrunofire	4337	619.6
12	#politics	56	#fashion	4247	606.7
13	#4milefire	56	#follow	3960	565.7
14	#wildfire	47	#ebc	3729	532.7
15	#hermine	46	#pets	3728	532.6
16	#shelter	37	#tweetmyjobs	3319	474.1
17	#soccer	36	#pakistan	3319	474.1
18	#fb	35	#bookmark	2897	413.9
19	#world	35	#1	2747	392.4
20	#fifa	35	#business	2679	382.7
21	#2010	35	#iranelection	2607	372.4
22	#cup	35	#in	2553	364.7
23	#sports	34	#kaiser	2510	358.6
24	#colorado	34	#travel	2507	358.1
25	#offer	32	#teamfollowback	2472	353.1

percentages of users that mention at least one other user in the first day and first week period are 77.39% (308 users) and 79.08% (11,036 users), respectively. The total numbers of mentions seen during the first day and first week are 7,877 and 1,224,851, respectively. The average and median numbers of mentions per user during the first day are 19.79 and 1.00, respectively. The average and median numbers of mentions per user during the first week are 87.77 and 1.00, respectively.

In table 4.8 I review the number of addressed messages sent between users during the different time windows. Addressed messages are considered to be those that begin with either "*@username*" or prepended with a period as "*.@username*" (both cases without quotes). The highlighted users are part of the qualitative set, and we can see in this case that only two of the qualitative set appear at the top of this list. Again in the one week column, the noise appears when additional users are added into the event network and numerous messages are sent between users who are unknown and are most likely not about the Boulder fire. Interestingly, some users address messages to themselves, as demonstrated by user "back2lifeinc" ranked third in the one week data set. The percentages of messages that are addressed messages during the first day and first week are 18.85% (2,291 messages) and 15.90% (368,047 messages), respectively. The percentages of users who have sent at least one addressed message to others during the first day and first week are 57.04% (227 users) and 60.22% (8,404 users), respectively.

Table 4.9 shows the top re-tweeted users during the two time windows. Here we see a general pattern of news outlets being re-tweeted among people, but we do see some of the qualitative set of users appearing in the list. For the first time we see user "BoulderChannel1" appear using this metric. The percentages of messages that are considered re-tweets during the first day and first week are 32.88% (3,994 messages) and 21.81% (504,836 messages). The numbers of unique users re-tweeted during the first day and first week are 1,456 and 134,204, respectively. Of those users re-tweeted, the percentages of users actually speaking of the Boulder fire during the first day and first week are 24.45% (356 users) and 1.55% (2,085 users), respectively.

Table 4.10 shows the top re-tweeted source-target user pairs during the two time windows. This measure is a more fine-grained measure than the previous total count as show in table 4.9.

Table 4.7: A list of the top 25 username mentions in the 24-hour and one week data sets of the Boulder fire. Highlighted users in bold are part of the qualitatively influential set of users.

	24-Hour		One Week	
Rank	Username	Count	Username	Count
1	**fishnette**	**540**	addthis	4131
2	andrewhyde	223	breakingnews	3150
3	cbs4denver	197	jinxbeatz	2554
4	**epiccolorado**	**188**	mikes_web_page	2416
5	denverchannel	87	youtube	2310
6	**mediamum**	**79**	**epiccolorado**	**1861**
7	kwgndenver	69	cnnbrk	1812
8	wind4me	50	**fishnette**	**1732**
9	thefiretracker2	46	**laurasrecipes**	**1560**
10	bouldercounty	46	addtoany	1486
11	**humaneboulder**	**42**	justinbieber	1484
12	twitter	41	nytimes	1479
13	dailycamera	39	mashable	1383
14	**sandrafish**	**39**	dapitarchuletoy	1292
15	hlane	38	cbs4denver	1199
16	jamesazure	35	c1	1171
17	**laurasrecipes**	**34**	barackobama	1151
18	sarajuliet	33	tweetsmarter	1075
19	tuckertown	30	denverchannel	1071
20	**suzanbond**	**30**	mikeposner	1065
21	ev	28	denverpost	1064
22	coloradodaily	27	flipbooks	1064
23	beregond	27	cnn	1013
24	kgnu	26	nasa	941
25	shellimeyers	26	thefiretracker2	925

Table 4.8: A list of the top 25 counts of addressed messages originating from users speaking of the Boulder fire within the first day and first week. The message sent to the target user in this context does not imply the users were speaking of the fire directly. Users highlighted in bold are part of the qualitative set.

24-Hour			One Week		
Source	Target	Count	Source	Target	Count
bellamom	beregond	27	darebat	c1	809
wind4me	**fishnette**	25	kizziecherry	gogumba	697
tuckertown	lrockwellatty	20	back2lifeinc	back2lifeinc	579
brendanloy	inthebleachers	19	darebat	iquit	573
brendanloy	jeremysbn	19	ei_econewsfeed	ecointernet	521
brendanloy	kilroyfsu	16	socbookmarks	iquaks	415
wind4me	cbs4denver	13	seobookmarks	iquaks	415
tuckertown	drmarm	11	davidakhoa	c1	361
theroseinbloom	monkeyxplosion	11	yomarques	lynndaisbellamy	293
karoli	harrylyme	11	davidakhoa	iquit	274
hlane	scottc10	10	bookmarkingnet	youblr	260
mike_flys	cholubaz	9	radu_palanga	irinabbz	258
highnicole	sinisterlukey	9	garyblackmon	gumbyiam	255
andrewhyde	**fishnette**	9	lissaloucraigy2	sematalba	247
priscillast	**fishnette**	9	allstarmn	sarahallstar94	240
rdwnggrl	depmodechick	9	radu_palanga	100tulip	240
fishnette	priscillast	9	jesusfan_420	ofinfinitejest	231
mike_flys	dhindmanjr	9	allstarmn	itsheather7	223
sarajuliet	**fishnette**	9	radu_palanga	dearmikeyway	213
brendanloy	andy_staples	8	karu1402	bensonthehusky	205
fishnette	andrewhyde	8	beeblez	reddirtisland	202
fishnette	sarajuliet	8	mysticjeremy	forbescarolinev	199
scobleizer	kichigai	8	bluecornpie	ally1r	189
kkartphoto	**laurasrecipes**	7	tellydubby	anniewestdotcom	188
scobleizer	srikanth_br	7	theliterator	coloursfading	181

Table 4.9: A list of the top 25 re-tweeted users in the first day and first week of the Boulder fire. The one week counts are normalized to per-day averages for comparison with the 24-hour counts. Users highlighted in bold are part of the qualitative set.

	24-Hour		One Week		
Rank	User	Count	User	Count	Daily avg.
1	**fishnette**	**301**	addthis	4097	585.3
2	andrewhyde	143	breakingnews	3058	436.9
3	cbs4denver	140	youtube	2157	308.1
4	**epiccolorado**	**127**	addtoany	1486	212.3
5	denverchannel	69	cnnbrk	1354	193.4
6	kwgndenver	56	**epiccolorado**	**1185**	**169.3**
7	**mediamum**	**46**	nytimes	1132	161.7
8	bouldercounty	36	tweetsmarter	997	142.4
9	thefiretracker2	35	flipbooks	978	139.7
10	dailycamera	27	cbs4denver	969	138.4
11	wind4me	27	greychampion	910	130.0
12	breakingnews	25	denverchannel	899	128.4
13	kgnu	24	denverpost	882	126.0
14	coloradodaily	24	mashable	870	124.3
15	**boulderchannel1**	**23**	nasa	842	120.3
16	hkoren	21	**fishnette**	**819**	**117.0**
17	kkartphoto	21	thefiretracker2	748	106.9
18	schwartznow	20	**laurasrecipes**	**739**	**105.6**
19	cnnbrk	20	wxchannel	729	104.1
20	cuindependent	19	huffingtonpost	708	101.1
21	tuckertown	19	thenewsblotter	638	91.1
22	**humaneboulder**	**18**	detikcom	596	85.1
23	**suzanbond**	**18**	kike230	575	82.1
24	douginboulder	17	pixelproject	573	81.9
25	jamesazure	17	mparent77772	570	81.4

We immediately see that the noise of the week-long data set drowns out any interaction between users active in the Boulder fire. We do see many people re-tweeting user “fishnette” in the 24-hour period.

Table 4.11 shows the top-25 encountered urls in the two time windows. Users very frequently share URLs that are shortened using services from domains that are themselves short (such as “bit.ly”) and use a hash function to encode the remainder of the URL and then forward the request to the target URL. Due to this feature, I am unable to immediately see what are the actual top URL destinations. In order to determine this information, I must use URL expander services or write a script that follows the links until an HTTP 200 response is encountered, signifying the final destination of the links. Without this process, these URLs are meaningless by themselves. We also see one of the abhorrent behaviors of Twitter in this table with truncated URLs, such as “http://b” and others. This is due to requesting ATOM format in the API call, and in the case of re-tweets, Twitter will prepend the text with the “RT @username” construct and truncate the tail of the message which typically contains the URL. Requesting JSON format from the API call does not introduce this behavior. The total numbers of unique URLs seen during the first day and first week are 4,105 and 1,200,927, respectively.

Table 4.12 lists the top 25 applications utilized by users to access Twitter. There exist numerous applications that allow users to access Twitter. In the 24-hour period we see 85 unique applications and in the one week period we see 1,026 unique applications. The percentages listed in the table are calculated from totals of 12,147 and 2,314,700 tweets for the 24-hour and one week time windows, respectively. The application name actually appears as HTML and the name is parsed out from the HTML itself and is displayed in the table. By itself, this information isn’t very useful, but observing the dynamics of what applications are used by people during events compared to their normal behavior may yield interesting behavioral patterns.

Figures 4.1, 4.2, and 4.3 show the locations of tweets by 30 users among the 398 total users in the 24-hour data set, accounting for 7.53% of the users. Among these 30 users, 172 tweets contained latitude/longitude data, accounting for 1.42% of the 12,147 tweets seen during the first day. All of

Table 4.10: A list of the top 25 re-tweeted users in the first day and first week of the Boulder fire. The user in the *source* column is the originator of the tweet. Users highlighted in bold are part of the qualitative set.

24-Hour			One Week		
Re-tweeter	Source	Count	Re-tweeter	Source	Count
wind4me	**fishnette**	59	canadianinjured	greychampion	910
tanukun	**fishnette**	37	nygnowlivetv	addthis	663
cuindependent	**fishnette**	27	kike230	kike230	575
mediamum	**epiccolorado**	25	hopemarie_25	peacekaren_25	506
sarajuliet	**fishnette**	24	mamalou52	virtual_abbey	356
epiccolorado	**fishnette**	23	ishfaq01	addtoany	347
michaelevs	cbs4denver	20	retweeter33	flipbooks	346
theroseinbloom	**fishnette**	19	jap4haiti	haiti_11	340
twnstar2	cnnbrk	16	pacifictimesmag	youtube	335
wind4me	cbs4denver	16	viravita	viravita	315
tuckertown	cbs4denver	16	mudpie25	dylano	311
palen	**epiccolorado**	15	batin_is	detikcom	307
michaelevs	kwgndenver	14	lodispirit	pixelproject	282
wind4me	denverchannel	14	newmediadevotee	ericpratum	260
jennar	**fishnette**	12	actionscript3	shary20	254
torqueflite	tuckertown	12	jazzyaditya	detikcom	253
notperfume	nannersmom	12	quintinreports	thenewsblotter	253
velofemme	**fishnette**	11	8service	addthis	247
tanukun	thefiretracker2	9	thenewshome	addtoany	239
twnstar2	thefiretracker2	9	jonchan02	globalgrind	229
hlane	**fishnette**	9	angelwings0511	adamsconsulting	208
wind4me	andrewhyde	9	viravita	andinifaramitha	203
cuindependent	andrewhyde	9	riqhimon	riqhimon	200
notperfume	petmamma	9	batin_is	kompasdotcom	200
wind4me	joshlarson	8	cosmoweb	cosmoweb	191

Table 4.11: A list of the top 15 URLs appearing in the two data sets. Due to idiosyncrasies of the Twitter API when requesting ATOM format, some URLs are truncated.

	24-Hour	
Rank	Url	Count
1	http://bit.ly/ccX7kH	30
2	http://twitpic.com/2m1ghy	29
3	http://www.radioreference.com/apps/audio/?action=wp&feedId=591	23
4	http://	20
5	http://bit.ly/9yxpOQ	19
6	http://bit.ly/cYIVtM	16
7	http://bit.ly/9c2GjV	14
8	http://bit.ly/bh93Fu	13
9	http://twitpic.com/2lw39r	13
10	http://boulderoem.com/component/content/article/5	13
11	http://bit.ly/c2wnXh	12
12	http://post.ly/vyWH	12
13	http://bit.ly/9IuULq	12
14	http://bit.ly/9AX3JL	12
15	http://bit.ly/b3Mvqx	11

	One Week	
Rank	Url	Count
1	http://bit.ly/5yluCl	3028
2	http://bit.ly/9n3Ifc	2249
3	http://thenewslist.com	1766
4	http://bit.ly/b6xyyB	1595
5	http://	1489
6	http://malufor.ch	1111
7	http://bit.ly/4vPZm1	1062
8	http://bit.ly/9t0MRU	1055
9	http://CarlosGil.us	951
10	http://ow.ly/2k6oa	894
11	http://bit.ly/cbhpXo	848
12	http://ow.ly/2k4Sf	846
13	http://adf.ly/5FcS	820
14	http://www.bit.ly/lookingforwork	780
15	http://b	748

Table 4.12: A list of the top 25 source applications used by the users in the Boulder fire network. Tweet totals for both time windows are 12,147 and 2,314,700 for 24-hour and one week, respectively. The distribution percentages are given.

24-Hour			One Week		
Source	Count	Percent	Source	Count	Percent
web	3471	28.57	twitterfeed	801610	34.63
twitterfeed	2541	20.92	web	453113	19.58
TweetDeck	2457	20.23	TweetDeck	197643	8.54
Echofon	565	4.65	HootSuite	94240	4.07
Twitter for iPhone	548	4.51	ÜberTwitter	63977	2.76
HootSuite	418	3.44	Twitter for iPhone	63774	2.76
Tweetie for Mac	322	2.65	dlvr.it	44222	1.91
Seesmic Desktop	247	2.03	Echofon	38997	1.68
ÜberTwitter	165	1.36	Ping.fm	31788	1.37
Wowd api	165	1.36	Google	30675	1.33
Twitterrific	128	1.05	Twitter for BlackBerry	26068	1.13
Twitter for iPad	127	1.05	mobile web	23750	1.03
twidroid	100	0.82	txt	21477	0.93
hottopics7	94	0.77	twidroid	15056	0.65
Twitter for BlackBerry	88	0.72	Seesmic Desktop	13232	0.57
Twitpic	78	0.64	SocialOomph	10879	0.47
Twitter for Android	58	0.48	Tweetie for Mac	10648	0.46
mobile web	54	0.44	Twittelator	10477	0.45
Seesmic for Android	42	0.35	Tweet Button	8866	0.38
TweetMeme	37	0.30	TweetCaster	8705	0.38
txt	36	0.30	Twitter for Android	8261	0.36
Tweet Button	31	0.26	TwitBird	8238	0.36
TweetCaster	28	0.23	Visibli	6710	0.29
YoruFukurou	28	0.23	Facebook	6707	0.29
foursquare	26	0.21	Seesmic for Android	6556	0.28

the 172 points were plotted in a Google map using the Google Maps API.[6] The first map in figure 4.1 shows users from California, Nevada, Wisconsin and Indiana did speak about the Boulder fire in some way. Zooming in closer to Denver in figure 4.2 and 4.3 shows the higher density of Tweets appearing in Boulder.

