Structural Analysis of Large Networks: Observations and Applications

Mary McGlohon

December 2010
CMU-ML-10-111

Carnegie Mellon

Structural Analysis of Large Networks: Observations and Applications

Mary McGlohon

December 2010
CMU-ML-10-111

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christos Faloutsos, Co-chair
Alan Montgomery, Co-chair
Geoffrey Gordon
David Jensen, University of Massachusetts

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by Lawrence Livermore National Laboratory under grant number B580840, National Science Foundation under grant numbers IIS-0808661, DBI-0640543, DGE-0234630, DGE-0750271, and IIS-0534205, Lehigh University under grant number C000022761, and Google. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Social networks, data mining, network diffusion, anomaly detection

Dedicated to my father, who nurtured the inquisitiveness to begin this work,
and inspired the perseverance to see it to completion.

Abstract

Network data (also referred to as relational data, social network data, real graph data) has become ubiquitous, and understanding patterns in this data has become an important research problem. We investigate how interactions in social networks are formed and how these interactions facilitate diffusion, model these behaviors, and apply these findings to real-world problems.

We examined graphs of size up to 16 million nodes, across many domains from academic citation networks, to campaign contributions and actor-movie networks. We also performed several case studies in online social networks such as blogs and message board communities.

Our major contributions are the following: (a) We discover several surprising patterns in network topology and interactions, such as Popularity Decay power law (in-links to a blog post decay with a power law with -1.5 exponent) and the oscillating size of connected components; (b) We propose generators such as the Butterfly generator that reproduce both established and new properties found in real networks; (c) several case studies, including a proposed method of detecting misstatements in accounting data, where using network effects gave a significant boost in detection accuracy.

Acknowledgments

I am indebted to many people for this document's very completion, and to still more for making the journey a uniquely rewarding one.

I first thank my committee: Christos Faloutsos has been a constant source of expertise and exhortation. Alan Montgomery has patiently mentored me in marketing concepts, helped me pick out promising and important problems, and provided many exciting discussions. Geoff Gordon, who I firmly believe can teach anyone anything, has been a great help to me throughout my years in the Ph.D. program. David Jensen has encouraged me to look at things from a "big ideas" perspective, which has helped prevent "thesis tunnel vision" and reminded me why I like this field.

I thank my co-authors who were also directly responsible for this work. I have been fortunate to work with some of the best: Leman Akoglu, Markus Anderle, Stephen Bay, Natalie Glance, Mila Goetz, Matthew Hurst, U Kang, Jure Leskovec, Ravi Kumar, Mohammad Mahdian, Zach Reiter, and David Steier. Many of these people also hosted me on internships at PricewaterhouseCoopers, Microsoft, and Google. These experiences were highly influential in the production of this work and the trajectory of my career, serving to both ground my research in real applications and to inspire creativity in problem solving.

I am indebted to many other colleagues in Machine Learning, SCS, and the CS "extended family". They have played a less obvious but critical role in this work through helpful discussions, feedback on practice talks and papers, collaboration in coursework and other Ph.D. pursuits, and administrative and moral support. Some of these include: Andrew Arnold, Nels Beckman, Joseph Bradley, Alice Brumley, David Brumley, Andy Carlson, Sharon Cavlovich, Lucia Castellanos, Polo Chau, William Cohen, Catherine Copetas, Dec/5, Mike Dinitz, Khalid El-Arini, Steve Fienberg, Stano Funiak, Charlie Garrod, Joey Gonzales, Fan Guo, Sue Ann Hong, Jon Huang, Kevin Killourhy, Andreas Krause, Stacey Kuznetsov, Thomas LaToza, Gabe Levi, Lei Li, Austin McDonald, Brendan Meeder, Jamie Morgenstern, Andrew Moore, Chris Neff, Abe Othman, Purna Sarkar, Charles Schafer, Julia Schwarz, Cosma Shalizi, Sajid Siddiqi, SIGBOVIK, Rob Simmons, Diane Stidle, Michael Tschantz, Hanghang Tong, Gaurav Veda, Marilyn Walgora, and Charlotte Yano.

Before coming to CMU, I was introduced to the joys of research at the University of Tulsa. I am grateful to all my friends there. Particularly influential in my early pursuits as a scientist was my colleague and friend Paul Crider, who challenged me to push my own boundaries in thought and in action. Additionally, my research advisor Christian Constanda patiently guided me through my first summer research project. My research advisor Sandip Sen inspired me to take on new concepts and irritatingly answered any hesitation I voiced with "Why not?". Mentors Donna Farrior and Nona Charleston also helped me in my endeavors (while periodically reminding me that there are, in fact, important things in life besides work).

I thank my family: my mother Debbie, who has always encouraged me through the rough times; my father Dwight, who first taught me to think like a scientist; and my brothers Alan and Neil, who have been true role models in living on one's own terms and pursuing one's dreams with both fearlessness and a sense of humor.

Most of all, I thank my husband Austin McDonald, for believing in me, for stubbornly refusing to let me give up on this Ph.D. thing, and for being a source of joy in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Impact

Our main research goal is *to understand how social networks form and evolve over time*. Network (a.k.a. relational, graph) data have become ubiquitous and accessible, in domains such as online social networks, citation networks, and political campaign contributions. Understanding how these networks form and evolve is a critical data mining problem, with applications in fields such as sociology, marketing, security, and human-computer interaction. Our interests focus on three interrelated topics: the global *topology* of networks, *diffusion* within a network, and *applications* of network data.

We will refer to this kind of data as "network data," but note potential confusion in terms. We do not limit our study only to networks of computers, but to many other kinds of networks. We will use the terms "network," "social network," and "real graph" interchangeably.

The first goal is to to observe patterns in the *topological structure* of these networks. How do new nodes and links form in a network? Are these patterns common to all networks, or only those in certain domains? These observations provide intuition about the mechanisms driving network evolution, allow us to forecast future behavior, and help us spot anomalies.

In addition to knowing how networks form, it is important to understand the dynamics of *diffusion* inside a network. How do rumors or viruses spread through a network of people or computers? Do certain structural conditions allow for different patterns of propagation? One way to address these questions is to study conversation trees, or *cascades*, and the typical patterns they display.

Finally, we plan to use knowledge of networks and their behavior to build useful *applications*. For instance, can we use ideas about propagation to spot certain types of communities (such as blog communities or auction fraudsters) or forecast product adoption? Can we detect fraud in a cellular phone network by identifying deviations from typical call patterns? In online social networks, do certain structural patterns allow for a sub-group to thrive or cause it to dissolve? If we know what typical diffusion patterns are, how can we optimize marketing plans to target certain users and take advantage of network effects? These are only a few of the many areas that may be aided by a better understanding of networks.

1.2 Overview and Thesis Statement

This leads us to the thesis statement.

We investigate surprising patterns in the structure of network formation (topology) and network interactions (cascades), and create realistic generators to help explain these behaviors. We also apply these findings to real-world problems such as anomaly detection.

This work is divided into three parts. The first part, **Topology of Networks**, will examine the *global topology* of networks. This is typically what is thought of as a "social network": each node represents an individual person, machine, or other entity. The problems we address are:

- Given a collection of several different types of networks (political campaign contributions, computer network traffic, links in online social networks), can we identify patterns in the structure and formation of these networks? (Chapter 3)
- Given a knowledge of several established patterns (heavy-tailed degree distribution, shrinking diameter, densification), as well as newly discovered ones (small non-giant components, power laws in edge weights) can we propose intuitive models that will generate this behavior? (Chapter 4)

We are the first (or among the first) to discover patterns in weighted graphs and in secondary connected components. We discover several interesting patterns in weighted graphs, such as the *fortification law* explaining a power law relationship between the edges and weights in a network (Section 3.2). We show that the sizes of secondary components appear to oscillate (Section 3.1). We also contribute two generators: the agent-based Butterfly model, which produces emergent behavior; and the Recursive Tensor model, an easy-to-analyze generator which exploits self-similar behavior to produce realistic networks.

In the second part we take our observation to a finer scale, analyzing diffusion between nodes in networks. We perform case studies on two different types of online social networks. The first type is *blog-like*, and includes a collection of traditional blogs, as well as a collection crawled from Twitter, a micro-blogging domain. The second type of online social network is *group-like*, and consists of both Usenet and Yahoo! Groups, where members essentially join a mailing list or message board. Every member can easily access all posts in the community by going to a centralized location. The blog domain, on the other hand, is more decentralized, and boundaries between blogging communities are difficult to establish. Because of this key difference we will analyze these separately. We study cascades, or graph formed by conversations where each node is a posted object (rather than a person) and an edge indicates a reply from one to the other. The problems addressed in that part are:

- Given a set of interactions in an online environment—from blogs, message boards, or other media—what structural and temporal patterns can we identify in these interactions? (Chapter 6)
- Given a knowledge of several temporal and cascade patterns in online social networks, (power law in in-links, cascade shapes and sizes, etc.) can we propose intuitive models that will generate this behavior? (Chapter 7)

We find several interesting and surprising patterns, such as the Popularity Decay power law and Stars and Chains power laws (Section 6.1). We also propose generators for both blog behavior (Cascade Generation (Section 7.2.1) and Zero-crossing (Section 7.2.2) and groups behavior

	Observations	Models and Tools
Part I: Network Topology	Chapter 3: Patterns in connected components and weighted graphs [6, 153]	Chapter 4: "Butterfly" model [153], Recursive Tensor Model [6]
Part II: Network Diffusion	Chapter 6: Blog studies [140, 152], Online groups studies [131, 150, 151].	Chapter 7: Cascade Generation Model [140] for blogs, Zero Crossing Model [87] for blogs, TI-model for online groups [131]
Part III: Network Effects In Action	Chapter 8: Oddball: Anomaly detection in graphs [8] Chapter 9: SNARE: belief propagation for clustering and risk detection [154] Chapter 10: Star Quality: Analyzing online reviews [155]	

Table 1.1: Outline of thesis work.

(TI-model) (Section 7.1.2).

Finally, in the third part we will address a few challenges in analyzing network data. The first is anomaly detection in networks. We propose a framework, *Oddball*, which uses known properties common to networks to find anomalous nodes. The second is the application of risk detection in accounting data, which models accounting data as a network and uses diffusion of established risk, assuming "guilt by association," to pinpoint the top candidates for investigation of fraud. Thirdly, we will address the problem of ranking in a data set of online reviews (a bipartite network between authors and objects). Thus, the problems in this part include:

- Given a knowledge of patterns in real networks, how can we identify anomalous nodes in a tractable, accurate, and explainable manner? (Chapter 8)
- Given a network of interactions, and some (noisy) knowledge of the labels of a few of the nodes, can we label other nodes in the network? Can we apply this to a real problem of risk detection in accounting data? (Chapter 9)
- Given a set of online reviews aggregated from a variety of sources, how do we provide a reliable ranking of the rated objects? (Chapter 10)

Our "oddball" method in Chapter 8 finds several interesting structural patterns, and proves useful for detecting outliers in networks of blogs and campaign contributions. For the problem of misstatement detection, our method *SNARE* produces a lift of 6.5 in true positives over the heuristic baseline.

An outline of the thesis is shown in Table 1.2, along with citations to published work. A list of publications may also be found in Appendix C.

Part I

Topology and formation of networks

What is the structure of networks on a global scale? How do they evolve over time? How do the different components of a network form? What happens when we take into account multiple edges and weighted edges? What patterns do the weights obey? Do they follow a Gaussian distribution, for a given snapshot in time? How, if at all, is an edge weight related to the degree of its adjacent nodes? Do these patterns persist over time?

There has been extensive work focusing on static static snapshots of graphs, where fascinating properties have been discovered, the most striking ones being the 'small-world' phenomenon [208] (also known as 'six degrees of separation' [157]) and the power-law degree distributions [17, 70]. Time-evolving graphs have attracted attention only recently, where even more fascinating properties have been discovered, like *shrinking* diameters, and the so-called *densification power law* [136]. Moreover, we find interesting properties in terms of *multiple* edges between nodes, or edge weights.

In the next few chapters we will describe some of the most important properties apparent in networks, with a particular emphasis on dynamic properties, and some of the newer findings with respect to edge weights. Most previous work has focused on the 'giant connected component' (GCC), either explicitly or implicitly, and moreover it ignored multiple links between nodes or weights on edges. Taking into consideration data collection issues (discussed in 2.3.2), we will shift our focus to the components that are of moderate size but "disconnected" from the GCC of the undirected graph. We refer to the largest of these as the "next-largest connected components" (NLCCs), and in general the group of components that are not included in the GCC are referred to as "disconnected components" or "non-giant components". We will also look at edge weights, particularly at how weighted (or multi) edges are added over time. We will also propose new graph generators that mimic behavior known to occur in real networks, and prove some of their properties.

The questions of interest are:

- *How do networks behave over time?* Does the structure vary as the network grows? In what fashion do new entities enter a network? Does the network retain certain graph properties as it grows and evolves? Does the graph undergo a "phase transition" in which its behavior suddenly changes?
- *How do the non-giant connected components behave over time?* One might argue that they grow as new nodes are being added, and their size would probably remain a fixed fraction of the size of the GCC. Someone else might counter-argue that they shrink, as they are absorbed into the GCC. What is happening, in real graphs?
- *What distributions and patterns do weighted graphs maintain?* How does the distribution of weights change over time? Do we also observe a densification of weights as well as single-edges? How does the distribution of weights relate to the degree distribution? Is the addition of weight bursty over time, or is it uniform?
- *Can we produce generators that will mimic the above behaviors?* The *preferential attachment* model generates a single connected component. Most other generators try to mimic the heavy-tailed in- and out-degree distributions and suffer from the same issue. Our goal is to find a generator that mimics skewed degree distribution in unweighted graphs, as well as producing realistic behavior of non-giant connected components and producing other newly discovered patterns.

Answering these questions is important to understand how natural graphs evolve, and to spot anomalous graphs and sub-graphs, to answer questions about entities in a network and forecast various scenarios, and to discard unrealistic graph generators. Spotting anomalies is vital for determining abuse of social and computer networks, such as link-spamming in a web graph, fraudulent reputation building in e-auction systems [176], detection of dwindling/abnormal social sub-groups in a social-networking site like Yahoo-360 (`360.yahoo.com`), Facebook (`www.facebook.com`), or LinkedIn (`www.linkedin.com`), and network intrusion detection [132]. Analyzing network properties is also useful for identifying authorities and search algorithms [39, 47, 130], for discovering the "network value" of customers for using viral marketing [185], and for improving recommendation systems [26]. Simulating various scenarios is vital for extrapolation, provisioning and algorithm design. For example, if we expect that the number of links will double within the next year, we should provision for the appropriate hardware to store and process the upcoming queries. Finally, rules like the upcoming ones can help us eliminate unrealistic graph generators. Graph generators are also vital for simulation of algorithms (like computer network routing algorithms), for simulation of rumor (or virus, or influence) propagation, and many other settings. In several such settings, real graphs may be difficult or even impossible to collect: for example a who-believes-whom graph is only in the mind of the human subjects; a who-mails-whom graph may be protected by privacy laws.

In Chapter 3, we will examine both static and dynamic properties of weighted and unweighted graphs. We will describe previously-discovered patterns as well as contribute new ones. We will also, in Chapter 4 propose two different generators for producing known properties: one, the *Butterfly* generator is agent-based and formed using local decisions, while the *Recursive Tensor Model* (RTM) is formed through self-similarity and has several additional provable properties.

Before delving into new contributions, however, we in Chapter 2 establish terms and definitions, survey related work for patterns and models in network topology, and provide an introduction to the data under analysis.

Chapter 2

Preliminaries

In this chapter we will introduce concepts used for studying patterns in networks: graph basics, tensors, and heavy-tailed distributions. We will follow that introduction with a survey of related work in patterns and models of network topology, and then introduce the data sets we will use as case studies.

2.1 Definitions

Graph definitions: Networks as graphs

A network is typically represented as a *graph*. Because of this, networks are often referred to as *real graphs* in the literature.

A static, unweighted graph G consists of a set of nodes V and a set of edges $E \subset V \times V$, as $G = (V, E)$. We denote the sizes of V and E as $|V|$ and $|E|$. A graph may be *directed* or *undirected*: for instance, a phone call may be from one party to another, and will have a directed edge, or a mutual friendship may be represented as an undirected edge. Most properties we examine will be on undirected graphs.

Graphs may also be *weighted*, where there may be multiple edges occurring between two

Symbol	Description
$\mathcal{G}$	Graph representation of datasets
$\mathcal{V}$	Set of nodes for graph $\mathcal{G}$
$\mathcal{E}$	Set of edges for graph $\mathcal{G}$
N	Number of nodes, or $\|\mathcal{V}\|$
E	Number of edges, or $\|\mathcal{E}\|$
$e_{i,j}$	Edge between node i and node j
$w_{i,j}$	Weight on edge $e_{i,j}$
w_i	Weight of node i (sum of weights of incident edges)
$\mathbf{A}$	0-1 Adjacency matrix of the unweighted graph
$\mathbf{A}_w$	Real-value adjacency matrix of the weighted graph
$a_{i,j}$	Entry in matrix $\mathbf{A}$
λ_1	Principal eigenvalue of unweighted graph
$\lambda_{1,w}$	Principal eigenvalue of weighted graph

Table 2.1: Table of symbols used in notation.

nodes (e.g. repeated phone calls) or specific edge weights (e.g. monetary amounts for transactions). In a weighted graph $\mathcal{G}$, let $e_{i,j}$ be the edge between node i and node j. We shall refer to these two nodes as the *'neighboring nodes'* or *'incident nodes'* of edge $e_{i,j}$. Let $w_{i,j}$ be the weight on edge $e_{i,j}$. The *total weight* w_i of node i is defined as the sum of weights of all its incident edges, that is $w_i = \sum_{k=1}^{d_i} w_{i,k}$, where d_i denotes its *degree*, or number of unique neighbors. As we show later, there is an empirical relation between a given edge weight $w_{i,j}$ and the weights of its neighboring nodes w_i and w_j.

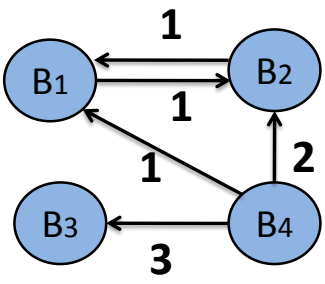
Graphs may be *unipartite* or *multipartite*. One usually considers social networks to be unipartite (people in a group, papers in a citation network, etc.). However, they may also be multipartite: that is, there are multiple classes of nodes and edges are only drawn between nodes of different classes. Bipartite graphs, like the movie-actor graph of IMDb, consist of disjoint sets of nodes V_1 and V_2, say, for authors and movies, with no edges between nodes of the same type.

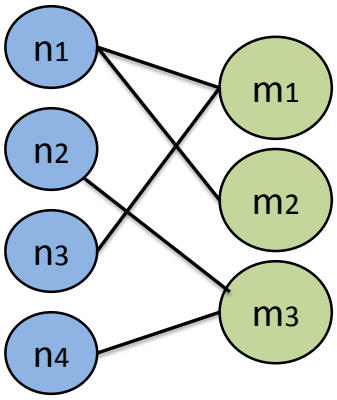
We also represent a graph as an *adjacency matrix* $\mathbf{A}$, where nodes are in rows and columns, and numbers in the matrix indicate the existence of edges. For unweighted graphs, all entries are 0 or 1; for weighted graphs the adjacency matrix contains the values of the weights. Another useful measure is the *eigenvalues* of a graph, which are defined as the eigenvalues of the adjacency matrix.[1]

For a given (static) graph, its *diameter* is defined as the maximum *distance* between any two nodes, where distance is the minimum number of hops (i.e., edges that must be traversed) on the path from one node to another. Intuitively, the diameter represents how much of a "small world" the graph is—how quickly one can get from one "end" of the graph to another. (Therefore, ignoring directionality and weights in the calculation of diameter will make more intuitive sense for the kinds of questions we will ask.) Calculating graph diameter is $\Omega(N^3)$. Therefore, we choose to estimate the graph diameter by sampling nodes from the giant component. For $s = \{1, 2, ..., S\}$, we choose two nodes at random and calculate the distance (using breadth-first search). We then choose to record the 90 percentile value of distances, so we take the $.9S$ largest recorded value, or the *effective diameter* [198]. Following earlier literature, we choose to use effective diameter not only because it is faster to calculate, but also because it is robust; degenerate structures such as long chains in a graph can have a large effect on the calculation of full diameter. Furthermore, experiments by Leskovec showed that effective diameter behaves quantitatively similar to full diameter [134]. We use sampling to estimate the diameter; alternative methods would include ANF [175]. We may observe the *diameter-plot* of the graph, that is, its diameter, over time, to answer some questions about the structure of networks.