The number of users in the one week data that gave location information is 858, accounting for 6.14% of the users. Among those users, the number of tweets that contain latitude/longitude location information is 17,903, accounting for 0.77% of the 2,314,700 tweets generated by users speaking of the fire in any manner during the week. This data is not plotted due to the volume of points. Points need to be grouped using clustering methods such as k-means (as described in section 3.5) in order to reduce the overall amount to be feasibly plotted on a map.

Table 4.13 shows information about the social graph associated with users speaking of the Boulder fire. The table includes the number of users in the network speaking of the fire and the number of edges connecting those users (considered "active" in the table), as well as all users and edges encountered in the friends/follower lists. Data are calculated for five different snapshots of the social network set one day apart from each other beginning with Tuesday, September 7th at 12:40pm and ending on Saturday, September 11th at 3:10pm. As can be seen, the number of users increases during the week. Although the data suggests people were leaving the network due to the decrease in users in columns three and four, this is in fact due to data collection errors encountered. The small-world property [122] of Twitter is immediately apparent with the unique number of users (in the millions) who are one connection away from those speaking about the Boulder fire.

Table 4.14 shows the ranking of users by the number of followers of users speaking during the first day of the fire collected on September 7th, 2010 at 12:41pm Mountain time. Both the global rankings and rankings only among the users who are speaking about the Boulder fire are listed. In the case of rankings only among the active users in the fire, all user IDs that are encountered that are not part of the active user set are removed and edges to those users are ignored. In the global list, no qualitative influencers are found. However, in the list where only users actively speaking

[6] http://code.google.com/apis/maps/documentation/javascript/reference.html

Figure 4.1: An image of latitude/longitude points extracted from tweet post metadata from posts by users speaking of the Boulder fire in the first 24 hours overlain on a map of the United States.

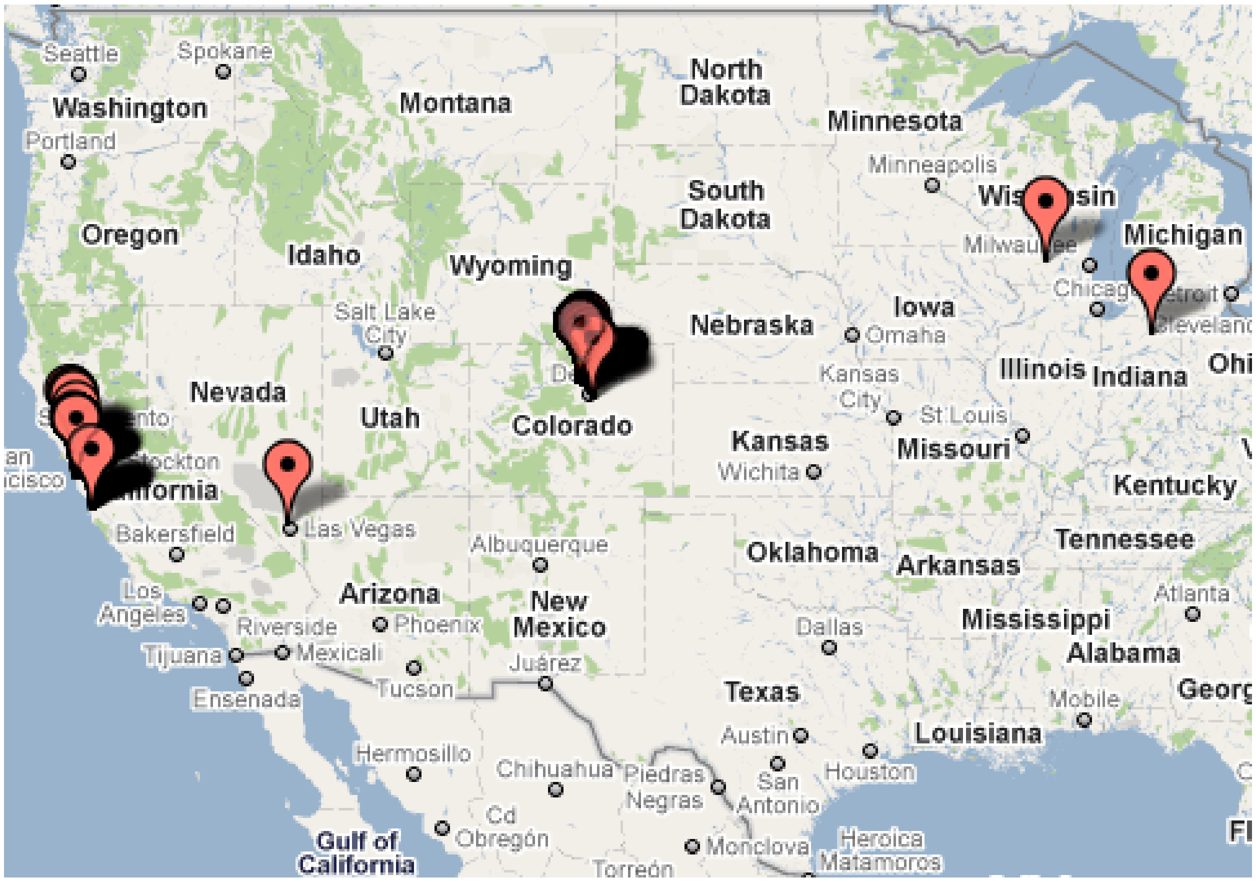


Table 4.13: A list of statistics of the social graph in one-day increments during the Boulder fire. Active edges connect the users who are actively speaking about the fire in some way. All edges and users are counts of all unique users encountered in the data sets.

	Timestamps (America/Denver)				
	2010-09-07 12:40:01	2010-09-08 12:40:01	2010-09-09 12:40:01	2010-09-10 12:40:01	2010-09-11 15:10:01
Users (active)	448	1,631	1,623	1,622	4,039
Users (all)	821,609	2,292,929	2,295,885	2,300,838	4,075,573
Edges (active)	3,142	25,193	25,484	25,664	87,539
Edges (all)	1,510,036	5,361,650	5,370,451	5,372,597	30,458,948

Figure 4.2: An image of latitude/longitude points extracted from tweet post metadata from posts by users speaking of the Boulder fire in the first 24 hours, with zoom over the Denver, Colorado metropolitan area.

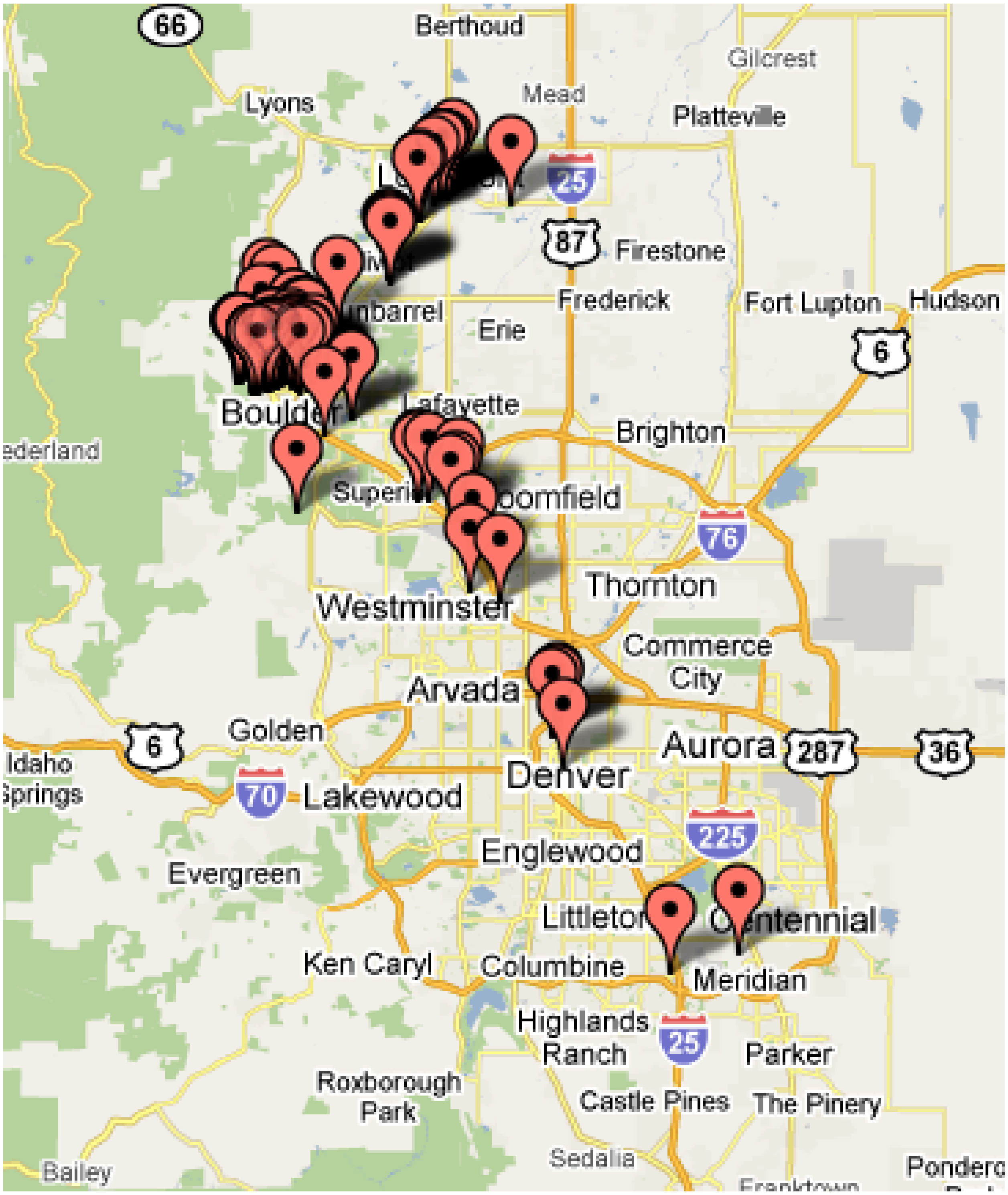
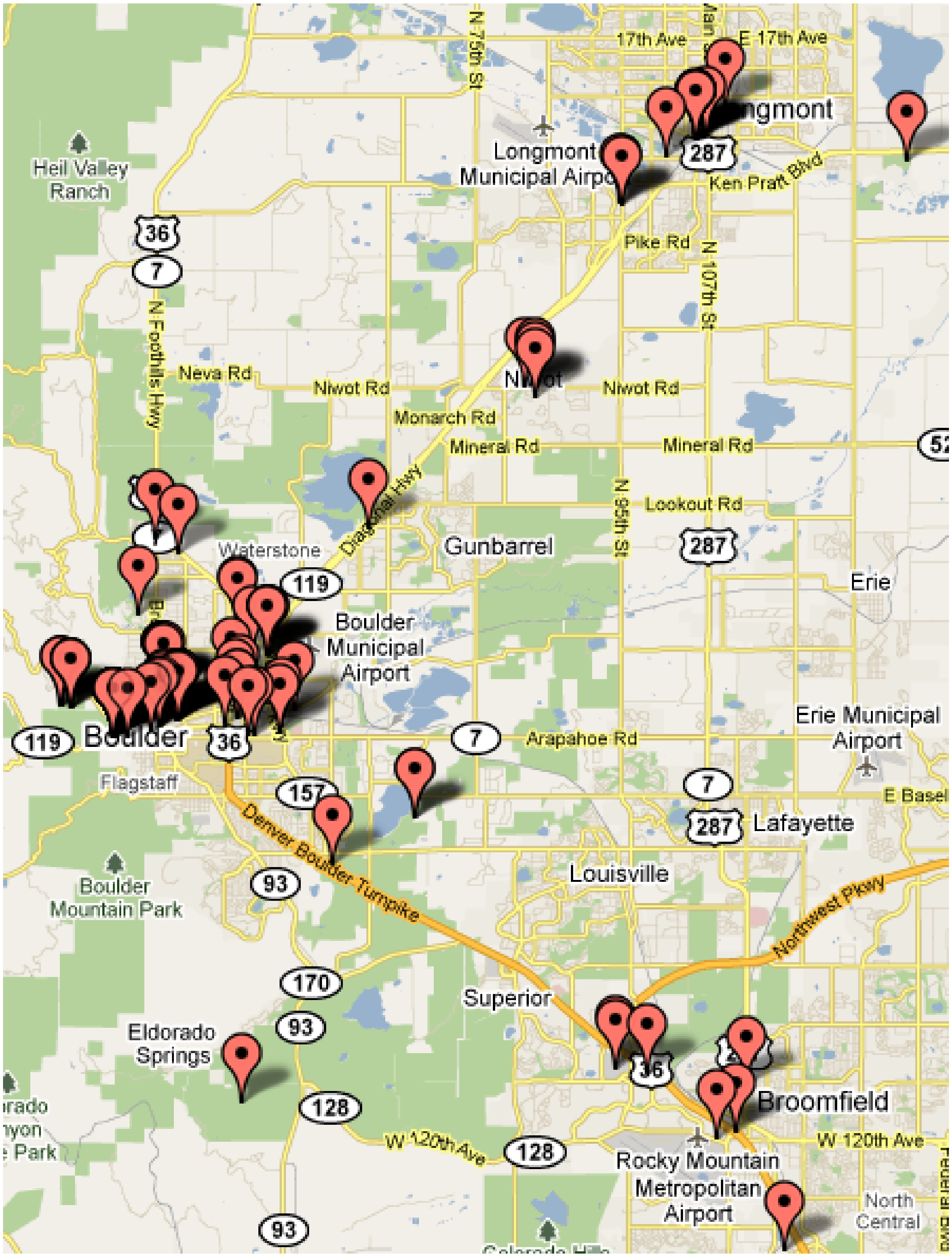


Figure 4.3: An image of latitude/longitude points extracted from tweet post metadata from posts by users speaking of the Boulder fire in the first 24 hours, with zoom over the Boulder, Longmont, and Broomfield cities in Colorado.



about the fire are considered, qualitative influencers begin to appear in the data set.