Another interesting property of a graph is its *component distribution*. We refer to a *connected component* in a graph as a set of nodes and edges where there exists a path between any two nodes in the set. (For directed graphs, this would be a *weakly connected component*, where a *strongly connected component* requires a directed path between any given two nodes in a set.) We find that in real graphs over time, a giant connected component (GCC) forms. However, it is also of interest to study the smaller components—when do they join the GCC, and what size do they reach before doing so? We will seek to answer these questions in the next chapter.

[1] In the context of random walks, the eigenvalues of the graph may also be defined as the eigenvalues of the transition matrix of its associated random walk; these two concepts are closely related, since normalizing the (0/1) adjacency matrix results in the transition matrix





	B1	B2	B3	B4
B1	0	1	0	0
B2	1	0	0	0
B3	0	0	0	0
B4	1	2	3	0

	n1	n2	n3	n4
m1	1	0	1	0
m2	1	0	0	0
m3	0	1	0	1

Figure 2.1: Illustrations of example graphs. On the left is a unipartite, directed, weighted graph and the corresponding adjacency matrix. On the right is an undirected, bipartite graph and the corresponding adjacency matrix.

Tensor definitions: Networks as tensors

As we have shown, we can represent a network as an $(N \times N)$ adjacency matrix. It is also possible to represent a network over time by adding a third mode to the matrix for time, making it a *tensor* with three modes, and dimensions $(N \times N \times \tau)$. A nonzero entry in the matrix (i, j, t) indicates an edge from node i to node j at time t. As in an adjacency matrix, the entries may be 0/1 for unweighted or integers or real numbers for weights. We will use several concepts of tensors in Chapter 4 for our Recursive Tensor Model.

Heavy-tailed Distributions

It is well-established that the degree distributions of most networks are *heavy-tailed* or *skewed distributions*.

While the Gaussian distribution is commonly observed in natural phenomena, there are many cases where the probability of events far to the right of the mean is significantly higher than in Gaussians. In the Internet, for example, most routers have a very low degree (perhaps "home" routers), while a few routers have extremely high degree (perhaps the "core" routers of the Internet backbone) [70] Heavy-tailed distributions attempt to model this. They are known as "heavy-tailed" because, while traditional exponential distributions have bounded variance (large deviations from the mean become nearly impossible), $p(x)$ decays polynomially quickly instead of exponentially as $x \to \infty$, creating a "fat tail" for extreme values on the PDF plot.

One of the more well-known heavy-tailed distributions is the power law distribution. Two variables x and y are related by a power law when:

$$y(x) = Ax^{\alpha} \tag{2.1}$$

where A is a positive constant and α is a negative constant. The constant α is often called the power law exponent. If $y(x)$ is a probability density function, $\alpha < 1$ [169].

Heavy-tailed distributions, such as power laws, occur very often in real-world graphs, as we will discuss. Figures 2.1(a) and 2.1(b) show two examples of power laws.

While power laws appear in a large number of graphs, deviations from a pure power law are sometimes observed. Two of the more common deviations are exponential cutoffs and lognormals.

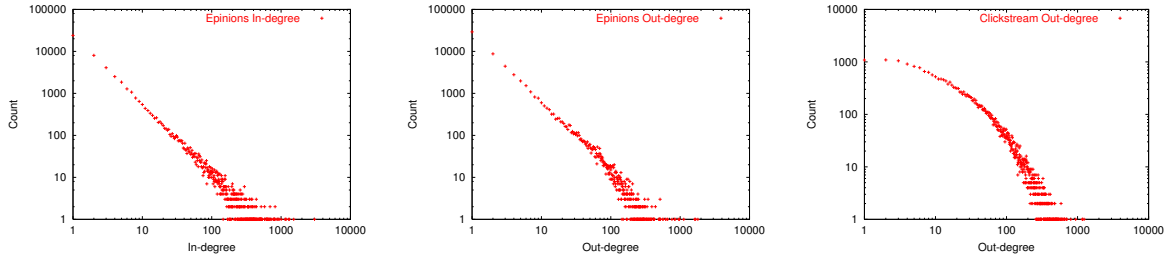
Sometimes, the distribution looks like a power law over the lower range of values along the x-axis, but decays very fast for higher values. Often, this decay is exponential, and this is usually called an exponential cutoff:

$$y(x = k) \propto e^{-k/\kappa} k^{\alpha} \tag{2.2}$$

where $e^{-k/\kappa}$ is the exponential cutoff term and k^{α} is the power law term.

Another common skewed distribution is the lognormal distribution. A lognormal is a distribution whose logarithm is a Gaussian distribution; its PDF looks like a truncated parabola in log-log scales. The equation for the PDF is:

$$f_X(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(lnx-\mu)^2}{2\sigma^2}}, x > 0 \tag{2.3}$$



(a) Epinions In-degree (b) Epinions Out-degree (c) Clickstream Out-degree

Figure 2.2: *Power laws and deviations:* Plots (a) and (b) show the in-degree and out-degree distributions on a log-log scale for the *Epinions* graph (an online social network of $75,888$ people and $508,960$ edges [64]). Both follow power-laws. In contrast, plot (c) shows the out-degree distribution of a *Clickstream* graph (a bipartite graph of users and the websites they surf [162]), which deviates from the power-law pattern.

Similar distributions were studied by Bi et al. [30], who found that a discrete truncated log-normal (called the Discrete Gaussian Exponential or "DGX" by the authors) gives a very good fit. The DGX distribution has been used to fit the degree distribution of a bipartite "clickstream" graph linking websites and users (Figure 2.1(c)), telecommunications and other data.

Methods for fitting skewed distributions are described in [55, 169].

Burstiness and Entropy Plots

Another common trait we find in networks is *burstiness*. Human activity, including weight additions in graphs, is often bursty. Among the many measures for non-uniformity, we propose to use the entropy [191]. Recall that **entropy** on a random variable $\mathcal{X}$, (e.g., the outcome of a random dice) is defined as

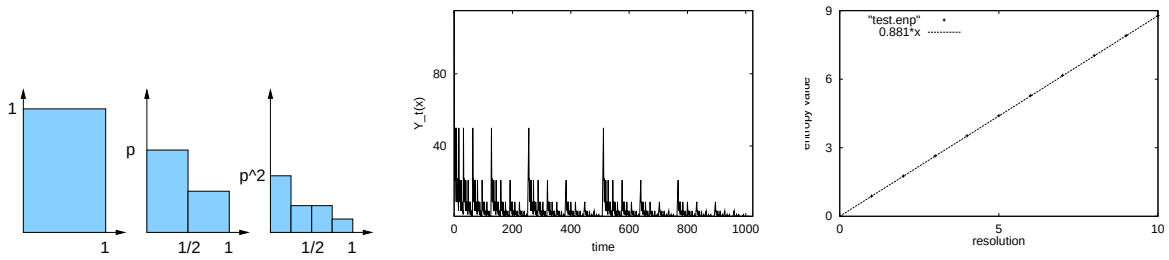
$$H(\mathcal{X}) = -\sum_{i=1}^{N} p_i \log_2 p_i, \tag{2.4}$$

where p_i is the probability of each outcome ($1/6$ in the case of a dice) and N is the total number of possible outcomes (e.g. N=6, for the dice). $H(\mathcal{X})$ is close to 0 if the distribution is highly skewed while a uniform distribution gives the maximum value of $\log_2 N$ for H.

We propose to measure the non-uniformity of a time sequence like the number of weight-additions (e.g. nonunique edges added to a graph over time) $W(t)$ ($t = 1, \cdots, T$) as follows. Let $W_{total} = \sum_{t=1}^{T} W(t)$ be the total weight added to the graph, and let $p(t) = W(t)/W_{total}$ be the percentage of weight added at time t. Then we use the entropy H_p of the time sequence $p(t)$ as a measure of non-uniformity:

$$H_p = -\sum_{t=1}^{T} p(t) log_2(p(t)) \tag{2.5}$$

Thus, if the $p(t)$ activity is uniform over time, the value of its entropy H_p will be maximum. By looking at the (bursty) time-plots of Figure 6.3(a-f) we expect that the entropies will be much



(a) generation (b) synthetic data (c) entropy plot

Figure 2.3: Illustration of the *b-model*: *(*a*)* the recursive 80-20 procedure in its first three iterations *(*b*)* the generated synthetic activity *(e.g., number of posts, over time)* (c) its *entropy plot (*entropy versus resolution - see text*)* Because the synthetic input traffic is self-similar, the entropy plot is linear, that is, *scale free*. Its slope is 0.881, much different than 1.0, which would be the uniform distribution *(*50-50*)*

lower than the entropy maximum. It turns out that we have an even stronger way to characterize our traffic, because it is *self-similar*: if we focus on a smaller sub-sequence, it will be statistically similar to the longer, mother-sequence it came from. This is a variation of the "80-20 law," where 80% of the effects come from 20% of the causes (for example, 80% of the wealth is controlled by 20% of the population). Here, any window of the sequence will have, for example, 80% of the mass on one side and 20% on the other. (Under the traditional interpretation, this would be called an "80-50 law".)

The "b"-model: 80-20 recursively. How would such self similarity appear? It turns out that the recursive application of an 80-20 "law" results in such bursty *and* self-similar behavior. The "b"-model with *bias parameter* b generates activity as follows ($0.5 \leq b \leq 1.0$): if the total activity is, say, P total number of posts during the full interval of observation, and b=0.8 (80-20 law), the first half of the time interval receives b=80% fraction of the activity, and the second half receives the remaining 20%; the first quarter recursively receives 80% of the first half activity, and so on. That is, every sub-interval has exactly the same un-balance like its parent (and uncle, and grand-parent) interval. Figure 2.3(a) illustrates the first few steps of the recursive generation of such bursty traffic. Figure 2.3(b) plots the generated traffic, with bias factor b=0.8, after 2^{10} subdivisions. Notice how bursty the generated traffic is. Of course, we don't have to always favor the left sub-interval: we could occasionally flip our bias, to generate more natural-looking traffic.

Measuring the burstiness: the entropy plot.

There are two questions: (a) How accurately does the *b-model* characterize our blog activities? and (b) How do we measure the bias factor b, when we are given real traffic (e.g., number of edges $E(t)$ added to a network each day).

The answer comes from the theory of fractals and disk traffic modeling, where the *entropy plot* [204] has been used successfully. Again, let's focus on the number of edges over time $E(t)$. The idea is to compute the entropy H_E at the original resolution (1 day), as well as at coarser resolutions (sum of windows of size 2, 4, 8 days and so on). The way the entropy changes with

the resolution answers both questions.

For simplicity, suppose that the number of days T is a power of 2: $T = 2^r$. If not, we can zero-pad the sequence, or clip it to the highest power of 2. Let r stand for the original *resolution*, and let $H_p(r')$ denote the entropy at resolution r' ($0 \leq r' \leq r$). The sequence at resolution r is the original sequence, with duration $T = 2^r$; at resolution $r-1$, the sequence is the sum of successive, disjoint windows of size 2, with duration $T/2$. In general, at resolution $r-i$, we divide the original sequence into disjoint windows of size 2^i, sum them, and compute the entropy $H_p(r-i)$.

Clearly, for resolution 0, the whole sequence collapses to one number, the sum of the sequence, and its entropy is zero (the entropy of an unfair coin that always shows "Heads").

Figure 2.3(c) gives an example. The horizontal axis is the *resolution* r' (0 for the whole interval, 1 for two halves, e.t.c.) and the vertical axis is the entropy $H_p(r')$ of the activity, as described above.

As discussed in [204], traffic generated by the b-model is self-similar, and its entropy plot is linear. Surprisingly, many of the blogs we examined showed activity that *also* resulted in linear entropy plots, in all features we tried: number of posts per day, number of in-links per day, etc., as shown in Figure 6.3 (g,h,i).

To estimate the bias parameter b, we have the following Lemma:

Lemma 1 *For traffic generated by a b-model, the slope s of the entropy plot and the bias factor b obey the equation*

$$s = -b\log_2 b - (1-b)\log_2(1-b)$$

Proof: See [204].

We can use the value s to measure the "burstiness" of a sequence. Notice that bias b=0.5 corresponds to the uniform distribution (fifty-fifty splits for each sub-interval, and slope s=1 for the entropy plot). In the extreme case of b=1.0, all the activity is zero everywhere, except for a burst at one single day, and the slope s=0 for the entropy plot. Therefore, lower values of s are more bursty.[2]

2.2 Related Work

In the next chapters we contribute several newly discovered patterns in networks, and propose generative models to help account for these patterns. To help motivate these studies, we first survey previously discovered patterns and previously proposed generative models.

2.2.1 Previously Discovered Patterns

Patterns in real graphs can be both static (describing what a graph looks like at a single point in time) or dynamic (describing what happens to a graph over time), and weighted or unweighted. We first present the 'laws' that apply to static snapshots of real graphs without considering the weights on the edges. Those include the patterns in degree distributions, the number of hops

[2] s is also called the "information fractal dimension" and estimates the intrinsic dimensionality of a cloud of points.

pairs of nodes can reach each other, local number of triangles, eigenvalues and communities. Then, we describe the dynamic patterns, observations of how the graph evolves over time.

Community Structure

Real-world graphs are found to exhibit a modular structure, with nodes forming groups, and possibly groups within groups [75, 83, 189]. This is often referred to as the "small world property." In a modular graph, the nodes form communities where groups of nodes in the same community are tighter connected to each other than to those nodes outside the community. In [170], Newman and Girvan provide a quantitative measure for such a structure, called *modularity*.

Heavy-tailed Degree Distribution

The degree distribution of many real graphs obey a power law of the form $f(d) \propto d^{-\alpha}$, with the exponent $\alpha > 0$, and $f(d)$ being the fraction of nodes with degree d. Such power-law relations as well as many more have been reported in [46, 70, 124, 169]. Intuitively, power-law-like distributions for degrees state that there exist many low degree nodes, whereas there are only a few high degree nodes in real graphs.

Several datasets have shown deviations from a pure power law [11, 30, 160, 183]: examples include the electric power-grid of Southern California, the network of airports, several topic-based subsets of the WWW, Web "clickstream" data, sales data in retail chains, file size distributions, and phone usage data.

Small Diameter

One of the most striking patterns that real-world graphs have is a small diameter, which is also known as 'six degrees of separation'.

For a given static graph, its diameter is defined as the maximum *distance* between any two nodes, where distance is the minimum number of hops (i.e., edges that must be traversed) on the path from one node to another, usually ignoring directionality. Intuitively, the diameter represents how much of a "small world" the graph is: how quickly one can get from one "end" of the graph to another.

Many real graphs were found to exhibit surprisingly small diameters (as established historically [10, 18, 157]).

Triangle Power Law

A network will have a large number of triangles: groups of three nodes connected to each other. However, it is also worthy to note how nodes participate in these triangles. Some nodes will be a member of many triadic groups and others will not. According to work in [199], the number of triangles that a given node participates in, recorded for each node in the network, follows a power-law distribution. That is, the number $f(\Delta)$ of nodes that participate in Δ triangles obeys $f(\Delta) \propto \Delta^{\sigma}$, with the exponent $\sigma < 0$. Intuitively, while many nodes have only a few triangles in their neighborhoods, a few nodes participate in many number of triangles with their neighbors. The local number of triangles is related to the clustering coefficient of graphs.

Eigenvalue Power Law

Siganos et.al. [192] examined the spectrum of the adjacency matrix of the autonomous systems Internet topology and reported that the 20 or so largest eigenvalues of the Internet graph are power-law distributed. Michail and Papadimitriou [156] later provided an explanation for the 'Eigenvalue Power Law', showing that it is a consequence of a power-law degree distribution.

We next shift our focus to dynamic properties of networks: using several snapshots over the course of network evolution, there are patterns that network measurements follow over time.

Shrinking Diameter

An established dynamic property is that of shrinking graph diameter. Leskovec et al. [136] showed that not only is the diameter of real graphs small, but it also shrinks over time (eventually stabilizing).

Densification Power Law

A related dynamic property is *densification*. Time-evolving graphs follow the 'Densification Power Law' with the approximate equation $E(t) \propto N(t)^{\beta}$, at all times t [136], where β is the densification exponent, and $E(t)$ and $N(t)$ are the number of edges and nodes at time t, respectively.

All networks we studied displayed densification, with β between 1.03 and 1.7. $\beta > 1$ indicates super-linearity between the number of nodes and the number of edges in real graphs. For example, when the number of nodes N in a graph doubles, the number of edges E more than doubles, hence densification. It also helps explain the shrinking diameter phenomenon.

Other network topology work

Park et. al. analyzed autonomous systems graphs [178], reporting measurements both on static snapshots of these networks and their dynamic properties. Chi et al. studied the evolution of communities over time [54]. The work by Kumar et al. [129] studies patterns in components, finding that many of the non-giant components feature star-like structures.

2.2.2 Models and generators of network topology

There has been significant of work on developing tractable mathematical models for real-world graphs and social networks, starting with the seminal Erdös-Rényi G_{np} model, where edges are randomly placed between nodes. Although unrealistic, this model leads to the fascinating phenomenon of *phase transition*: at a critical ratio of edges to nodes, the graph suddenly has high probability to have a 'giant connected component' (GCC). The GCC has size $O(n^{\frac{2}{3}})$ while no other component contains more than $O(\log n)$ vertices [68]. In the next chapter we will describe a phase transition that occurs in real graphs, the "gelling point."

Name	Uni/bipartite	Weights	$\|N\|$,$\|E\|$,$time$	Description
PostNet	Unipartite	None	250K, 218K, 80 days	Posts from blogs
NIPS	Unipartite	None	2K, 3K, 13 yr.	Citation network from NIPS
arXiv	Unipartite	None	30K, 60K, 13 yr.	Physics citations
Patent	Unipartite	None	4M, 8M, 17 yr.	Patent citations
IMDb	Bipartite	None	757K, 2M, 114 yr.	Actor-movie network
Netflix	Bipartite	None	125K, 14M, 72 mo.	User-movie ratings
BlogNet	Unipartite	Multi-edges	60K, 125K, 80 days	Social network of blogs based on citations
NetTraffic	Unipartite	Edge-weights (Packet-size)	21K, 2M, 52 mo.	Network traffic
Oregon	Unipartite	None	12K, 38K, 6 mo.	Autonomous systems
Auth-Conf	Bipartite	Multi-edges	17K, 22K, 25 yr.	DBLP Author-to-Conference associations
Key-Conf	Bipartite	Multi-edges	10K, 23K, 25 yr.	DBLP Keyword-to-Conference associations
Auth-Key	Bipartite	Multi-edges	27K, 189K, 25 yr.	DBLP Author-to-Keyword associations
CampOrg	Bipartite	Edge-weights (Amounts)	23K, 877K, 28 yr.	U.S. electoral campaign donations from organizations to candidates (available from FEC)
CampIndiv	Bipartite	Edge-weights (Amounts)	6M, 10M, 22 yr.	Election donations from individuals to organizations

Table 2.2: The datasets studied in this work.

Additional, more realistic models include the small world model [208], the *preferential attachment model* [9], the *Forest Fire Model* [136] and numerous more (the copying model, the 'winner does not take all' model [183], Heuristically Optimized Trade-offs [69]). We refer to the above as *emergent* generators, because they all have local rules (like preferential attachment), and yet they still manage to produce the macroscopic patterns we observe (small diameter, etc). There is a whole family of non-emergent generators, like degree-sequence matching; and models such as "Kronecker" graphs [135] that use global (rather than local) rules to generate networks. For a detailed survey of graph models and generators, the readers are referred to [35, 45, 66, 122].

We seek to improve upon these models. First, most of these models do not take into account multiple components, only modeling the GCC. Second, they do not take edge-weights into account. We will propose generators to account for these issues.

2.3 Data

We find several patterns in a wide range of real networks. These are described in detail in Table 2.2. They include both bipartite and unipartite, and weighted and unweighted graphs.