4.5.3 Statistics of User “fishnette”

Now I will investigate statistics for individual users. Since user “fishnette” appears a lot in the statistics from the previous section, I will look at the same statistics from that user’s point of view. This user’s statistics will be viewed from two different time periods: the time period before the start of the fire on September 6th, 2010 at 10:00am and data after the start of the fire. This user joined Twitter Monday, April 28, 2008 at 7:39pm. For comparison with the data from the previous section, the total number of tweets posted by this user is 394 for the first day as seen in table 4.5 resulting in the top ranking. This user posted 879 for the first week resulting in a ranking of 563 which was lost among the high-volume users in the one week time period.

Figure 4.4 shows the monthly post rates for user “fishnette”. Even though this user joined the network in April of 2008, only the most recent 3,200 tweets are visible due to the limits imposed by Twitter [118]. However, even with this limitation the general posting behavior is visible. A significant increase can be seen at the start of September 2010 which correlates with the Boulder fire. Figure 4.5 shows the posts aggregated by hour of the day. A large amount of posts can be seen between 11:00am and 12:00pm Mountain time, indicating a possible lunch hour spike in activity. Sleep hours can also be seen between 11:00pm and 7:00am.

Table 4.15 shows the hashtags in use by “fishnette” during the pre-fire and post-fire time periods. A variety of hashtags can be seen in the pre-fire time period, but the frequency is low relative to the week during the fire. The number of hashtags used before the Boulder fire is 273 yielding a ratio of 8.53% of hashtags to tweets out of 3,200 tweets. During the first day of the fire the ratio increases to 86.80% of 342 hashtags within 394 tweets. This ratio decreases during the one week data set to 63.82% of 561 hashtags within 879 tweets. Different events can be seen in hashtags during the one week period as well, such as the **#sanbruno** hashtag referencing the San Bruno, California gas fire mentioned previously. The appearance of support hashtags such as **#evac**, **#loc**, and **#needs** appear as the event evolves and relief needs become apparent and

Table 4.14: A list of the top 25 users ranked by their number of followers in the data sets collected for September 7th, 2010 at 12:41pm. The first rankings are of global follower counts and the second rankings are of counts only among the active users speaking about the Boulder fire. Users highlighted in bold are in the qualitatively influential set.

	All Users			Active Users		
Rank	User ID	User Name	Count	User ID	User Name	Count
1	13348	Scobleizer	138067	841791	andrewhyde	123
2	24269775	marcelomedici	77627	14504258	dailycamera	98
3	19816859	PublishersWkly	76555	16119689	downtownboulder	97
4	15731368	HowardKurtz	36212	14522630	coloradodaily	88
5	18460854	shellykramer	33082	16313592	cbs4denver	81
6	16971072	taylorphinney	22639	**14574055**	**fishnette**	**72**
7	11425612	EthanJaynes	22006	817209	davidcohen	52
8	56597144	mr_q	18212	13348	Scobleizer	52
9	4374531	gwenbell	15239	**5124341**	**Mediamum**	**51**
10	16877611	FoxSports	15137	29527113	OnlyInBoulder	50
11	841791	andrewhyde	11790	12627132	melsidwell	40
12	126444618	THEKASESHOW	11709	14911286	Tekee	40
13	80619999	Freedomman11	10301	4374531	gwenbell	40
14	39553219	KaliMarcum	9485	**64876717**	**HumaneBoulder**	**40**
15	16313592	cbs4denver	8860	**85647240**	**epiccolorado**	**39**
16	53153534	craigslistjobs	8767	14945780	ejoep	35
17	980611	Karoli	8396	6752232	Greeblemonkey	35
18	23996960	Jastewart15	8198	16733088	gratzo	34
19	35346943	pops131	8155	14208757	bouldertweep	33
20	22795877	bigdandbubba	7640	1142941	jenmyronuk	33
21	41326847	salivates	6962	16971072	taylorphinney	33
22	21726778	_BellaCullen18_	6872	14317338	KezzaMcDezza	32
23	14705116	AmeriCares	6578	24663013	hkoren	31
24	18124625	casinclair	6300	19661756	hlane	31
25	45797123	eraseyourdebt	6275	15791661	fionaschlachter	30

Figure 4.4: A graph of the posts created each month by user "fishnette" going back to the most recent 3,200 tweets. A spike in posting activity can be seen in the month of September, correlating with the start of the Boulder fire.

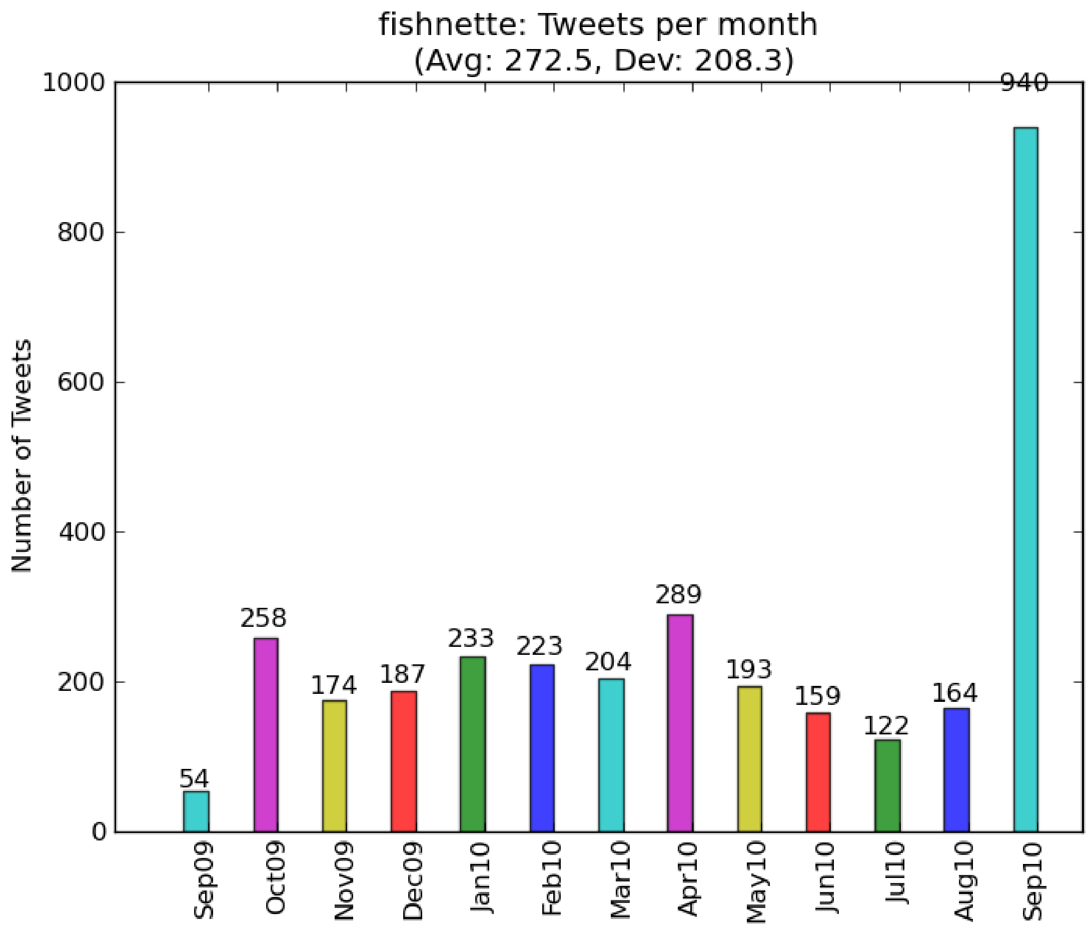
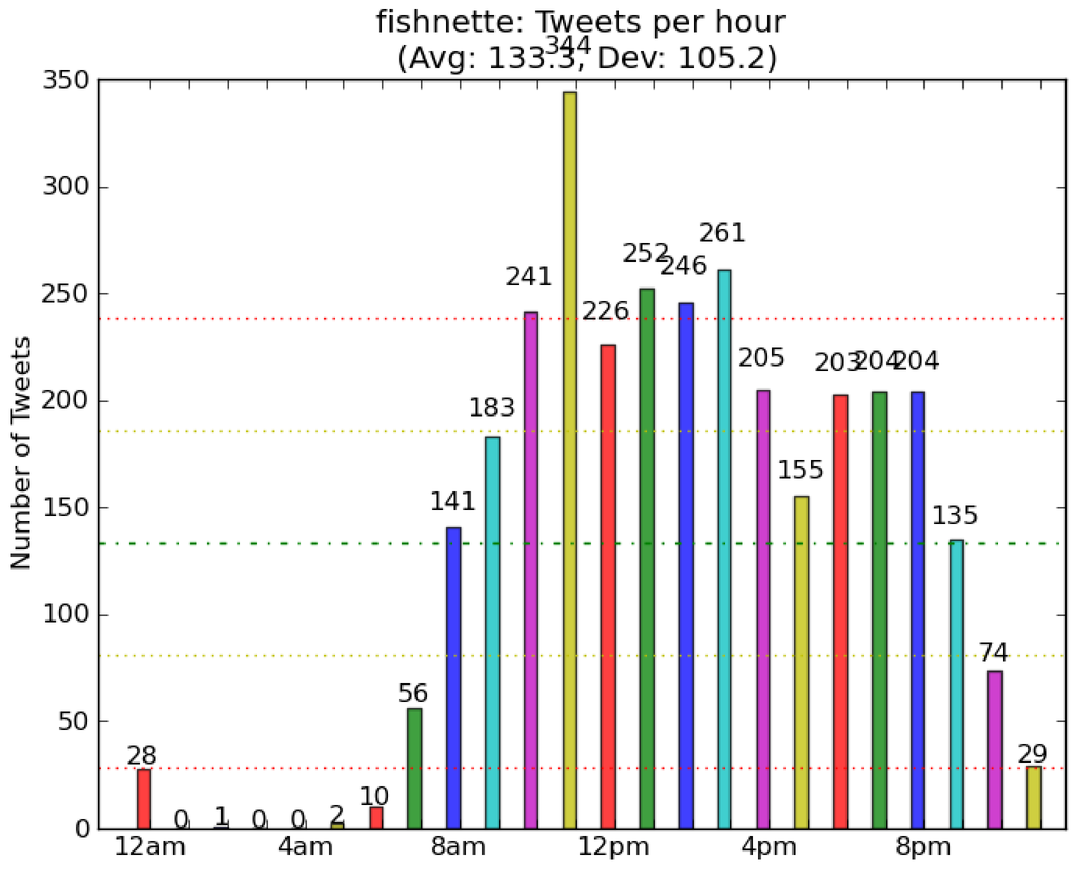


Figure 4.5: A graph of the posts created aggregated by hour of the day by user “fishnette” in the Mountain time zone. The green line is the average and the red lines are ±1 standard deviation. A lunch hour and sleep hours are visible.



vocalized in the community.

Table 4.16 shows the top 25 usernames mentioned in posts by user “fishnette”. As seen in the overall event data described in the previous section, user “andrewhyde” appears again who is the local entrepreneur. Other names that are starting to appear multiple times are “wind4me”, “sarajuliet”, “priscillast”, and “douginboulder”. User “fishnette” also mentions herself a number of times during the week.

Table 4.17 shows the counts of messages addressed to and from user “fishnette” in both the 24-hour and one week time periods. Table 4.18 shows the top users re-tweeting user “fishnette” and the top users retweeted *by* “fishnette”. Finally, the source applications used by user “fishnette” pre-fire and during the one week time period are listed in table 4.19.

Table 4.15: A list of the top 15 hashtags in use by user "fishnette" before the Boulder fire, during the first day of the fire, and during the first week of the fire.

Pre-fire		24-Hour		One Week	
Hashtag	Count	Hashtag	Count	Hashtag	Count
#cwa2010	46	#boulderfire	206	#boulderfire	387
#cogopassembly	45	#boulder	51	#boulder	53
#ragbrai	15	#fire	50	#fire	50
#cosen	12	#evac	22	#evac	25
#boulder	9	#bocofire	5	#sanbruno	6
#nicar	7	#cu	2	#bocofire	5
#tcot	7	#fourmilecanyonfire	2	#flatironsfire	4
#spj9	6	#loc	1	#src	3
#igniteboulder	6	#info	1	#loc	3
#gypsyjazz	5	#src	1	#info	2
#cologopassembly	5	#evacuated	1	#scanner	2
#2601	5	-	-	#cu	2
#cogop	4	-	-	#fourmilecanyonfire	2
#ipad	4	-	-	#needs	1
#jaws25	3	-	-	#fingerscrossed	1

Table 4.16: A list of the top 25 usernames mentioned by user "fishnette" in the 24-hour and one week time windows during the Boulder fire. Users highlighted in bold are part of the qualitatively influential set of users.

	24-Hour		One Week	
Rank	Username	Count	Username	Count
1	andrewhyde	18	**epiccolorado**	**31**
2	sarajuliet	12	**laurasrecipes**	**29**
3	wind4me	10	andrewhyde	21
4	priscillast	10	sarajuliet	17
5	denverchannel	8	denverchannel	15
6	stephanieldavis	6	**sophiabliu**	**13**
7	hmcwilliams	6	priscillast	13
8	**mediamum**	**6**	wind4me	12
9	pachecod	5	danielpetty	12
10	greeblemonkey	5	**fishnette**	**11**
11	catawu	5	**mediamum**	**11**
12	**fishnette**	**4**	hmcwilliams	10
13	**epiccolorado**	**4**	stephanieldavis	9
14	ofuttufo	3	lauren_hannah	7
15	davidherrold	3	pugofwar	7
16	douginboulder	3	artworkss	7
17	lioncaller	2	bouldercounty	7
18	tashigonpo108	2	douginboulder	7
19	suesalinger	2	zarchasmpgmr	6
20	metroroadies	2	chelseamoll	6
21	artworkss	2	davidherrold	6
22	hkoren	2	melindadc	5
23	paullewin	2	matical	5
24	trjh	2	ickaickaicka	5
25	milliman	2	catawu	5

Table 4.17: A list of the top 15 users who have received addressed messages *from* user "fishnette" and who have sent addressed messages to user "fishnette" in the 24-hour and one week time periods. Users highlighted in bold are part of the qualitatively influential set.

	Messages *from* user "fishnette"			
	24-Hour		One Week	
Rank	Username	Count	Username	Count
1	priscillast	9	**laurasrecipes**	**20**
2	andrewhyde	8	sarajuliet	12
3	sarajuliet	8	danielpetty	11
4	wind4me	7	priscillast	10
5	stephanieldavis	5	**sophiabliu**	**10**
6	**mediamum**	**5**	andrewhyde	10
7	catawu	5	stephanieldavis	8
8	pachecod	4	wind4me	8
9	greeblemonkey	3	lauren_hannah	7
10	davidherrold	2	**mediamum**	**7**
11	tashigonpo108	2	chelseamoll	6
12	artworkss	2	zarchasmpgmr	6
13	lioncaller	2	artworkss	6
14	vanillagrrl	2	**epiccolorado**	**5**
15	metroroadies	2	catawu	5

	Messages addressed *to* user "fishnette"			
	24-Hour		One Week	
Rank	Username	Count	Username	Count
1	wind4me	25	wind4me	30
2	andrewhyde	9	**laurasrecipes**	**29**
3	priscillast	9	sarajuliet	15
4	sarajuliet	9	andrewhyde	13
5	ofuttufo	5	priscillast	11
6	**mediamum**	**5**	**sophiabliu**	**9**
7	pachecod	5	**mediamum**	**9**
8	greeblemonkey	3	danielpetty	7
9	suesalinger	3	zarchasmpgmr	7
10	lioncaller	3	lauren_hannah	6
11	nealmcb	2	nuancechaser	6
12	tekee	1	**suzanbond**	**6**
13	dhendersonco	1	ofuttufo	5
14	joseph_flasher	1	pachecod	5
15	velofemme	1	ickaickaicka	5

Table 4.18: A list of the top 15 users who have been re-tweeted *by* “fishnette” and who have re-tweeted “fishnette” in the 24-hour and one week time periods. Users highlighted in bold are part of the qualitatively influential set.