2.3.1 Data sets

Several of our graphs had no obvious weighting scheme: for example, a single patent or publication will cite another only a single time in the list of references.[3] The graphs that did have weights are also further divided into two schemes, *multi-edges* and *edge-weights*. In the edge-weights scheme, there is an obvious weight on edges, such as amounts in campaign donations, or packet-counts in network traffic. For multi-edges, weights are added if there is more than

[3]While a publication can cite another more than once in the text, our data did not provide this information. Studying how multiple citations occur within one paper is an intriguing problem, but was outside the scope of this work.

one interaction between two nodes. For instance, if a blog cites another blog at a given time, its weight is 1. If it cites the blog again later, the weight becomes 2.

The datasets are gathered from publicly available data. *NIPS*[4], *arXiv* and *Patent* [136] are academic paper or patent citation graphs with no weighting scheme. *IMDb* indicates movie-actor information, where an edge occurs if an actor participates in a movie [17]. *Netflix* is the dataset from the Netflix Prize competition[5], with user-movie links (we ignored the ratings); we also noticed that it only contained users with 100 or more ratings. *BlogNet* and *PostNet* are two representations of the same data, hyperlinks between blog posts [140]. In *PostNet* nodes represent individual posts, while in *BlogNet* each node represents a blog. Essentially, *PostNet* is a paper citation network while *BlogNet* is an author citation network (which contains multi-edges)[6].

NetTraffic records IP-source/IP-destination pairs, along with the number of packets sent, per unit time, and *Oregon* is an autonomous systems network.[7] *Auth-Conf*, *Key-Conf*, and *Auth-Key* are all from DBLP[8], with the obvious meanings. The data included 11 conferences, like "KDD," "SIGMOD," etc. *CampOrg* and *CampIndiv* are bipartite graphs from U.S. Federal Election Commission, recording donation amounts from organizations to political candidates and individuals to organizations.[9]

2.3.2 Issues in data collection

There are a few questions raised with respect to collecting network data, which are worthwhile to discuss before moving on.

- *Does edge deletion occur?* In a social network, it may be the case that an edge is considered to be removed from the network. For instance, if one person does not contact another within an email network over a certain period of time, the edge may be considered deleted. However, it is difficult to precisely define when an edge should be removed from the network, as there are cases where an edge is re-activated. Therefore, since edge deletion does not explicitly occur in any of our data sets, we do not remove edges.
- *How do we know when we have the whole network, especially for nodes not connected to the GCC?'* This is pertinent to our question of the non-GCC components. If a node is not connected to the GCC, it is difficult to detect using, for instance, the methods in web crawling. It is essentially impossible to detect all nodes in non-centralized domains such as the web or blogs. However, several of our data sets are centralized, such as political campaign contributions, *Patent*, *NIPS*, and *arXiv*. That the patterns we observe hold up in those data sets lends credence to the other data sets. A similar argument applies to other sampling errors: since the data sets acquired were gathered using different techniques, and the patterns still hold, that the same sampling error occurred in all is less likely.

[4] `www.cs.toronto.edu/~roweis/data.html`

[5] `www.netflixprize.com`

[6] See Section 5.1.1 on page 54 for a detailed description of how these two networks are constructed from raw data.

[7] University of Oregon *Route Views* project, `www.routeviews.org`

[8] `dblp.uni-trier.de/xml/`

[9] `www.cs.cmu.edu/~mmcgloho/fec/data/ fec_data.html`

In the next chapter we will use these data sets to find new patterns in networks.

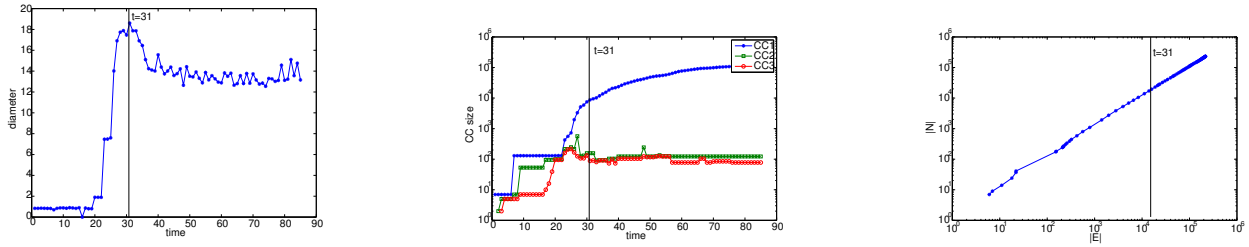
Chapter 3

New patterns in network formation

PROBLEM STATEMENT: *Given a collection of several different types of networks (political campaign contributions, computer network traffic, links in online social networks), can we identify patterns in the structure and formation of these networks?*

In this chapter we contribute several new patterns found in networks. As we discussed in the previous chapter, most work has focused on the *giant connected component* and ignored edge weights. Here we will examine some of these properties of networks. Our properties fall into two categories: unweighted (that is, properties that do not pay attention to edge weights) and weighted (properties that do take a weighting scheme into account).

3.1 Unweighted graphs



(a) Diameter(t) (b) GCC, CC2 and CC3 sizes (log-lin) (c) N(t) vs E(t)

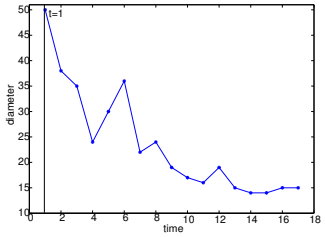
Figure 3.1: Properties of *PostNet* network. Notice that we experience an early gelling point (point of maximum diameter) at (a) (diameter versus time), stabilization/oscillation of the DC sizes in (b) (size of 1st, 2nd, and 3rd CC, versus time). The vertical line marks the gelling point. Part (c) gives $N(t)$ vs $E(t)$ in log-log scales - the good linear fit agrees with the Densification Power Law.

3.1.1 Pattern UW1: Gelling point

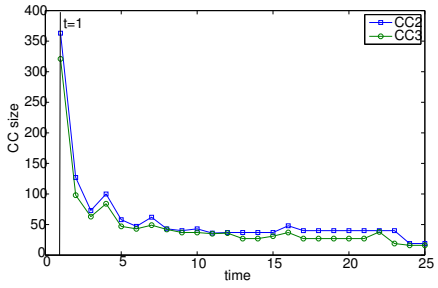
Studying the effective diameter of the graphs, we notice that there is often a point in time when the diameter spikes. Before that point, the graph is more or less in an establishment period, typically consisting of a collection of components, all of small size. This "*gelling point*" seems to also be the time where the GCC "takes off." After the gelling point, the graph obeys the expected rules, such as the densification power law; its diameter decreases or stabilizes; the giant connected component keeps growing, absorbing the vast majority of the newcomer nodes.

Observation 3.1.1 (Gelling point) *Real graphs exhibit a gelling point, at which the diameter spikes and (several) disconnected components gel into a giant component. The gelling point is defined as the point of maximum diameter.*

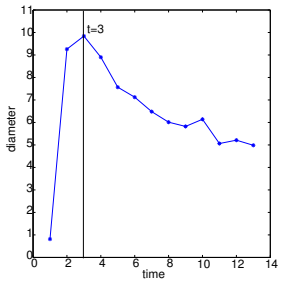
In most of these graphs, both unipartite and bipartite, there are clear gelling points. For example, in *NIPS* the diameter spikes at $t = 8$ years, which is a reasonable time for an academic community to gel. In some networks, we only see one side of the spike, as the time resolution is often coarse enough that the gelling point happens within the first time bin (*Patent*).



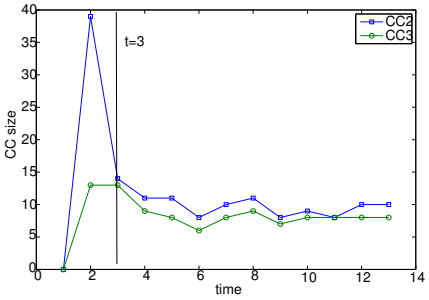
(a) *Patent* Diam(t)



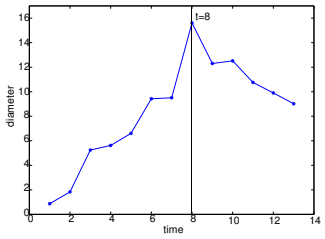
(b) *Patent* second and third DC



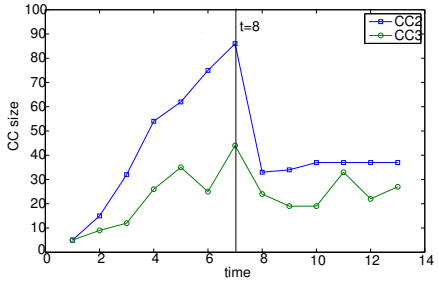
(a) *arXiv* Diam(t)



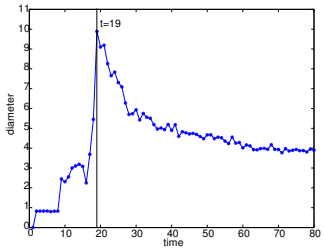
(b) *arXiv* second and third CCs



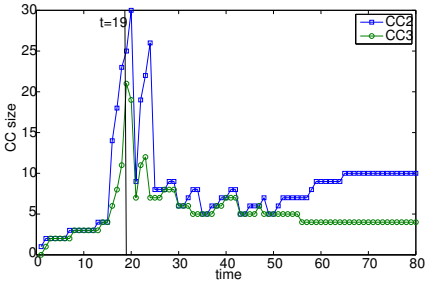
(a) *NIPS* Diam(t)



(b) *NIPS* second and third CCs

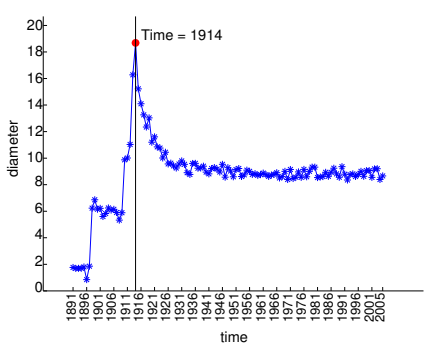


(a) *BlogNet* Diam(t)

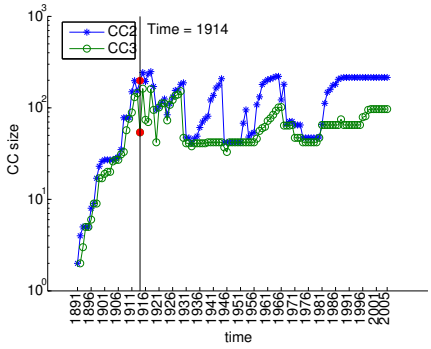


(b) *BlogNet* second and third CCs

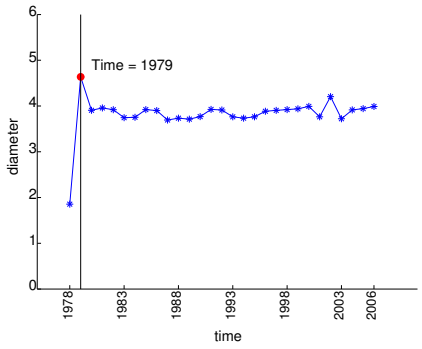
Figure 3.2: Properties of unipartite networks. Diameter plot (left column), and second and third CCs over time (right); vertical line marks the gelling point.



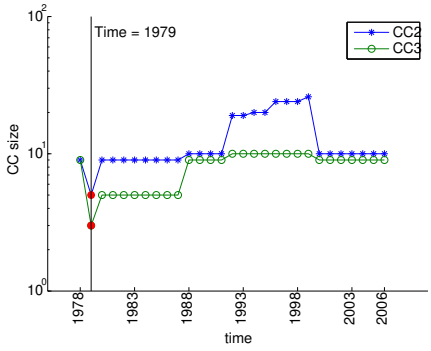
(a) *IMDb* Diam(t)



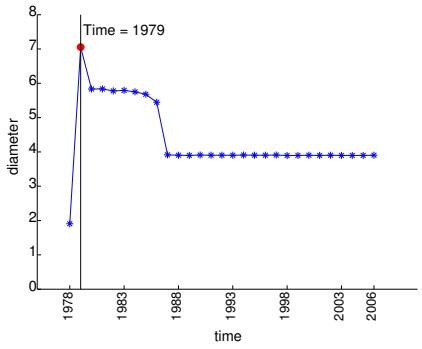
(b) *IMDb* second and third CCs



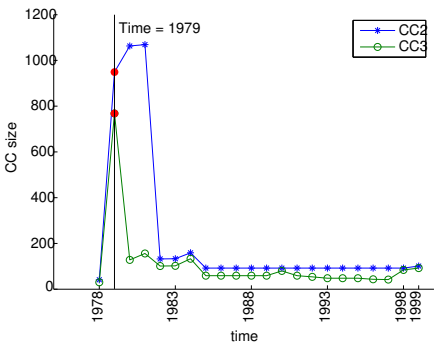
(c) *CampOrg* Diam(t)



(d) *CampOrg* second and third CCs



(e) *CampIndiv* Diam(t)



(f) *CampIndiv* second and third CCs

Figure 3.3: Properties of bipartite networks. Diameter plot (left column), and second and third CCs over time (right), with vertical line marking the gelling point. Again, all datasets exhibit an early gelling point, and stabilization of the second and third CCs.

3.1.2 Pattern UW2: Oscillating secondary components

What happens to the smaller components when the GCC takes off? We particularly studied the second- and third-largest connected component over time.[1] We notice that, after the gelling point, the sizes of these components *oscillate* over time. Further investigation shows that the oscillation may be explained as follows: new-comer nodes typically link to the GCC; very few of the newcomers link to the 2nd (or 3rd) CC, helping them to grow slowly; in very rare cases, a newcomer links both to an DC, as well as the GCC, thus leading to the absorption of the DC into the GCC. It is exactly at these times that we have a drop in the size of the 2nd CC: Note that edges are not removed, thus, what is reported as the size of the 2nd CC is actually the size of yesterday's 3rd CC, causing the apparent "oscillation."

An unexpected (to us, at least) observation is that the largest size these components can get seems to be constant in time (though variable across different networks). This is counter-intuitive: based on Erdös-Rényi random graphs, we would expect the size of the smaller components to grow with increasing N. Using scale-free arguments, we would expect the DCs to have size that would be a (small, but constant) fraction of the size of the GCC. To our surprise, this *never* happened on any of the real graphs we studied.

Observation 3.1.2 (Oscillating secondary components) *After the gelling point, the secondary and tertiary connected components remain of approximately constant size, with small oscillations.*

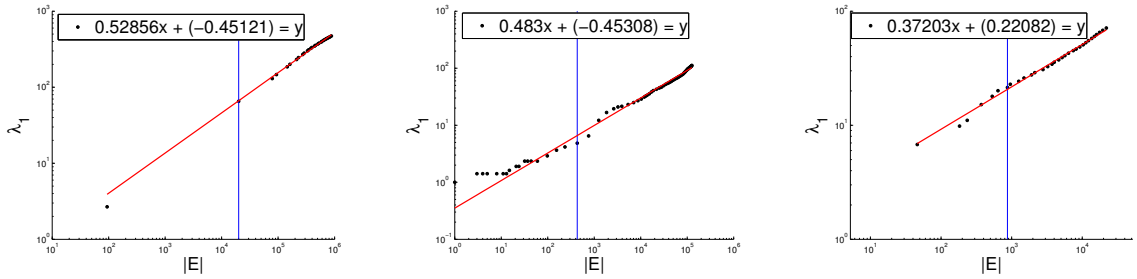
We show results for *PostNet* in Fig. 3.1, including the diameter plot (Fig. 3.1(a)), sizes of the first, second, and third-largest connected components on log-linear scale (Fig. 3.1(b)), and densification plot (Fig. 3.1(c)). Results from other networks are similar, and are shown in condensed form (Fig. 3.2 for unipartite graphs, and Fig. 3.3 for bipartite graphs). The left column shows the diameter plots, and the right column shows the second and third-largest CCs.

The second columns of Fig. 3.2 and Fig. 3.3 show the second and third-largest CC sizes versus time. Notice that, after the "gelling" point (marked with a vertical line), they all oscillate about constant value (different for each network). The only extreme cases are datasets with unusually high connectivity. For example, *Netflix* has very small DCs. This may be explained by the fact the dataset is masked, omitting users with less than a hundred ratings (possibly to further protect the privacy of the encrypted user-ids). This may be related to other issues discussed in Section 2.3.2. Therefore, the graph has abnormally high connectivity.

3.1.3 Pattern UW3: Principal eigenvalue over time

Plotting the largest (principal) eigenvalue of the 0-1 adjacency matrix $\mathbf{A}$ of our datasets over time, we notice that the principal eigenvalue grows following a power law with increasing number of edges. This observation is true especially after the *gelling point*, or the time of maximum diameter, as we described earlier. See [136] for details.

[1] We call these the "next-largest connected components" in [153]. We adopt the general term "disconnected components" for all non-giant components in [114], referring to the second-largest connected component as "DC1," etc.. We note this term is ambiguous, as the components are not disconnected within themselves, but rather disconnected from the GCC.



(a) Committee - Candidate (b) Blog Network (c) Author - Conference

Figure 3.4: Illustration of the LPL. 1^{st} eigenvalue $\lambda_1(t)$ of the *0-1* adjacency matrix **A** versus number of edges $E(t)$ over time. The vertical lines indicate the gelling point.

Observation 3.1.3 (λ_1 Power Law (LPL)) *In real networks, the principal eigenvalue $\lambda_1(t)$ and the number of edges $E(t)$ over time follow a power law with exponent less than 0.5, especially after the 'gelling point'. That is,*

$$\lambda_1(t) \propto E(t)^{\alpha},\ \alpha \leq 0.5$$

We report the power law exponents in Fig. 3.4. Note that we fit the given lines *after* the gelling point which is shown by a vertical line for each dataset. Notice that the given slopes are less than 0.5, with the exception of the *CampaignOrg* dataset, with slope ≈ 0.53. See [5] for details.

3.1.4 Pattern UW4: Stable Graph Fractal Dimension among components

We know from Observation 3.1.2 that DCs only reach a certain size. But will they grow with the same rate (before absorption)? To answer this, we look at the *graph fractal dimension* of the three largest connected components over time. We define the graph fractal dimension as the ratio of the logarithm of the number of edges to the logarithm of the number of nodes, the densification exponent as described in Section 2.2.1.

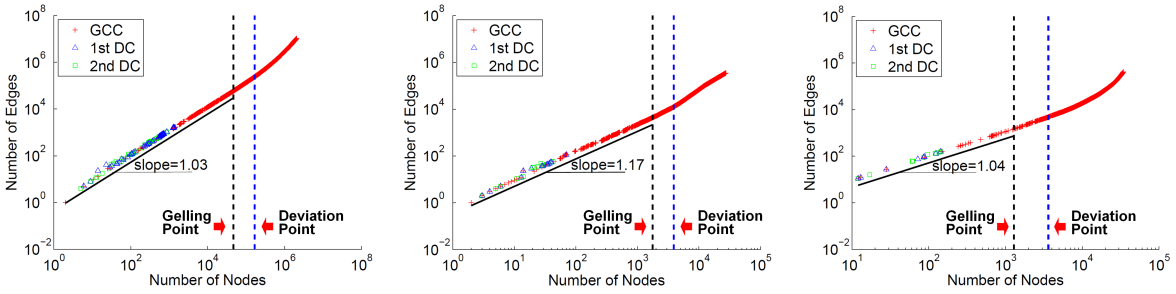
Observation 3.1.4 (Stable component GFD) *The first, second, and third largest connected components grow with the same rate. Their graph fractal dimensions remain the same until a deviation point. The deviation point is close to the "gelling" point where the diameter starts to shrink.*

Results are shown in Figure 3.5.

This observation is interesting, since it implies that some barriers between the nodes seem to collapse after the gelling point, and the nodes in the network are connected with higher rate than before the gelling point.

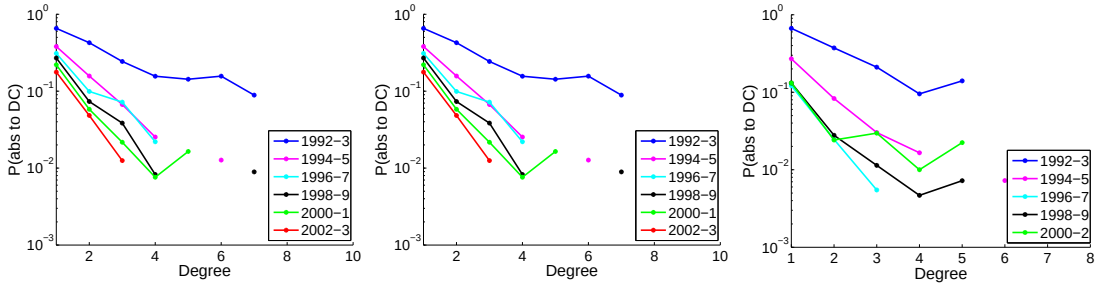
3.1.5 Pattern UW5: Exponential "Rebel" probability (ERP)

The next question is, what is the probability of newcomers not joining to the GCC? Given the degree of a newcomer, what is the probability that it will be absorbed, or not absorbed to GCC?