	Users re-tweeting “fishnette”			
	24-Hour		One Week	
Rank	Username	Count	Username	Count
1	wind4me	59	wind4me	63
2	**tanukun**	**37**	**tanukun**	**46**
3	cuindependent	27	**epiccolorado**	**32**
4	sarajuliet	24	sarajuliet	31
5	**epiccolorado**	**23**	theroseinbloom	29
6	theroseinbloom	19	cuindependent	27
7	jennar	12	**laurasrecipes**	**25**
8	velofemme	11	luthers	25
9	hlane	9	tweetingdonal	18
10	suesalinger	8	jennar	18
11	pachecod	7	**suzanbond**	**15**
12	agahran	5	hlane	14
13	**mediamum**	**4**	suesalinger	13
14	kwgndenver	4	fourmilefire	12
15	greeblemonkey	3	thefiretracker2	12

	Users re-tweeted *by* “fishnette”			
	24-Hour		One Week	
Rank	Username	Count	Username	Count
1	andrewhyde	8	denverchannel	14
1	denverchannel	8	**epiccolorado**	**12**
1	hmcwilliams	4	andrewhyde	9
1	douginboulder	3	hmcwilliams	8
1	wind4me	3	douginboulder	5
1	sarajuliet	3	wind4me	4
1	**epiccolorado**	**2**	coloradodaily	4
1	cbs4denver	2	**laurasrecipes**	**4**
1	hkoren	2	boulderrescue	3
1	greeblemonkey	2	dailycamera	3
1	stephanieldavis	1	cbs4denver	3
1	yulsman	1	bouldercounty	3
1	shellimeyers	1	hkoren	3
1	pachecod	1	sarajuliet	3
1	davidherrold	1	yulsman	3

Table 4.19: A list of the top 10 source applications used by user “fishnette”, pre-fire, during the first day, and during the first week of the Boulder fire.

Pre-fire		24-hour		One Week	
TweetDeck	1925	TweetDeck	389	TweetDeck	857
Twitter for iPhone	312	Twitter for iPhone	5	Twitter for iPhone	17
API	16	-	-	Twitter for iPad	3
web	14	-	-	Tweety Got Back	1
txt	10	-	-	txt	1
TwitPic	9	-	-	-	-
TweetMeme	2	-	-	-	-
ROFLquiz	1	-	-	-	-
Twitter for iPad	1	-	-	-	-
TimesPeople	1	-	-	-	-

Chapter 5

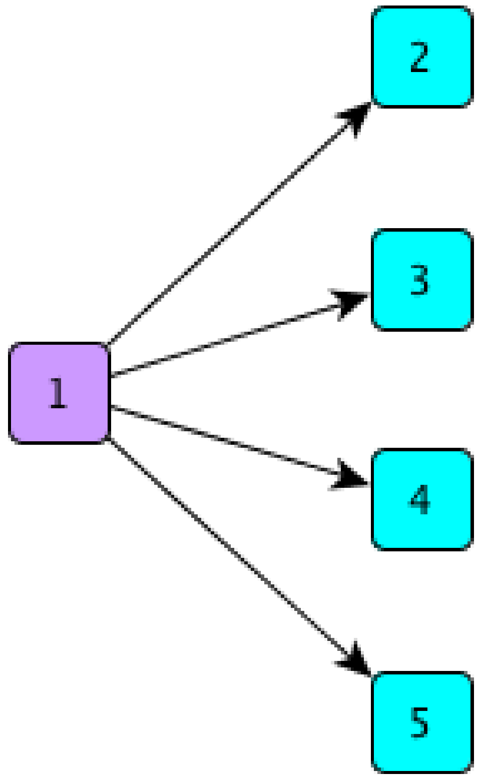
Analysis of HITS Algorithm

In this chapter I investigate the efficacy of the rankings produced by the Hyperlink-Induced Topic Search (HITS) algorithm (described in section 3.14) on various types of social graphs from the September 2010 Boulder fire used as inputs as a means to determine influential users in the network being considered. Prior to my analysis of the algorithm, I will include an explanation of the meaning of the ranking values as well as examples to assist the reader in visualizing how the values correspond to the structure of the graph. The goal is to find a variation of the HITS algorithm input graph to find the most qualitatively influential users listed in table 4.4.

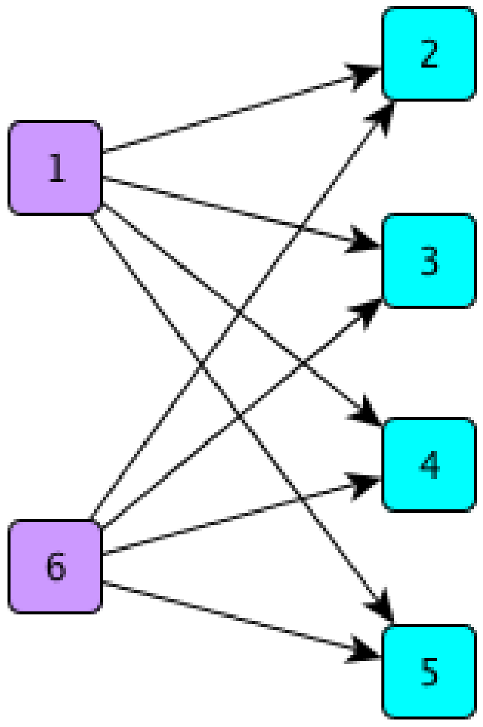
5.1 Hub and Authority Rank Values

The process of generating hub and authority rank values has been described in section 3.14.2. However, before describing our analysis of the algorithm, I want to introduce the reader to how the values relate to the structure of the graph. As described by Kleinberg [65], "Hubs and authorities exhibit what could be called a mutually reinforcing relationship: a good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs." The circular dependency of these values is apparent in the description, but as long as the adjacency matrix of the graph has a dominant eigenvalue, the algorithm is guaranteed to converge on values for hubs and authorities given enough iterations.

To illustrate the relationship between hubs and authorities, I present two simple graphs in figure 5.1 of 5 vertices and 4 edges (5.1(a)) and 6 vertices and 8 edges (5.1(b)), respectively. The



(a) A simple network graph of 5 vertices and 4 edges. Vertex 1 is the only hub in the network and vertices 2-5 all equally split the authority score in the network.



(b) A simple network graph of 6 vertices and 8 edges. Vertices 1 and 6 equally split the hub score in the network and vertices 2-5 again equally split the authority score in the network.

Figure 5.1: Sample graphs to illustrate the relationships between hubs and authorities. Hubs are colored in lavender and authorities are colored in cyan.

Table 5.1: Hub and Authority values corresponding to the graphs in figure 5.1(a).

Rank	Hub	Value	Auth.	Value
1	2	0.5	1	1.0
2	3	0.5	2	0.0
3	4	0.5	3	0.0
4	5	0.5	4	0.0
5	1	0.0	5	0.0

invariant maintained during the calculation of the hub and authority values is the sum of the squares of the values must equal 1. Table 5.1 shows the scores for the graph in figure 5.1(a). Intuitively we see that vertex 1 is the only hub in the entire graph, and vertices 2-5 share the authority scores equally. Table 5.2 shows the scores for the graph in figure 5.1(b). This time we see vertices 1 and 6 equally share the hub values for the graph, and the authority values for vertices 2-5 are once again split equally. We can also see that the invariant is held true as the sum of the hub values in the graph in figure 5.1(b) sum to one ($0.7071 \sim \sqrt{2}/2$, $2 \times \sqrt{2}/2^{2} = 1$).

Figure 5.2 demonstrates a "clique" graph, which is a fully inter-connected directed graph. The values in table 5.3 show the resulting identical scores for every single vertex in the graph for both hubs and authorities, as intuitively, no vertex is any more a hub or authority than the previous.

5.2 Social Graph Data

As described in section 4.5.2, during the Boulder fire the social graph of friends and followers was collected for each user every 30 minutes starting September 6th, 2010 at 8:40pm Mountain time. For the analysis in this chapter, one-day increments of the social graph will be analyzed. Tables 4.13 and 4.14 in section 4.5.2 show statistics of the social graph data collected as well as rankings of users by in-degree. This will be the data used for comparison with rankings generated by the HITS algorithm.

Various types of graphs will be input into the HITS algorithm to look for influential users.

Figure 5.2: A fully inter-connected directed graph, also called a “clique”.

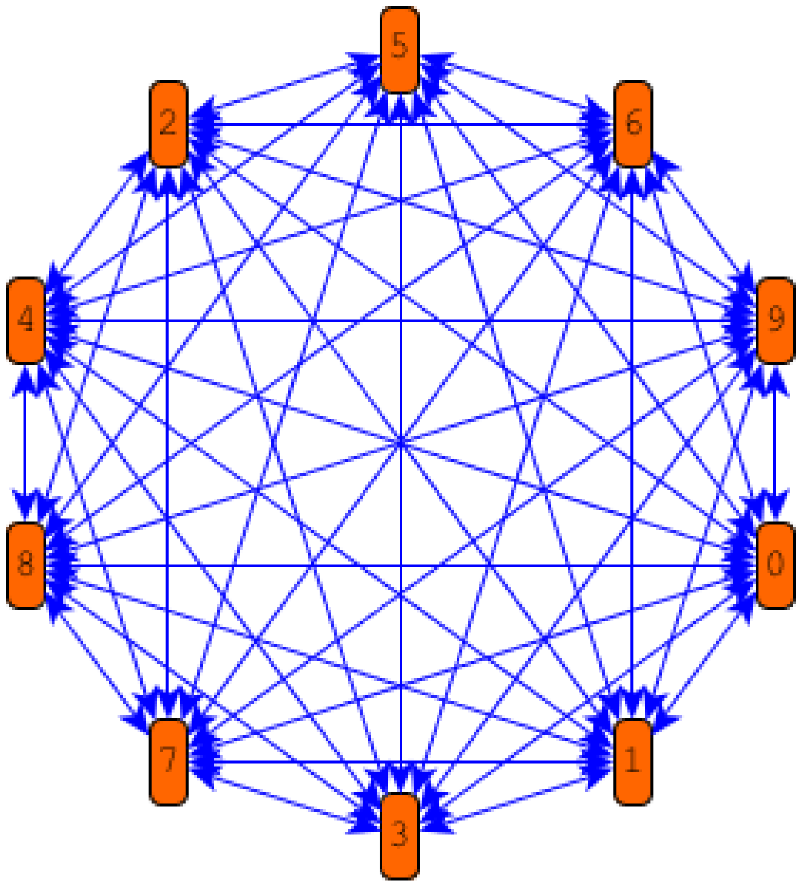


Table 5.2: Hub and Authority values corresponding to the graphs in figure 5.1(b).

Rank	Hub	Value	Auth.	Value
1	2	0.5	1	0.7071
2	3	0.5	6	0.7071
3	4	0.5	2	0.0
4	5	0.5	3	0.0
5	1	0.0	4	0.0
6	6	0.0	5	0.0

Types of graphs include the declared social graphs with variations of that graph as well as graphs generated from how users are interacting with each other with addressed messages or with username mentions.

5.2.1 Global Friends/Followers Graph

The global friends/followers graph is considered to be the aggregate data returned for each user's individual social graph who were speaking about the Boulder fire. As shown in table 4.13, the number of users speaking of the Boulder fire on the first day is small (448 users). However, the total number of unique IDs that are only one hop away from those users is 821,609. The initial analysis of the HITS algorithm will include all of the unique users seen in the data collection, as well as pruning of the user list by the frequency by which they appear.

Table 5.4 shows the rankings produced by HITS on varying input graphs after the first day of the fire on September 7th, 2010 at 12:40pm Mountain time. The first column includes all 821,609 users from the first day, and subsequent columns reduce the number of users included in the graph. To reduce the number, users are first sorted descending by the frequency in which they appear in the social graph for this group, and then the top N users are retained in the graph, and all other users and edges are culled out. In the second and third columns, the top 20,000 and 5,000 most frequent users are retained in the graph. The fifth column is a reproduction of the ranking of all users by in-degree from table 4.14 as a baseline comparison.

As can be seen in the table, users who already have a high-indegree are highly ranked by

Table 5.3: Hub and Authority values corresponding to the graphs in figure 5.2.

Rank	Hub	Value	Auth.	Value
1	1	0.3162	1	0.3162
2	2	0.3162	2	0.3162
3	3	0.3162	3	0.3162
4	4	0.3162	4	0.3162
5	5	0.3162	5	0.3162
6	6	0.3162	6	0.3162
7	7	0.3162	7	0.3162
8	8	0.3162	8	0.3162
9	9	0.3162	9	0.3162
10	10	0.3162	10	0.3162

HITS. Even though user "BarackObama" did not speak about the Boulder fire, he is included in the set because many of the users who do speak of the Boulder fire also link to "BarackObama". This causes this user to appear to have a high in-degree, and therefore ranked high among the users in the set.

5.2.2 Keyword Friends/Followers Graph

A variation on the input graph into HITS is the graph only containing the users who speak directly about the fire. This graph eliminates any users who are not actively speaking about the fire, therefore targeting the very users in which I am interested. This eliminates the issue with users like "BarackObama" from appearing in the set so active users can be targeted for influence. To create this graph, only users who have used a keyword related to the Boulder fire are considered and their declared friend/follower edges included in the graph.

Table 5.5 shows the HITS rankings of the users speaking of the Boulder fire on the graph collected on September 7th, 2010 at 12:40pm Mountain time. For comparison, the in-degree rankings of the same users are given in the second column, only counting edges among those speaking of the Boulder fire. Comparing the two columns, HITS is performing no better than simply ranking users by in-degree, and in fact only differ by two users in each set. The HITS ranking includes users "benjaminchait" and "highfiredanger" at rankings 23 and 25, respectively. The in-degree ranking

Table 5.4: Rankings of the top 25 users produced by the HITS algorithm on the social graph collected after the first day of the fire on September 7th, 2010 at 12:40pm Mountain. Column one includes all users, and columns two and three only include the top N most frequently appearing users. The fourth column is a reproduction of all users ranked by in-degree from table 4.14 as a baseline comparison. Users highlighted in bold are part of the qualitatively influential set listed in table 4.4.

Rank	All (821,609)	Top 20,000	Top 5,000	In-degree
1	Scobleizer	shellykramer	andrewhyde	Scobleizer
2	shellykramer	EthanJaynes	shellykramer	marcelomedici
3	andrewhyde	Scobleizer	**Mediamum**	PublishersWkly
4	HowardKurtz	andrewhyde	cbs4denver	HowardKurtz
5	gwenbell	studentoflife	Scobleizer	shellykramer
6	PublishersWkly	KaliMarcum	downtownboulder	taylorphinney
7	EthanJaynes	Freedomman11	EthanJaynes	EthanJaynes
8	agahran	pops131	gwenbell	mr_q
9	Karoli	**Mediamum**	coloradodaily	gwenbell
10	davidcohen	2drinksbehind	Greeblemonkey	FoxSports
11	2drinksbehind	cbs4denver	dailycamera	andrewhyde
12	Greeblemonkey	shannonevans	studentoflife	THEKASESHOW
13	shannonevans	gwenbell	Tekee	Freedomman11
14	**Mediamum**	salivates	OnlyInBoulder	KaliMarcum
15	bpm140	Greeblemonkey	bouldertweep	cbs4denver
16	jenmyronuk	downtownboulder	ejoep	craigslistjobs
17	mtlb	eraseyourdebt	hlane	Karoli
18	studentoflife	coloradodaily	KezzaMcDezza	Jastewart15
19	casinclair	wind4me	jenmyronuk	pops131
20	AmeriCares	dailycamera	2drinksbehind	bigdandbubba
21	KaliMarcum	AmeriCares	melsidwell	salivates
22	taylorphinney	hlane	davidcohen	_BellaCullen18_
23	wind4me	MaxSportsNet	wind4me	AmeriCares
24	delchoness	Tekee	**fishnette**	casinclair
25	cbs4denver	cabowabochris	BarackObama	eraseyourdebt

replaces these two users in the list with “Scobleizer” and “taylorphinney” at rankings 8 and 21, respectively.

5.2.3 Mentions Activity Graph

The mentions graph is created by parsing the messages created by users speaking of the Boulder fire during the event looking for usernames that appear anywhere in the text. The same process is used as the one that generated the data in tables 4.7 and 4.16 except the source user mentioning that user is saved. An edge source is a user A who has mentioned user B, who is the target. Multiple edges may be created within a single message if a source user A mentions multiple users in the text. This graph corresponds to the idea of an “activity network” or “interaction graph” as described in section 3.12. The idea is to create a graph representation of user interactions in place of the declared social graph as users only interact with about 10% of their declared friends [52].

Table **??** shows users in the mentions activity graph ranked in HITS during the first day of the Boulder fire from September 6th, 2010 at 10:00am Mountain to September 7th, 2010 at 10:00am Mountain. The first column includes all usernames encountered even if those target users were not speaking of the Boulder fire, and the second column is the ranking in this graph by in-degree for comparison. The third column is rankings of only the users speaking of the Boulder fire and the fourth column is the ranking in this smaller graph by in-degree for comparison. The all-users mentions graph contains 2,794 users with 4686 edges. The active-users mentions graph contains 261 users with 1,077 edges.

Again, the issue of no real difference between in-degree ranking and HITS is apparent in the table. Even more of an issue is that the percentage of users actually mentioning others in the active set, for example, is only 58.25% (261 users out of 448 during the first day). This method of using mentions has the potential of leaving out influential people who aren’t mentioning others at all, and yet are interacting heavily in the network during the event. We do see more of the qualitatively influential users in this table, but we are still missing about half of them (8-9 out of 16).

Table 5.5: Rankings of the top 25 users produced by the HITS algorithm on the social graph collected after the first day of the fire on September 7th, 2010 at 12:40pm Mountain. The graph only includes those users and edges who used a specific keyword in the Boulder fire. The second column is a reproduction of these users ranked by in-degree from the right half of table 4.14. Users highlighted in bold are part of the qualitatively influential set listed in table 4.4. Users with an asterisk are unique per set.

Rank	HITS	In-degree
1	andrewhyde	andrewhyde
2	downtownboulder	dailycamera
3	dailycamera	downtownboulder
4	coloradodaily	coloradodaily
5	**fishnette**	cbs4denver
6	cbs4denver	**fishnette**
7	**Mediamum**	davidcohen
8	Tekee	Scobleizer*
9	melsidwell	**Mediamum**
10	OnlyInBoulder	OnlyInBoulder
11	davidcohen	melsidwell
12	gwenbell	Tekee
13	KezzaMcDezza	gwenbell
14	jenmyronuk	**HumaneBoulder**
15	bouldertweep	**epiccolorado**
16	hkoren	ejoep
17	hlane	Greeblemonkey
18	Greeblemonkey	gratzo
19	ejoep	bouldertweep
20	fionaschlachter	jenmyronuk
21	gratzo	taylorphinney*
22	**HumaneBoulder**	KezzaMcDezza
23	benjaminchait*	hkoren
24	**epiccolorado**	hlane
25	highfiredanger*	fionaschlachter

Table 5.6: Rankings of the top 25 users produced by the HITS algorithm on the *mentions* activity graph during the first day of the Boulder fire from September 6th, 2010 at 10:00am Mountain to September 7th, 2010 at 10:00am Mountain. Columns one and two include all username mentions even if those users were not speaking of the Boulder fire and ranked by HITS and in-degree, respectively. Columns three and four only include the users speaking of the boulder fire ranked by HITS and in-degree, respectively. Users highlighted in bold are part of the qualitatively influential set listed in table 4.4.

	All Users (2,794)		Active Users (261)	
Rank	HITS	In-degree	HITS	In-degree
1	**fishnette**	andrewhyde	andrewhyde	andrewhyde
2	andrewhyde	**fishnette**	**fishnette**	**fishnette**
3	**epiccolorado**	cbs4denver	**epiccolorado**	cbs4denver
4	cbs4denver	**epiccolorado**	cbs4denver	**epiccolorado**
5	**mediamum**	**mediamum**	**mediamum**	**mediamum**
6	denverchannel	kwgndenver	kwgndenver	kwgndenver
7	bouldercounty	denverchannel	**humaneboulder**	dailycamera
8	kwgndenver	jamesazure	coloradodaily	coloradodaily
9	jamesazure	dailycamera	dailycamera	**humaneboulder**
10	twitter	bouldercounty	wind4me	kkartphoto
11	shellimeyers	coloradodaily	hlane	wind4me
12	**humaneboulder**	twitter	hkoren	hlane
13	kgnu	**humaneboulder**	**campsteve**	**campsteve**
14	**suzanbond**	kkartphoto	cuindependent	downtownboulder
15	**sandrafish**	**boulderchannel1**	downtownboulder	hkoren
16	coloradodaily	ev	pachecod	melsidwell
17	wind4me	**suzanbond**	sarajuliet	cuindependent
18	hlane	redcrossdenver	colo_kea	pachecod
19	hkoren	**laurasrecipes**	greeblemonkey	greeblemonkey
20	dailycamera	denverpost	kkartphoto	timescall
21	ev	wind4me	kktv11news	sallyfrancklyn
22	**boulderchannel1**	hlane	joshlarson	scobleizer
23	**campsteve**	**campsteve**	melsidwell	emergcommnetwrk
24	redcrossdenver	**sandrafish**	mattbeaty	mattbeaty
25	cuindependent	downtownboulder	tuckertown	sarajuliet

5.2.4 Addressed Messages Graph

The next type of graph to be analyzed is the addressed messages graph. This is a subset of the mentions graph discussed above as only the messages considered "addressed" to another user are counted as an edge. Addressed messages are explained in section 4.5.2 and addressed message counts are shown in table 4.8 for the first day and first week of the Boulder fire. This exact information of source and target user is used in the creating of the addressed messages graph.

Table 5.7 shows the top 25 recipients of addressed messages during the first 24 hours of the Boulder fire from September 6th, 2010 at 10:00am Mountain to September 7th, 2010 at 10:00am Mountain. The first two columns include all users encountered, even if they were not speaking of the fire. Columns three and four only contain users who were speaking of the Boulder fire. The all-users graph contains 1,177 users and 1,345 edges, and the active users graph contains 261 users with 258 edges between them. Again some of the qualitative users are seen in the list, but a number of them are still missing.

5.3 Analysis and Discussion

Four different types of graphs have been constructed for input into the HITS algorithm: the global friends/followers graph, the keyword friends/followers graph, the mentions graph and the addressed messages graph. Including variations of each graph, a total of eight graphs were input into HITS. For comparison, the rankings by in-degree for each graph was included as a baseline comparison.

Ranking users by in-degree is not a sufficient measure to find influential users. This problem has motivated research in areas of betweenness centrality [42], eigenvector centrality calculations such as HITS here [65] and PageRank [16], or k-shell decomposition [64]. HITS has been used to analyze the propagation of URLs within Twitter [101], or for finding influential users on the social graph just like the analysis in this chapter [57].

Although HITS is an attractive algorithm for use in the context of finding influential users

Table 5.7: Rankings of the top 25 users produced by the HITS algorithm on the *addressed messages* activity graph during the first day of the Boulder fire from September 6th, 2010 at 10:00am Mountain to September 7th, 2010 at 10:00am Mountain. Columns one and two include all target users of addressed messages even if those users were not speaking of the Boulder fire and ranked by HITS and in-degree, respectively. Columns three and four only include the users speaking of the boulder fire ranked by HITS and in-degree, respectively. Users highlighted in bold are part of the qualitatively influential set listed in table 4.4.

	All Users (1,177)		Active Users (130)	
Rank	HITS	In-degree	HITS	In-degree
1	**fishnette**	**fishnette**	**fishnette**	**fishnette**
2	andrewhyde	andrewhyde	andrewhyde	andrewhyde
3	sarajuliet	**epiccolorado**	**epiccolorado**	**epiccolorado**
4	wind4me	wind4me	wind4me	wind4me
5	eatplaylove	joeyschusler	**mediamum**	**mediamum**
6	**epiccolorado**	**mediamum**	sarajuliet	cbs4denver
7	stiricide	**laurasrecipes**	melsidwell	hlane
8	tashigonpo108	eatplaylove	cbs4denver	sarajuliet
9	vococreative	cbs4denver	priscillast	melsidwell
10	joeyschusler	stiricide	scobleizer	kwgndenver
11	**mediamum**	**suzanbond**	**tanukun**	scobleizer
12	**laurasrecipes**	sarajuliet	tekee	theinnermarykay
13	davidherrold	kwgndenver	greeblemonkey	suesalinger
14	tekee	kyleindenver	joshlarson	colo_kea
15	morganbast	hlane	wiscobeth	lioncaller
16	greeblemonkey	melsidwell	suesalinger	joelwish
17	**tanukun**	**sandrafish**	paullewin	mattbeaty
18	bouldercounty	suesalinger	mattbeaty	chrisennis
19	joshlarson	**kate30_cu**	joelwish	tuckertown
20	scobleizer	nuancechaser	chrisennis	**tanukun**
21	melsidwell	colo_kea	ofuttufo	thenoodleator
22	userealbutter	tashigonpo108	pachecod	wiscobeth
23	vanillagrrl	liminalison	michaeldwan	joseph_flasher
24	kyleindenver	scobleizer	dhendersonco	theroseinbloom
25	priscillast	davetaylor	nattyz	woodardj

within a specific event, the situation is not analogous to the original problem described by Kleinberg. The data collection process for this research does include a search performed on the network for messages matching a set of keywords followed by the collection of the social graph for those users representing web pages in the original algorithm context, so it appears to be very similar. The original problem was web page search results on old search engines such as HotBot[1] which returned results based on keyword frequency within the web pages. This led to hosting sites inflating their result rankings by hiding keywords among the page that are not visible to human readers, but would be seen by web crawler bots.

The key point as to why the two situations are not analogous is that web pages typically represent one or very few topics. A node in the input graph is a page, and links to other pages rarely include pages of a different topic. However, users within social networks represent what could be considered many different "topics" within their lives and links represent connections to other users (pages) that are significantly varied between one another. The edges represent many different relationships, such as co-worker, gaming friend, exercise partner, sister, and many others. This is the fundamental issue that causes the HITS algorithm to perform no better than ranking users by in-degree only. HITS by design places all nodes in the same context, and as such the results are confounded among many relationship types. Further, users tend to link to each other with very little impetus leading to a very densely connected graph, which is also unlike the internet web graph.

The conclusion is that the HITS algorithm applied to specific events is not very effective. However, HITS may still be useful if an even has very large scale, and could possibly cull out users who may be discussing an event without being involved such as the Haiti earthquakes, for example. The speed and ability to parallelize the HITS algorithm would make it attractive for this purpose. Further analysis would be required on the subset of users returned to find influencers within the context of the event. More analysis is warranted on HITS, but for small events the algorithm is not effective.

[1] HotBot no longer exists in the same form as it did in 1998.

Chapter 6

Context-Specific Indegree Ranking

Here I examine a dynamic approach to finding influential users within a specific context. The Boulder fire data will once again be used in this analysis. I develop an algorithm that records the changes over time to the social graph among users speaking of the Boulder fire. The context of this ranking algorithm is built around the messages collected from the Twitter network using the keywords described in section 4.5. Every 30 minutes the unique list of users appearing in the search results were obtained and then those users' friends and followers lists were collected. In this chapter only one-day increments are reviewed for brevity.

The idea is to observe the change in the network over time in order to determine who is being listened to the most. Twitter does not save the timestamp in which a user follows another user which motivated the 30-minute collection window in order to see the dynamics of the network. In this chapter the one-day incremental data described in table 4.13 is used for the dynamic analysis.

6.1 Ranking by Global Follower Counts

The first way to look at the dynamics of the network is to simply look at the global friend/follower counts that appear in the profile for all users. Table 4.1 shows two integer values, *friends_count* and *followers_count*, corresponding to the current number of friends and followers for a user at the moment the profile is collected. Algorithm 5 describes the process for counting the change in *followers_count* values in profiles collected with the friend/follower lists for each user involved in the Boulder fire.

The algorithm takes snapshots of the profiles for each user speaking about the Boulder fire at the times described in table 4.13 as previously mentioned. For each snapshot, the *followers_cont* value is stored for each user and changes in this value are computed for each subsequent snapshot. Users who are added to the network later in the event are simply given count values of -1 and delta values of 0. Rankings can be computed on either the overall counts or the largest deltas at any snapshot to inspect what is occurring in the social graph at that moment.

Table 6.1 shows the rankings of users using algorithm 5 over the five network snapshots each one day apart of the Boulder fire from September 7th, 2010 at 12:40pm Mountain to September 11th, 2010 at 3:10pm Mountain. Only the change in follower counts are shown in the table instead of the raw counts. Deltas with a value of zero occur due to the user appearing later during the Boulder fire event and therefore the social graph was not collected at that time. Although we do see three users from the qualitatively influential set, most users that appear are already popular in the network in some way. Users such as “NASA”, “Scobleizer” and various news outlets appear to have large gains in followers globally within Twitter during this time period. Using only profile *follower_count* metadata, one is unable to determine where these connections are being formed and why.