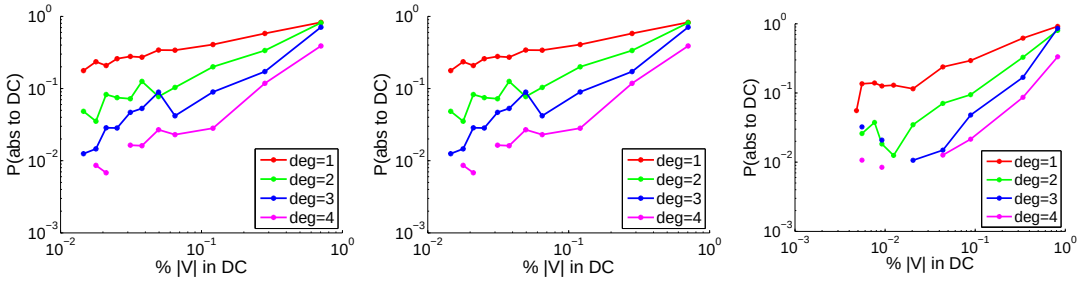
(a) U.S. Patent: Top 3 CC (b) HEP-TH: Top 3 CC (c) HEP-PH: Top 3 CC

Figure 3.5: Growth of connected components in terms of the graph fractal dimension. Each point represents the snapshot of a connected component over time. The largest component is "GCC," the second-largest the "1st DC," and the third-largest is the "2nd DC." Notice that the graph fractal dimension (slope of the plots) remains constant until a 'deviation point'(the second vertical line) close to a 'gelling point'(the first vertical line), and starts to increase after that. The deviation points are about one year after the gelling points.



(a) U.S. Patent (b) HEP-TH (c) HEP-PH

Figure 3.6: Pattern UW5: P(Absorption to DC) vs. Degree in log-lin scale. Notice the linear drop of the probability as the degree increases.



(a) U.S. Patent (b) HEP-TH (c) HEP-PH

Figure 3.7: Pattern UW5: Probability of "rebelling"; that is, joining to a "disconnected component" outside the GCC. The plots show P(Absorption to DC) vs. Portion of nodes in disconnected components in log-log scale. Notice that the slopes of curves increase as degree increases.

We call it the "rebel" probability and give its relations to the degree of newcomers and the portion of nodes in the "disconnected components." We first give the relationship of the rebel probability and the degree of newcomers (as they arrive) in Figure 3.6. In Figure 3.6, we see that the probability is linear to the degree in log-lin scale where the slope decreases as the network grows. In addition, we show the relationship of the rebel probability and the portion of nodes in DC in Figure 3.7. From Figure 3.7, we see the probability is linear to the portion of nodes in DC in log-log scale, and the slope increases as the degree increases. Given these two observations, we give empirical rebel probability of newcomers as a function of the degree(d) and the portion(s) of nodes in DC in Observation 3.1.5 which we call the ERP (Exponential Rebel Probability) pattern.

Observation 3.1.5 (Exponential Rebel Probability (ERP)) *Given the node portion s of DCs, the probability P_{rebel} of a newcomer to be absorbed in DCs is exponential to the product of some number α (which depends on the graph at that point in time), the degree d of the newcomer, and the log of the node portion s:*

$$P_{rebel} \propto e^{\alpha d(logs)} = s^{\alpha d}$$

3.2 Weighted graphs

Here we try to find patterns that weighted graphs obey. In this section, we consider graphs to be directed (and impose a single direction in bipartite graphs), as this will be an important consideration on the weights. To illustrate this pattern we will use the *NetTraffic* data set. The dataset consist of quadruples: (IP-source, IP-destination, timestamp, number-of-packets), where timestamp is in increments of, say, 30 minutes. Thus, we have multi-edges, as well as total weight for each (source, destination) pair. Let $W(t)$ be the total weight up to time t (i.e., the grand total of all exchanged packets across all pairs), $E(t)$ the number of distinct edges up to time t, and $E_d(t)$ the number of multi-edges (the d subscript stands for *duplicate* edges), up to time t.

We present several empirical laws that our datasets seem to follow.

3.2.1 Pattern W1: Weight Power Law (WPL)

As defined above, suppose we have a computer network. Let $E(t)$ be the total unique edges up to time t (i.e., the count of pairs of machines with at least one connection between them). Let $W(t)$ be the total count of packets up to time t. Is there a relationship between $W(t)$ and $E(t)$? If every pair generated k packets, the relationships would be linear: if the count of pairs double, the packet count would double, too. This is reasonable, but it doesn't happen! In reality, the packet count over-doubles, following the WPL below. We shall refer to this phenomenon as the "*fortification effect*": more edges in the graph imply super-linearly higher total weight.

Observation 3.2.1 (Weight Power Law (WPL)) *Let $E(t)$, $W(t)$ be the number of edges and total weight of a graph, at time t. Then, they follow a power law*

$$W(t) = E(t)^w$$

where w is the weight *exponent. Power-laws also link the number of nodes $N(t)$, and the number of multi-edges $E_d(t)$, to $E(t)$, with exponents n and w_{dup}, respectively.*

The weight exponent w ranges from 1.01 to 1.5 for the real graphs we have studied. The highest value corresponds to campaign donations (see Figure 3.8). Super-active organizations that support many campaigns also tend to spend even more money per campaign than the less active organizations. For bipartite graphs, we show the $N_{\rm src}$, $N_{\rm dst}$ exponents for the source and destination nodes (which also follow power laws: $N_{src}(t) = E(t)^{N_{\rm src}}$ and similarly for $N_{dst}(t)$).

Fig. 3.11 shows all these quantities, versus $E(t)$, for several datasets. The plots are all in log-log scales, and straight lines fit well. We report the slopes in Table 3.1.

3.2.2 Pattern W2: Edge Weights Power Law

We observe that the weight of a given edge and weights of its neighboring two nodes are correlated. Our observation is similar to Newton's Gravitational Law stating that the gravitational force between two point masses is proportional to the product of the masses.

Observation 3.2.2 (Edge Weights Power Law (EWPL)) *Given a real-world graph $\mathcal{G}$, 'communication' defined as the weight of the link between two given nodes has a power law relation with the weights of the nodes. In particular, given an edge $e_{i,j}$ with weight $w_{i,j}$ and its two neighbor nodes i and j with weights w_i and w_j, respectively,*

$$w_{i,j} \propto \left(\sqrt{(w_i - w_{i,j}) * (w_j - w_{i,j})} \right)^{\gamma}$$

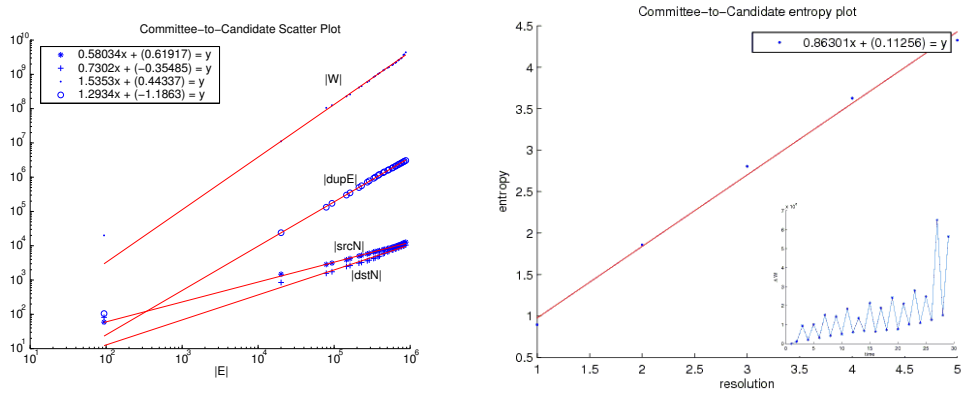
We report corresponding experimental findings in Fig. 3.9. Note that in the committee-candidate network, the fit only applies for edge weights around 10 and above. That is, the smallest edge weights tended to occur between nodes of high edge weights otherwise. This is most likely because in that network, most edges are of high weight: only 3 percent of edges had weight 10 (dollars) or less; in general for networks we observe that many edges have the smallest weight possible, so this is a deviation. Therefore, in this case, a node may simply need a high weight in order to by chance have such a low-weight edge. Or, it could have something to do with the domain (heavy-weight nodes are high-funded candidates such as those in presidential elections, and may receive many small-amount donations).

3.2.3 Pattern W3: Snapshot Power Laws (SPL)

What about a static snapshot of a graph? If node i has out-degree out_i, what can we say about its out-weight $outw_i$? It turns out that there is a "fortification effect" here, too, resulting in more power laws, both for out-degrees/out-weights as well as for in-degrees/in-weights.

Specifically, at a given point in time, we plot the scatter plot of the in/out weight versus the in/out degree, for all the nodes in the graph, at a given time snapshot. An example of such a plot is in Fig. 3.10 (a) and (b). There, every point represents a node and the x and y coordinates are its degree and total weight, respectively. To achieve a good fit, we bucketize the x axis with logarithmic binning [169], and, for each bin, we compute the median y.

We observed that the median values of weights versus mid-points of the intervals follow a power law for all datasets studied. Formally, the "Snapshot Power Law" is:



(a) WPL plot (b) entropy plot

Figure 3.8: Weight properties of *CampOrg* donations: (a) shows all the power laws as well as the WPL; the slope in (b) is ~ 0.86 indicating bursty weight additions over time.