6.2 Ranking by Active Users with Pre-existing Network

To improve upon the algorithm given in the previous section, I eliminate edges in the social graph to users who are not involved in the event. This is a first step in eliminating users who are globally influential in some manner such as those described in the previous section. The graph analyzed in this algorithm is identical to the graph in section 5.2.2 as it only considers edges among users who are speaking of the Boulder fire.

Algorithm 6 describes the process for counting connections made only among those speaking about the Boulder fire. The algorithm functions differently than the profile algorithm as it operates on the social graph snapshots collected during the Boulder fire. As users speak of the Boulder fire using the keywords, they are added to a set of users to be collected. These users are contained in

Algorithm 5 Calculates the change in the global profile *follower_count* values for users considered active (using a keyword) during an event.

```
1:  Given the following:
2:  List of profile snapshots $\mathcal{S}$
3:  Assign $snapshotIndex = 0$
4:  Assign $inDegreeCounts = \{\}$
5:  Assign $inDegreeDeltas = \{\}$
6:  for each snapshot $\mathcal{P}_i \in \mathcal{S}$ do
7:      for each profile $p \in \mathcal{P}_i$ do
8:          Assign $username = p['screen\_name']$
9:          if $username \notin inDegreeCounts$ then
10:             Assign $inDegreeCounts\{username\} = []$
11:             Assign $inDegreeDeltas\{username\} = []$
12:             for $i \in$ range[0,$snapshotIndex$) do
13:                 Append $-1$ to $inDegreeCounts\{username\}$
14:                 if $i < snapshotIndex - 1$ then
15:                     Append 0 to $inDegreeDeltas\{username\}$
16:                 end if
17:             end for
18:         end if
19:         Assign $count = p\{'followers\_count'\}$
20:         Append $count$ to $inDegreeCounts\{username\}$
21:         if $snapshotIndex > 0$ then
22:             if $inDegreeCounts\{username\}\{snapshotIndex\} \neq -1$ then
23:                 Assign $delta = count - inDegreeCounts\{username\}\{snapshotIndex - 1\}$
24:                 Append $delta$ to $inDegreeDeltas\{username\}$
25:             else
26:                 Append 0 to $inDegreeDeltas\{username\}$
27:             end if
28:         end if
29:     end for
30:     Assign $snapshotIndex = snapshotIndex + 1$
31: end for
32: Return $inDegreeCounts$,$inDegreeDeltas$
```

Table 6.1: Rankings of the top 25 users speaking of the Boulder fire during September 7th, 2010 through September 11th, 2010 by the most followers gained globally within Twitter. Users highlighted in bold are from the qualitatively important set listed in table 4.4.

Rank	Name	Delta 1	Delta 2	Delta 3	Delta 4	Total
1	NASA	0	1818	1532	1732	5082
2	marcelomedici	478	514	404	476	1872
3	THEKASESHOW	-12	531	468	530	1517
4	PublishersWkly	281	324	298	269	1172
5	Scobleizer	427	263	233	218	1141
6	HowardKurtz	128	149	169	117	563
7	zaibatsu	0	120	168	221	509
8	alwaysbestrts	0	92	282	120	494
9	DellU_MA	0	121	163	108	392
10	adventurevida	0	144	94	123	361
11	fema	0	121	121	90	332
12	thaz7	0	9	259	32	300
13	LGEsolutions	89	46	43	57	235
14	**epiccolorado**	**58**	**73**	**89**	**15**	**235**
15	LauraMoore7	110	30	36	35	211
16	dailycamera	41	46	93	19	199
17	USDAgov	0	81	80	37	198
18	**laurasrecipes**	**0**	**73**	**85**	**37**	**195**
19	taylorphinney	59	55	41	35	190
20	10rWfe	0	24	98	63	185
21	tlrd	0	6	94	78	178
22	**HumaneBoulder**	**77**	**39**	**34**	**27**	**177**
23	MissingScoop	0	26	97	44	167
24	TheFireTracker2	0	4	161	2	167
25	thundercatsnyy	0	32	94	38	164

the set $\mathcal{U}_\sigma$ in the algorithm. If a user is not a part of this set, she is ignored when counting the number of followers for an active user in the event. Line 14 shows the calculation of the size of the set resulting from the intersection of the set of users speaking of the event and the current user's follower list from the previous line.

Table 6.2 shows the top 25 users ranked by the number of followers gained in the event only among users speaking of the event as calculated by algorithm 6 during the same time period from September 7th, 2010 at 12:40pm Mountain to September 11th, 2010 at 3:10pm Mountain. The qualitatively influential users who do not fall in the top 25 are listed in the lower section of the table for reference. Three users are missing completely from the set, specifically **kate30_CU**, **eadvocate**, and **BoulderChannel1**. These users are missing from most of the data collection due to errors in the collection process.

Half of the qualitatively influential set are still not appearing high on the rankings list, and a number of globally influential users are ranked highly once again. This is due to pre-existing edges counting in favor of users active within the network. The inflation of rank in this regard occurs when a user begins interacting during the event any time after the start of the event. The situation occurs as follows:

(1) User A has a pre-existing connection to user B

(2) The Boulder fire event begins

(3) User A speaks of the Boulder fire using a keyword

(4) The social graph for user A is collected along with those of all others speaking of the event

(5) User B speaks of the Boulder fire at some time Δt after N snapshots of the event network are taken

(6) The pre-existing edge from user A to user B counts as a gain of one follower for user B based on the logic in algorithm 6.

Algorithm 6 Calculates the change in follower lists of the users considered active (using a keyword) during an event, ignoring all other users who appear in the follower lists and are not speaking of the event.

```
1: Given the following:
2: Set of all users $\mathcal{U}_\sigma$ involved in the event
3: List $\mathcal{S}$ of sets of users involved at each snapshot of the event
4: List $\mathcal{F}$ of user follower sets for each user in $\mathcal{U}_\sigma$
5: Assign $inDegreeCounts = \{\}$
6: Assign $inDegreeDeltas = \{\}$
7: for each $user \in \mathcal{U}_\sigma$ do
8:    Assign $currentUsers = []$
9:    for $snapshotIndex \in$ range[0,$|\mathcal{F}|$) do
10:       Assign $currentUsers = \mathcal{S}[snapshotIndex]$
11:       if $user \in currentUsers$ then
12:          Assign $followers = \mathcal{F}\{user\}[snapshotIndex]$
13:          Assign $common = currentUsers \cap followers$
14:          Assign $count = |common|$
15:          Append $count$ to $inDegreeCounts\{user\}$
16:       else
17:          Append $-1$ to $inDegreeCounts\{user\}$
18:       end if
19:       if $snapshotIndex > 0$ then
20:          Assign $prevCount = inDegreeCounts\{user\}[snapshotIndex - 1]$
21:          Assign $curCount = inDegreeCounts\{user\}[snapshotIndex]$
22:          if $prevCount \neq -1$ and $curCount \neq -1$ then
23:             Append $(curCount - prevCount)$ to $inDegreeDeltas\{user\}$
24:          else
25:             Append 0 to $inDegreeDeltas\{user\}$
26:          end if
27:       end if
28:    end for
29: end for
30: Return $inDegreeCounts$,$inDegreeDeltas$
```

Table 6.2: Rankings of the top 25 users speaking of the Boulder fire during September 7th, 2010 through September 11th, 2010 by the most followers gained only among those speaking of the Boulder fire. In this ranking, any pre-existing edges between active users that existed before the start of the fire count as a gain in followers. Users highlighted in bold are from the qualitatively important set listed in table 4.4.

Rank	Name	Delta 1	Delta 2	Delta 3	Delta 4	Total
1	cbs4denver	211	4	1	165	381
2	dailycamera	194	2	9	150	355
3	downtownboulder	184	0	2	161	347
4	andrewhyde	185	2	4	145	336
5	coloradodaily	162	0	2	130	294
6	NASA	0	0	0	244	244
7	Scobleizer	101	0	1	138	240
8	**HumaneBoulder**	**107**	**4**	**3**	**95**	**209**
9	OnlyInBoulder	113	0	2	76	191
10	**epiccolorado**	**73**	**16**	**15**	**87**	**191**
11	**fishnette**	**103**	**4**	**12**	**66**	**185**
12	zaibatsu	0	0	1	182	183
13	gwenbell	98	-1	1	72	170
14	CFHeather	0	-1	0	166	165
15	**ConnectColorado**	**0**	**7**	**3**	**152**	**162**
16	**Mediamum**	**82**	**7**	**2**	**70**	**161**
17	hlane	83	5	3	60	151
18	BrettGreene	0	9	5	136	150
19	KDVR	74	2	3	68	147
20	Greeblemonkey	85	-1	3	60	147
21	shellykramer	64	0	0	76	140
22	davidcohen	85	1	0	49	135
23	bouldertweep	67	0	1	57	125
24	DenverChannel	0	3	1	119	123
25	Tekee	72	-1	0	49	120
37	**laurasrecipes**	**0**	**19**	**11**	**75**	**105**
45	**metroseen**	**0**	**7**	**1**	**84**	**92**
57	**CampSteve**	**23**	**2**	**3**	**50**	**78**
59	**suzanbond**	**0**	**14**	**9**	**51**	**74**
80	**Org9**	**0**	**13**	**2**	**43**	**58**
123	**sophiabliu**	**0**	**2**	**6**	**36**	**44**
163	**palen**	**10**	**0**	**4**	**20**	**34**
427	**Tanukun**	**7**	**3**	**0**	**2**	**12**

6.3 Ranking by Active Users with New Edges

To eliminate the inflation of a user's in-degree ranking that occurs from the process in algorithm 6, a simple modification is made to the algorithm to prevent pre-existing edges from counting in favor of a user. Line 9 includes the addition of a new variable that holds a reference to the first set of followers seen for this user as assigned on lines 13 to 14. This set is used for set subtraction from any subsequent follower set at any future snapshot which occurs on line 18. This prevents any pre-existing edge counting in favor of a user.

Table 6.3 shows the rankings of the top 25 users speaking of the Boulder fire during the same time window. Immediately seen are the top seven users ranked are part of the qualitatively influential list as well as 12 of the 16 qualitatively influential users appearing within the top 23 users overall. Aside from the three users who do not appear at all within the data set, user **Tanukun** appears very low in the list. All other users that appear in the list are related to the Denver/Boulder area in some way. News outlets or personnel include users **dailycamera**, **kwgndenver**, **BrettGreene**, **coloradodaily**, and **cbs4denver**. Local city or county organizations include **bouldercounty**, **bouldercolorado**, **boulderpolice**, and **redcrossdenver**. The remaining users are all local individuals: **Colo_kea**, **LizEmmettMattox**, **andrewhyde**, and **SchwartzNow**.

6.4 Analysis and Discussion

The time for each algorithm is similar as each has to iterate over the users in each snapshot, resulting in a baseline performance of $O(ns)$ where n is the average number of users over all snapshots, and s is the number of snapshots. This is the performance of algorithm 5 as it only has to compute count derivatives for the integer value given in each profile snapshot. Algorithms 6 and 7 require extra steps to perform set operations in order to find the users common to both the current active set and the set of followers for any particular user, and also subtract the pre-existing edges from user follower sets in algorithm 7. This costs an additional factor d which is the average in-degree of all users in the set, resulting in an overall running time of $O(nsd)$.

Algorithm 7 Calculates the change in follower lists of the users considered active (using a keyword) of an event, ignoring all other users who appear in the follower lists and are not speaking of the event as well as ignoring any edges that existed between active users before the event started.

```
1: Given the following:
2: Set of all users $\mathcal{U}_\sigma$ involved in the event
3: List $\mathcal{S}$ of sets of users involved at each snapshot of the event
4: List $\mathcal{F}$ of user follower sets for each user in $\mathcal{U}_\sigma$
5: Assign $inDegreeCounts = \{\}$
6: Assign $inDegreeDeltas = \{\}$
7: **for** each $user \in \mathcal{U}_\sigma$ **do**
8:    Assign $currentUsers = []$
9:    Assign $initFollowers = null$
10:   **for** $snapshotIndex \in$ range[0,$|\mathcal{F}|$) **do**
11:      Assign $currentUsers = \mathcal{S}[snapshotIndex]$
12:      **if** $user \in currentUsers$ **then**
13:         **if** $initFollowers = null$ **then**
14:            Assign $initFollowers = \mathcal{F}\{user\}[snapshotIndex]$
15:         **end if**
16:         Assign $followers = \mathcal{F}\{user\}[snapshotIndex]$
17:         Assign $common = currentUsers \cap followers$
18:         Assign $count = |common \setminus initFollowers|$
19:         Append $count$ to $inDegreeCounts\{user\}$
20:      **else**
21:         Append $-1$ to $inDegreeCounts\{user\}$
22:      **end if**
23:      **if** $snapshotIndex > 0$ **then**
24:         Assign $prevCount = inDegreeCounts\{user\}[snapshotIndex - 1]$
25:         Assign $curCount = inDegreeCounts\{user\}[snapshotIndex]$
26:         **if** $prevCount \neq -1$ and $curCount \neq -1$ **then**
27:            Append $(curCount - prevCount)$ to $inDegreeDeltas\{user\}$
28:         **else**
29:            Append 0 to $inDegreeDeltas\{user\}$
30:         **end if**
31:      **end if**
32:   **end for**
33: **end for**
34: Return $inDegreeCounts$,$inDegreeDeltas$
```

Table 6.3: Rankings of the top 25 users speaking of the Boulder fire during September 7th, 2010 through September 11th, 2010 by the most followers gained only among those speaking of the Boulder fire. In this ranking, only new edges created between active users count as a gain in followers. Any pre-existing edges between users are ignored. Users highlighted in bold are from the qualitatively influential set listed in table 4.4.

Rank	Name	Delta 1	Delta 2	Delta 3	Delta 4	Total
1	**epiccolorado**	**16**	**16**	**15**	**43**	**90**
2	**laurasrecipes**	**0**	**19**	**11**	**32**	**62**
3	**HumaneBoulder**	**17**	**4**	**3**	**26**	**50**
4	**fishnette**	**6**	**5**	**14**	**24**	**49**
5	**suzanbond**	**0**	**16**	**10**	**16**	**42**
6	**CampSteve**	**1**	**2**	**3**	**32**	**38**
7	**ConnectColorado**	**0**	**7**	**3**	**26**	**36**
8	dailycamera	6	2	9	16	33
9	bouldercounty	0	11	9	12	32
10	kwgndenver	3	8	2	11	24
11	bouldercolorado	0	3	9	12	24
12	**Org9**	**0**	**13**	**2**	**9**	**24**
13	BrettGreene	0	9	5	9	23
14	coloradodaily	5	0	2	15	22
15	Colo_kea	3	6	1	11	21
16	**metroseen**	**0**	**7**	**1**	**13**	**21**
17	**Mediamum**	**3**	**8**	**2**	**7**	**20**
18	LizEmmettMattox	3	11	1	4	19
19	**palen**	**0**	**1**	**4**	**14**	**19**
20	cbs4denver	4	4	1	9	18
21	andrewhyde	5	3	4	5	17
22	boulderpolice	0	5	4	7	16
23	**sophiabliu**	**0**	**2**	**7**	**7**	**16**
24	redcrossdenver	0	0	3	13	16
25	SchwartzNow	0	3	3	10	16
146	**Tanukun**	**0**	**3**	**0**	**1**	**4**

As mentioned in section 6.2, users missing from the data collection are due to errors in the collection process. These errors are easily remedied through more robust collection techniques. However, given these collection errors, the results of algorithm 7 clearly yield influential users during an event, specifically 12 out of the 16 qualitatively influential users are found. These results indicate a significant improvement in ranking over every other metric used in this research, including all statistical measures in chapter 4, all variations of the HITS algorithm described in chapter 5, and in first two algorithms presented in this chapter.

However, some limitations of the algorithm exist, as illustrated by user **Tanukun**. This user appears very low in the list indicating other features account for this user's influence that are not illustrated by this data. Also, this data alone is a summation of the total change in followers by users active within the event. An improvement to be made on the algorithm is to separate the total change into separate loss and gain counts of followers among the users speaking of the event. This could possibly inform of activity that may be considered untenable by users in the network, such as user **BoulderChannel1** who is considered to be a delinquent. A limitation that will never be addressed is the fact that gains or losses in followers in between snapshot times may be missed if a single user A un-follows and re-follows user B. Also, this approach does not properly identify users as being influential who are already influential among the population as it only measures new links being created in the network during the event. Therefore users like **andrewhyde** who are locally popular in Boulder may be very important, but are not discovered through this algorithm.

Aside from the intuitive nature of this algorithm counting the gain in followers during an event, the rankings can be viewed within the context of a specific snapshot in order to determine who is entering the network at that time and gaining a following. This approach does not account for on-going influential measures after a followers are gained, and that information must be gained through analyzing the content of the messages sent among the network. Also, news outlets appear in the list, and using other statistical or natural language processing techniques, there may be ways of categorizing these accounts through their tweeting patterns to find bot or news behavior in order to remove these users from the rankings.

The process in algorithm 7 appears to be a good start to finding many influential users, but will need to be augmented by other means in order to be robust in many varying events.

Chapter 7

Future Work and Conclusions

In this research I have described the four layers of the problem of attempting to infer reputation on users during an event, reviewed literature for each of the areas touched by those four layers, investigated the data available on Twitter, analyzed the existing HITS graph ranking algorithm, and proposed my own ranking algorithm for finding influencers during an event on the Twitter network verified by qualitative data. This research area is rich with problem areas and hypotheses to be tested.

7.1 Future Work

The problem space for this research is quite large as described in chapter 2. Numerous issues arise when trying to answer the questions in each layer as described in figure 2.1. Even simply finding *influencers* immediately introduces the problem of understanding *why* users are influential. Definitions are also a problem, and must be clearly stated when working on these issues. The different mechanisms that can be used to define "influence", such as number of followers, frequency of tweets, or frequency of username mentions each describe different types of influence, and clarification of what those types can be is necessary for understanding the different roles that users may play in the network.

The notion of reputation is becoming increasingly important as numerous organizations and websites are appearing which are discussing the need of determining online reputation or trying to quantitatively infer reputation on people or information. Blogger Jeff McCord discusses how online information is affecting recruiters interviewing candidates for positions [77]. An example of a company attempting to quantify reputation of users on Twitter is Klout which has developed a reputation ranking based on numerous variables similar to those described in chapter 4 [67]. Former Twitter CEO Evan Williams recently mentioned the fact that Twitter maintains internal reputation scores on users for its own recommendation features [76].

One of the biggest challenges in determining a quantitative reputation score on users or information is the diversity in how people use social networks. For example, the metrics used in the Klout score may apply to some users and not others depending on how they interact on Twitter. For example, the addressed messages as displayed in chapter 4 only account for in 15%-18% of all messages seen, and among those messages it's very difficult to determine whether or not the addressed message is a reply to another message (what Klout considers a spark in "conversation") or is simply a targeted initial message from one user to another. The essence of the issue is that the features of interaction are not equally used by everyone in the network. As such, all scores must be normalized to what is considered "average" behavior *per user*. This requires analysis of a user's historical activity on the network, much like that seen for user "fishnette" in chapter 4. Figure 4.4 illustrates this point where the average number of tweets per month is much lower than appears during the Boulder fire at the start of September in the graph.

A second challenge involves determining reputation within the context of a specific event as investigated in this research. The reputation scores mentioned previously from Klout and Twitter are for *global* reputation. This information may be interesting when looking at a user's activity over many months or years of activity, but is not useful when analyzing specific events. The same argument holds true in the issue illustrated by Jeff McCord [77] when potential employers will search for candidate information online in Facebook or elsewhere which can be taken out of context.

The remaining third major challenge is composing the different measures of user activity performed in chapter 4 or by techniques used by Klout into a conclusion of trust, reputation, or influence about a user. Composability was described in section 2.2.1 as the property of combining measures into a single conclusion just as this. Again, due to the variation of how users behave on the network, normalization of activity must be taken into account in order to make sense of the different activities, and great care must be taken to not confound two types of measures as measuring the same property of a user. Any measurements must also be explicitly and carefully defined to remove any ambiguity of what property is being measured about a user.

Reputation must be *validated* as described as the third layer of the overall problem. This requires incorporating multiple sources of data in order to cross-reference and verify claims made by users within the different events that are occurring. A clear direction to take within this layer is to expand to additional social networks or websites that contain public data in order to illustrate a broad picture of what is occurring online beyond Twitter. Shared links may be expanded and followed, major news websites may be tracked for news releases, and blogs may be monitored for activity around an event. This would require an expansion of data collection and a concise analysis plan to compose the new and existing data into a coherent form, as each of these different data sources have different attributes themselves. Blogs tend to be more thoughtful and come later within an event, and articles posted by news organizations lie somewhere between Twitter's real-time user stream and blog posts.

Security is a major concern in all of the three previous layers, but is listed at the top to be the governor of what information is publicly reported. Algorithm 7 described in chapter 6 is vulnerable to many types of attacks, such as a simple user interacting using keywords and generating followers by acting as an information source. More sophisticated techniques such as Sybil attacks would be easily employed to infiltrate an event activity network to manipulate who are considered influencers by the algorithm. Additional mechanisms of cross-validation, most likely with user location data or inference of location using the posted text is necessary in order to try to combat these sorts of attacks.

The data analysis in chapter 4 is only a first attempt at simple statistics and pattern analysis to assist in the security and validation layers. Many more interesting multi-variate questions may be asked of the data such as, "what words or other hashtags co-occur with any particular hashtag or keyword?", or "what are the similarities or differences among user profiles or messages of those who speak of the boulder fire?" In fact, many different applications of similarity measures such as those described in section 3.2 are possible among all the available data.

The methods in chapter 3 are worth exploring in terms of their efficacy and how they may expose patterns occurring within the users during an event. Analysis could follow a similar style much like the analysis performed on the HITS algorithm in chapter 5 with implementations and execution over various forms of the data from Twitter events.

Empirical measurements of the dynamics of the social network occurring during an event is possible as performed by algorithm 7 presented in chapter 6. With data collected from many different types of events, models can be developed to simulate activities on the network to predict where future links may be created, or determine the expected influence of a user or number of users who may fill those influential roles, all compared to measured empirical data from each event.

Many more techniques are worth investigating and are not fully explored here.

7.2 Conclusions

While the area of reputation scoring has a set of methods as reviewed in chapter 2, the area of determining reputation through inference is wide open for more investigation. As seen in chapter 2, few researchers can agree on the definitions of trust and reputation which creates difficulty in defining goals for inferring trust and reputation on users or data which have no associated explicit ranking scores. Ideas of what is meant by "influence" are also not in agreement, and many assumptions are made about graph structures that may not make sense in the context of social network graphs. Notions of centrality and betweenness may have applications in flow-based networks where the flow of information is governed by explicitly defined behaviors, such as routers on the internet. However, people are much less predictable in how they decide to allow information to flow over their links, and as such, each method, tool, or algorithm considered for use must always be analyzed through this type of lens.

The analysis in chapters 3 through 6 begin to demonstrate that features of the social network being analyzed (such as Twitter in the case of this research) largely determine how effective the available tools and algorithms will be in the analysis. For example, the process in algorithm 7 in section 6.3 would not apply to Facebook, for example, as the design of the friend/follower relationship is fundamentally different in Facebook and the algorithm would not find influential users in the same way. In addition to the differences in design, existing features are changed and new features are added requiring maintenance of existing data collection and algorithms to accommodate these changes.

The analysis of the HITS algorithm in chapter 5 yields results that do not perform any better than simply ranking users by their number of followers. Any variation of the social graph, whether taken from the declared friend/follower lists of users or from interaction graphs from how users are speaking to each other. These results are due to the fact that the graphs are very densely connected and do not differentiate between users who are and are not active during an event.

The algorithm developed in chapter 6 proves to be very useful in finding many of the qualitatively influential users listed in table 4.4. Although the method has some limitations as described in that chapter, it is a good first step toward finding influential users to begin to ask the follow-up question of “why are these users influential?”

To return to the original thesis question, I ask whether or not the question of ”are there tools available to determine influential users in social networks in the context of a specific event?” has been answered. The newly developed algorithm in chapter 6 performs well specifically in the Twitter network in the context of the Boulder fire, finding influencers. However, the question remains to be answered for future events of types and sizes different than the Boulder fire but that exist on Twitter. Further algorithms must be developed to incorporate other types of social networks or websites in use by people involved in these events.

Bibliography

[1] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6, page 6007, Washington, DC, USA, 2000. IEEE Computer Society.

[2] J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. k-core decomposition: a tool for the analysis of large scale internet graphs. CoRR Networks and Heterogeneous Media, 3:371–393, 2008.

[3] Jeffrey Jensen Arnett. The myth of peer influence in adolescent smoking initiation. Health Educ. Behav., 34:594–607, 2007.

[4] Yasuhito Asano, Yu Tezuka, and Takao Nishizeki. Improvements of hits algorithms for spam links. IEICE - Trans. Inf. Syst., E91-D(2):200–208, 2008.

[5] Ganesh Bagler and Somdatta Sinha. Assortative mixing in protein contact networks and protein folding kinetics. Bioinformatics, 23(14):1760–1767, 2007.

[6] James P. Bagrow. Evaluating local community methods in networks. J. Stat. Mech., 2008.

[7] James P. Bagrow and Erik M. Bollt. Local method for detecting communities. Phys. Rev. E, 72(4):046108, Oct 2005.

[8] Coralio Ballester, Antoni Calv-Armengol, and Yves Zenou. Who's who in networks. wanted: The key player. Econometrica, 74(5):1403–1417, 2006.

[9] Coralio Ballester, Antoni Calv-Armengol, and Yves Zenou. Delinquent networks. Journal of the European Economic Association, 8(1):34–61, 2010.

[10] Ziv Bar-Yossef and Li-Tal Mashiach. Local approximation of pagerank and reverse pagerank. In CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management, pages 279–288, New York, NY, USA, 2008. ACM.

[11] Vladimir Batagelj and Matjaz Zaversnik. An o(m) algorithm for cores decomposition of networks. CoRR, cs.DS/0310049:1–9, 2003.

[12] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pages 104–111, New York, NY, USA, 1998. ACM.

[13] Mikhail Bilenko and Ryen W. White. Mining the search trails of surfing crowds: identifying relevant websites from user activity. In WWW '08: Proceeding of the 17th international conference on World Wide Web, pages 51–60, New York, NY, USA, 2008. ACM.

[14] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3:993–1022, 2003.

[15] Ulrik Brandes. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology, 25:163–177, 2001.

[16] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst., 30(1-7):107–117, 1998.

[17] Javier Carbo, Jesus Garcia, and Jose M. Molina. Subjective trust inferred by kalman filtering vs. a fuzzy reputation. In Conceptual Modeling for Advanced Application Domains, volume 4389/3005, pages 496–505, 2004.

[18] Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. A model of internet topology using k-shell decomposition. Proc. Natl. Acad. Sci. USA, 104:11150–11154, 2007.

[19] James Caverlee, Ling Liu, and Steve Webb. Socialtrust: tamper-resilient trust establishment in online communities. In JCDL, pages 104–114, 2008.

[20] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. In IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pages 1–14, New York, NY, USA, 2007. ACM.

[21] Meeyoung Cha, Alan Mislove, and Krishna P. Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In WWW '09: Proceedings of the 18th international conference on World wide web, pages 721–730, New York, NY, USA, 2009. ACM.

[22] Alice Cheng and Eric Friedman. Manipulability of pagerank under sybil strategies. In First Workshop on the Economics of Networked Systems (NetEcon06), 2006.

[23] Hyunwoo Chun, Haewoon Kwak, Young-Ho Eom, Yong-Yeol Ahn, Sue Moon, and Hawoong Jeong. Comparison of online social relations in volume vs interaction: a case study of cyworld. In IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, pages 57–70, New York, NY, USA, 2008. ACM.

[24] Aaron Clauset. Finding local community structure in networks. Phys. Rev. E, 72(2):026132, Aug 2005.

[25] Aaron Clauset, M. E. J. Newman, and Christopher Moore. Finding community structure in very large networks. Phys. Rev. E, 70, 2004.

[26] Mark Claypool, Phong Le, Makoto Waseda, and David Brown. Implicit interest indicators. In IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces, pages 33–40, New York, NY, USA, 2001. ACM.

[27] CNN. http://money.cnn.com/1999/01/28/technology/yahoo_a/, January 1999. Visiting June 2010.

[28] Luciano da F. Costa, Francisco A. Rodrigues, Gonzalo Travieso, and Villas P. R. Boas. Characterization of complex networks: A survey of measurements. Advances in Physics, 56(1):167–242, Aug 2007.

[29] Stefania Costache, Wolfgang Nejdl, and Raluca Paiu. Personalizing pagerank-based ranking over distributed collections. In CAiSE, pages 111–126, 2007.

[30] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. Phys. Rev. Lett., 94(16):160202, Apr 2005.

[31] Ying Ding, Erjia Yan, Arthur Frazho, and James Caverlee. Pagerank for ranking authors in co-citation networks. Journal of the American Society for Information Science and Technology, 60-11:2229–2243, 2009.

[32] Martin G. Everett and Stephen P. Borgatti. Analyzing clique overlap. Connections, 21(1):49–61, 1998.

[33] Facebook. http://blog.facebook.com/blog.php?post=91242982130, June 2009. Visited June 2010.

[34] Illés Farkas, Dániel Ábel, Gergely Palla, and Tamás Vicsek. Weighted network modules. New J. Phys., 9:180, 2007.

[35] Randy Farmer and Bryce Glass. Building Web Reputation Systems. O'Reilly Media, Inc., 2010.

[36] Gary William Flake, Steve Lawrence, C. Lee Giles, and Frans M. Coetzee. Self-organization and identification of web communities. Computer, 35(3):66–71, 2002.

[37] R. Forsati and M. R. Meybodi. Effective page recommendation algorithms based on distributed learning automata and weighted association rules. Expert Syst. Appl., 37(2):1316–1330, 2010.

[38] Ana L.N. Fred and Anil K. Jain. Robust data clustering. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2:128, 2003.

[39] Linton C. Freeman. A set of measures of centrality based on betweenness. Sociometry, 40(1):35–41, March 1977.

[40] Daniel Gayo-Avello. Nepotistic relationships in twitter and their impact on rank prestige algorithms. CoRR, abs/1004.0816, 2010.

[41] Rumi Ghosh and Kristina Lerman. Predicting influential users in online social networks. CoRR, abs/1005.4882, 2010.

[42] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. PROC.NATL.ACAD.SCI.USA, 99:7821, 2002.

[43] Peter A. Gloor, Jonas Krauss, Stefan Nann, Kai Fischbach, and Detlef Schoder. Web science 2.0: Identifying trends through semantic social network analysis. Computational Science and Engineering, IEEE International Conference on, 4:215–222, 2009.

[44] Manuel Gomez Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. In KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1019–1028, New York, NY, USA, 2010. ACM.

[45] Google. http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html, August 2006. Visited June 2010.

[46] Google. http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html, July 2008. Visited June 2010.

[47] Sonja Grabner-Kräuter, Ewald A. Kaluscha, and Marliese Fladnitzer. Perspectives of online trust and similar constructs: a conceptual clarification. In ICEC '06: Proceedings of the 8th international conference on Electronic commerce, pages 235–243, New York, NY, USA, 2006. ACM.

[48] Weisen Guo and Steven B. Kraines. Inferring trust from recommendations in web-based knowledge sharing systems. In Advances in Intelligent Web Mastering, volume 43/2007, pages 148–153. Springer Berlin / Heidelberg, 2007.

[49] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases, pages 576–587. VLDB Endowment, 2004.

[50] Andreas Herzig, Emiliano Lorini, Jomi F. Hubner, and Laurent Vercouter. A logic of trust and reputation. Logic Jnl IGPL, 18(1):214–244, 2010.

[51] John E. Hopcroft and Daniel Sheldon. Manipulation-resistant reputations using hitting time. Internet Mathematics, 5(1):71–90, 2008.

[52] Bernardo A. Huberman, Daniel M. Romero, and Fang Wu. Social networks that matter - twitter under the microscope. First Monday, 14(1-5), 2009.

[53] Lee Humphreys, Phillipa Gill, and Balachander Krishnamurthy. How much is too muich? privacy issues on twitter. In 60th Conference of the International Communication Association, 2010.

[54] Curtis Huttenhower, Avi I Flamholz, Jessica N Landis, Sauhard Sahi, Chad L Myers, Kellen L Olszewski, Matthew A Hibbs, Nathan O Siemers, Olga G Troyanskaya, and Hilary A Coller. Nearest neighbor networks: clustering expression data based on gene neighborhoods. BMC Bioinformatics, 8:250, 2007.

[55] Khalid Jaber, Nur'Aini Abdul Rashid, and Rosni Abdullah. The parallel maximal cliques algorithm for protein sequence clustering. American Journal of Applied Sciences, 6:1368–1372, 2009.

[56] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. Bulletin del la Société Vaudoise des Sciences Naturelles, 37:547–579, 1901.

[57] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, pages 56–65, New York, NY, USA, 2007. ACM.

[58] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, pages 154–161, New York, NY, USA, 2005. ACM.

[59] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. Decis. Support Syst., 43(2):618–644, 2007.

[60] Daniel Jurafsky and James H. Martin. Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence). Prentice Hall, 2 edition, 2008.

[61] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In WWW '03: Proceedings of the 12th international conference on World Wide Web, pages 640–651, New York, NY, USA, 2003. ACM.

[62] R. Karp. Reducibility among combinatorial problems. Complexity of Computer Computations, pages 85–103, 1972.

[63] W. O. Kermack and Ag McKendrick. A contribution to the mathematical theory of epidemics. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 115(772):700–721, August 1927.

[64] Maksim Kitsak, Lazaros K. Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H. Eugene Stanley, and Hernan A. Makse. Identifying influential spreaders in complex networks. In Proceedings of the International School and Conference on Network Science (NetSci2010), 2010.

[65] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. J. ACM, 46(5):604–632, 1999.

[66] Konstantin Klemm, M Angeles Serrano, Victor M Eguiluz, and Maxi San Miguel. Dynamical influence: how to measure individual contributions to collective dynamics in complex networks. Technical Report arXiv:1002.4042, Feb 2010. Comments: 6 pages, 4 figures, 1 table.

[67] Klout.com. http://klout.com/kscore, 2010. Visited Nov 2010.

[68] Balachander Krishnamurthy, Phillipa Gill, and Martin Arlitt. A few chirps about twitter. In WOSP '08: Proceedings of the first workshop on Online social networks, pages 19–24, New York, NY, USA, 2008. ACM.

[69] Ugur Kuter and Jennifer Golbeck. Sunny: a new algorithm for trust inference in social networks using probabilistic confidence models. In AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence, pages 1377–1382. AAAI Press, 2007.

[70] Joris Lammers, Diederik A. Stapel, and Adam D. Galinsky. Power increases hypocrisy moralizing in reasoning, immorality in behavior. Psychological Science, 21-5:737–744, 2010.

[71] Amy N. Langville and Carl D. Meyer. Google's PageRank and Beyond: The Science of Search Engine Rankings. Princeton University Press, Princeton, NJ, USA, 2006.

[72] Longzhuang Li, Yi Shang, and Wei Zhang. Improvement of hits-based algorithms on web documents. WWW '02: Proceedings of the 11th international conference on World Wide Web, pages 527–535, 2002.

[73] S. Lloyd. Least squares quantization in pcm. Information Theory, IEEE Transactions on, 28(2):129–137, January 2003.

[74] Feng Luo, James Z. Wang, and Eric Promislow. Exploring local community structures in large networks. pages 233–239, 2006.

[75] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281–297. University of California Press, 1967.

[76] Mashable.com. `http://mashable.com/2010/11/17/twitter-reputation-scores/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+Mashable+%28Mashable%29&utm_content=Google+Reader`, Nov 2010. Visited Nov 2010.

[77] Jeff McCord. http://www.jeffmccord.org/reputation-20/, 2008. Visited Nov 2010.

[78] Marcelo Mendoza, Barbara Poblete, and Carlos Castillo. Twitter under crisis: Can we trust what we rt? In 1st Workshop on Social Media Analytics (SOMA '10). ACM Press, July 2010.

[79] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pages 29–42, New York, NY, USA, 2007. ACM.

[80] Alan Mislove, Ansley Post, Peter Druschel, and Krishna P. Gummadi. Ostra: leveraging trust to thwart unwanted communication. In NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pages 15–30, Berkeley, CA, USA, 2008. USENIX Association.

[81] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. You are who you know: inferring user profiles in online social networks. In WSDM '10: Proceedings of the third ACM international conference on Web search and data mining, pages 251–260, New York, NY, USA, 2010. ACM.

[82] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society.

[83] Bonnie A. Nardi, Diane J. Schiano, Michelle Gumbrecht, and Luke Swartz. Why we blog. Commun. ACM, 47(12):41–46, 2004.

[84] M. E. J. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. Phys. Rev. E, 64(1):016132, Jun 2001.

[85] M. E. J. Newman. Assortative mixing in networks. Phys. Rev. Lett., 89(20):208701, Oct 2002.

[86] M. E. J. Newman. Mixing patterns in networks. Phys. Rev. E, 67(2):026126, Feb 2003.

[87] M. E. J. Newman. Fast algorithm for detecting community structure in networks. Phys. Rev. E, 69, 2004.

[88] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. Phys. Rev. E, 69(2):026113, Feb 2004.

[89] David M. Nichols. Implicit rating and filtering. In In Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering, pages 31–36, 1998.

[90] Brendan O'Connor, Michel Krieger, and David Ahn. Tweetmotif: Exploratory search and topic summarization for twitter. In Proceedings of the International AAAI Conference on Weblogs and Social Media, Washington DC, May 2010.

[91] Paul Ormerod and Kristin Glass. Predictability in an unpredictable artificial cultural market. Technical Report arXiv:0908.1320, Aug 2009.

[92] Leysia Palen, Sarah Vieweg, Sophia B. Liu, and Amanda Lee Hughes. Crisis in a networked world. Soc. Sci. Comput. Rev., 27(4):467–480, 2009.

[93] Gergely Palla, Imre Derényi, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 435:814–818, 2005.

[94] Gergely Palla, Illés J Farkas, Péter Pollner, Imre Derényi, and Tamás Vicsek. Directed network modules. New J. Phys., 9:186, 2007.

[95] Wei Pan, Manuel Cebrian, Wen Dong, Taemie Kim, and Alex Pentland. Modeling dynamical influence in human interaction patterns. Oct 2010.

[96] Clara Pizzuti. Overlapped community detection in complex networks. In GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pages 859–866, New York, NY, USA, 2009. ACM.

[97] Péter Pollner, Gergely Palla, Dániel Ábel, Andrés Vicsek, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. Centrality properties of directed module members in social networks. Physica A: Statistical Mechanics and its Applications, 387(19-20):4959 – 4966, 2008.

[98] Krishna P.N. Puttaswamy, Alessandra Sala, and Ben Y. Zhao. Starclique: guaranteeing user privacy in social networks against intersection attacks. In CoNEXT '09: Proceedings of the 5th international conference on Emerging networking experiments and technologies, pages 157–168, New York, NY, USA, 2009. ACM.

[99] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. Proceedings of the National Academy of Sciences of the United States of America, 101(9):2658–2663, March 2004.

[100] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. Journal of Documentation, 60:2004, 2004.

[101] Daniel M. Romero, Wojciech Galuba, Sitaram Asur, and Bernardo A. Huberman. Influence and passivity in social media. Social Science Research Network Working Paper Series, August 2010.

[102] Jordi Sabater and Carles Sierra. Social regret, a reputation model based on social relations. SIGecom Exch., 3(1):44–56, 2002.

[103] M.J. Salganik, P.S. Dodds, and D.J. Watts. Experimental study of inequality and unpredictability in an artificial cultural market. Science, 311(5762):854–856, 2006.

[104] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: news in tweets. In GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 42–51, New York, NY, USA, 2009. ACM.

[105] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In Proceedings of International Conference on Weblogs and Social Media (ICWSM), 2009.

[106] Stephen B. Seidman. Network structure and minimum degree. Social Networks, 5:269–287, 1983.

[107] Wanita Sherchan, Seng W. Loke, and Shonali Krishnaswamy. A fuzzy model for reasoning about reputation in web services. In SAC '06: Proceedings of the 2006 ACM symposium on Applied computing, pages 1886–1892, New York, NY, USA, 2006. ACM.

[108] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28:11–21, 1972.

[109] Kate Starbird, Leysia Palen, Amanda L. Hughes, and Sarah Vieweg. Chatter on the red: what hazards threat reveals about the social life of microblogged information. In CSCW '10: Proceedings of the 2010 ACM conference on Computer supported cooperative work, pages 241–250, New York, NY, USA, 2010. ACM.

[110] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In Proceedings of the 17th National Conference on Artificial Intel ligence: Workshop of Artificial Intelligence for Web Search (AAAI 2000), 30-31 July 2000, Austin, Texas, USA, pages 58–64. AAAI, July 2000.

[111] LA Times. http://latimesblogs.latimes.com/technology/2009/10/geocities-closing.html, October 2009. Visited June 2010.

[112] Riitta Toivonen, Lauri Kovanen, Mikko Kivelä, Jukka-Pekka Onne la, Jari Saramäki, and Kimmo Kaski. A comparative study of social network models: Network evolution models and nodal attribute models. Social Networks, 31(4):240–254, October 2009.

[113] Daniel Tunkelang. http://thenoisychannel.com/2009/01/13/a-twitter-analog-to-pagerank/, January 2009. Visited June 2010.

[114] Twitter.com. http://dev.twitter.com/pages/rate-limiting, 2010. Visited November 2010.

[115] Twitter.com. http://dev.twitter.com/pages/responses_errors, 2010. Visited November 2010.

[116] Twitter.com. http://dev.twitter.com/pages/streaming_api_methods, 2010. Visited November 2010.

[117] Twitter.com. http://engineering.twitter.com/2010/10/twitters-new-search-architecture.html, October 2010. Visited November 2010.

[118] Twitter.com. http://support.twitter.com/entries/13920-frequently-asked-questions, 2010. Visited November 2010.

[119] Stijn Van Dongen. Graph clustering via a discrete uncoupling process. SIAM Journal on Matrix Analysis and Applications, 30(1):121–141, 2008.

[120] Quang Hieu Vu, Mihai Lupu, and Beng Chin Ooi. Trust and reputation. In Peer-to-Peer Computing, pages 183–214. Springer Berlin Heidelberg, 2010.

[121] S. Wasserman and K. Faust. Social network analysis: Methods and applications. Cambridge Univ Pr, 1994.

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

[123] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In WSDM '10: Proceedings of the third ACM international conference on Web search and data mining, pages 261–270, New York, NY, USA, 2010. ACM.

[124] Sebastian Wernicke and Florian Rasche. Fanmod: a tool for fast network motif detection. Bioinformatics, 22(9):1152–1153, 2006.

[125] Ryen W. White, Joemon M. Jose, and Ian Ruthven. An implicit feedback approach for interactive information retrieval. In Information Processing and Management: an International Journal, volume 42, pages 166–190, Tarrytown, NY, USA, 2006. Pergamon Press, Inc.

[126] D. M. Wilkinson and B. A. Huberman. A method for finding communities of related genes. Proceedings of the National Academy of Sciences of the United Sta tes of America, 101(Suppl 1):5241–5248, April 2004.

[127] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna P.N. Puttaswamy, and Ben Y. Zhao. User interactions in social networks and their implications. Proceedings of the 4th ACM European conference on Computer systems, pages 205–218, 2009.

[128] Haifeng Yu, Michael Kaminsky, Phillip B. Gibbons, and Abraham Flaxman. Sybilguard: defending against sybil attacks via social networks. SIGCOMM Comput. Commun. Rev., 36(4):267–278, 2006.

[129] Cai-Nicolas Ziegler and Georg Lausen. Analyzing correlation between trust and user similarity in online communities. In Proceedings of Second International Conference on Trust Management, pages 251–265. Springer-Verlag, 2004.

[130] David Zuckerman. On unapproximable versions of np-complete problems. SIAM J. Comput., 25(6):1293–1304, 1996.