Observation 3.2.3 (Snapshot Power Law (SPL)) *Consider the i-th node of a weighted graph, at time t, and let out_i, $outw_i$ be its out-degree and out-weight. Then*

$$outw_i \propto out_i^{ow}$$

where ow is the out-weight-exponent *of the SPL. Similarly, for the in-degree, with in-weight-exponent iw.*

We studied the snapshot plots for several time-stamps (for brevity, we only report the slopes for the final timestamp in Table 2 for all the datasets we studied). We observed that SPL exponents of a graph over time remains almost constant. In Fig. 3.8 (c) ((d)), the inset plot shows how the iw (ow) exponent changes over time (years) for the *CampOrg* dataset. We notice that iw and ow take values in the range [0.9-1.2] and [0.95-1.35], respectively. That is:

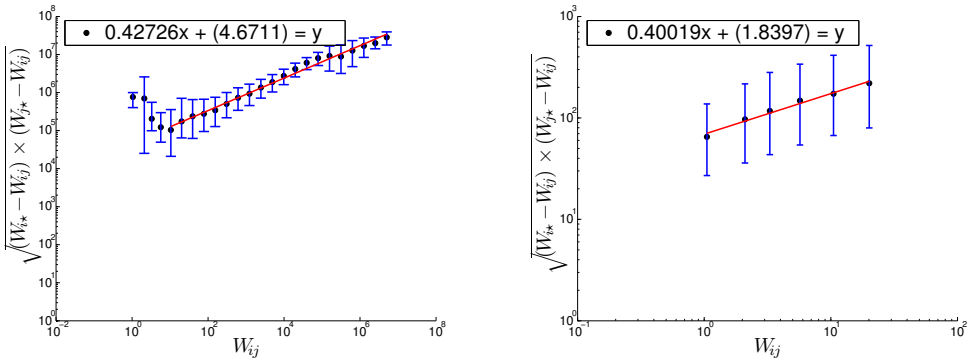
Observation 3.2.4 (Persistence of Snapshot Power Law) *The in- and out-exponents iw and ow of the SPL remain about constant, over time.*

We observe that all SPL exponents are > 1 (see Table 3.1), which imply a "*fortification effect*" with super-linear growth.

3.2.4 Pattern W4: Bursty/self-similar weight additions

Having established some properties of edge weights on static snapshots of graphs, we next observe dynamic properties.

We tracked how much weight a graph adds at each time interval. Using entropy plots, we observed that the weight additions over time display self-similarity. For those weighted graphs where the edge weight is defined as the number of recurrences of that edge, the slope of the entropy plot was greater than 0.95, suggesting uniformity. On the other hand, for those graphs where weight is not in terms of multiple edges but some other feature of the dataset such as the amount of donations for the FEC dataset, we observed that weight additions are more bursty, the slope being as low as 0.6 for the Network Traffic dataset. Fig. 3.11 (b) column shows the entropy

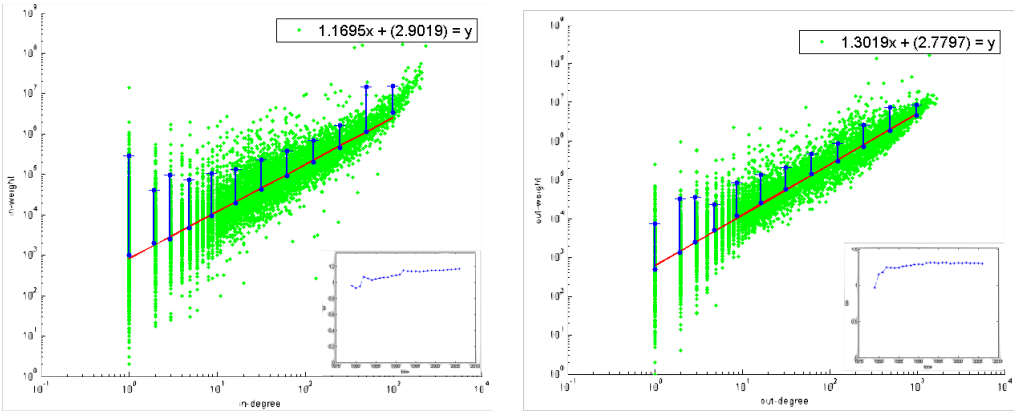


(a) Committee - Candidate (b) Blog Network

Figure 3.9: Illustration of the EWPL. Given the weight of a particular edge in the final snapshot of real graphs (x-axis), the multiplication of total weights (y-axis) of the edges incident to two neighboring nodes (minus the edges between them) follow a power law. A line can be fit to the median values after logarithmic binning on the x-axis. Upper and lower bars indicate 75% and 25% of the data, respectively.

	w	N_{src}	N_{dst}	w_{dup}	iw	ow	fd
CampOrg	1.53	0.58	0.73	1.29	1.16	1.30	0.86
CampIndiv	1.36	0.53	0.92	1.14	1.05	1.48	0.87
BlogNet	1.03	0.79	NA	NA	1.01	1.10	0.96
Auth-Key	1.01	0.90	0.70	NA	1.01	1.04	0.95
Auth-Conf	1.08	0.96	0.48	NA	1.04	1.81	0.96
Key-Conf	1.22	0.85	0.54	NA	1.26	2.14	0.95

Table 3.1: Power law exponents for all the weighted datasets we studied: The x-axis being the number of non-duplicate edges E, w: WPL exponent, N_{src} , N_{dst}: WPL exponent for source and destination nodes respectively (if the graph is unipartite, then N_{src} is the number of all nodes), w_{dup}: exponent for multi-edges, iw, ow: SPL exponents for indegree and outdegree of nodes, respectively. Exponents above 1 indicate fortification/superlinear growth. Last column, fd: slope of the entropy plots, or information fractal dimension. Lower fd means more burstiness.



(a) inD-inW snapshot (b) outD-outW snapshot

Figure 3.10: Snapshot Power Laws of *CampOrg* donations: (a) and (b) have slopes > 1 ("fortification effect"), that is, that the more campaigns an organization supports, the superlinearly-more money it donates, and similarly, the more donations a candidate gets, the more average amount-per-donation is received. Inset plots show iw and ow versus time. Note they are very stable over time.

plots for the weighted datasets we studied. ΔW values over time are also shown in insets at the bottom right corner of each figure.

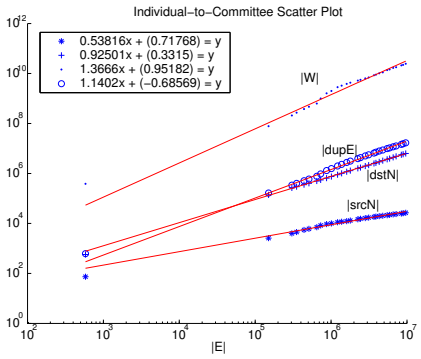
Observation 3.2.5 (Bursty/self-similar weight additions) *In all our graphs, the addition of weight ($\Delta W(t)$) was self-similar, with fractal dimension ranging from ≈1 (smooth/uniform), down to* 0.6 *(bursty).*

3.2.5 Pattern W5: LWPL: Weighted principal eigenvalue over time

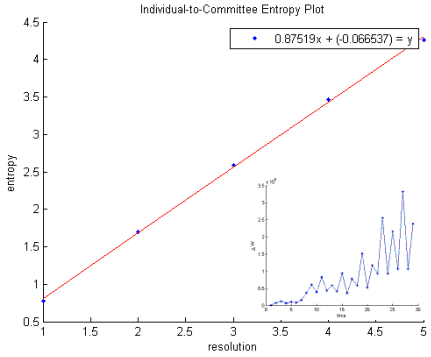
Given that unweighted (0-1) graphs follow the λ_1 Power Law, one may ask if there is a corresponding law for weighted graphs. To this end, we also compute the largest eigenvalue $\lambda_{1,w}$ of the *weighted* adjacency matrix $\mathbf{A}_w$. The entries $w_{i,j}$ of $\mathbf{A}_w$ now represent the actual edge weight between node i and j. We notice that $\lambda_{1,w}$ increases with increasing number of edges following a power law with a higher exponent than that of its λ_1 Power Law. We show the experimental results in Fig. 3.12.

Observation 3.2.6 ($\lambda_{1,w}$ Power Law (LWPL)) *Weighted real graphs exhibit a power law for the largest eigenvalue of the weighted adjacency matrix $\lambda_{1,w}(t)$ and the number of edges $E(t)$ over time. That is,*

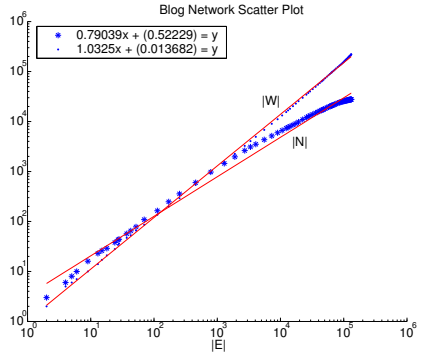
$$\lambda_{1,w}(t) \propto E(t)^{\beta}$$



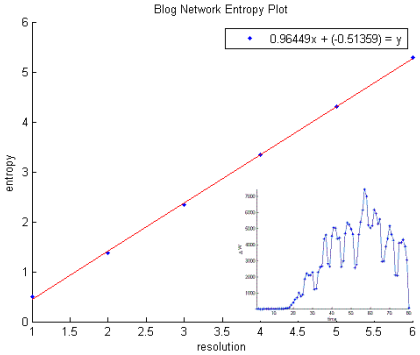
(a) *CampIndiv* WPLs



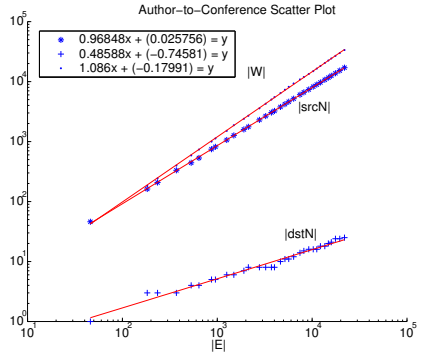
(b) *CampIndiv* entropy



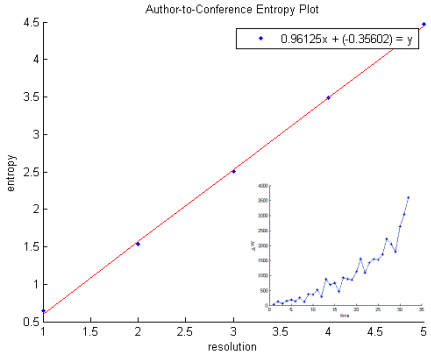
(c) *BlogNet* WPLs



(d) *BlogNet* entropy

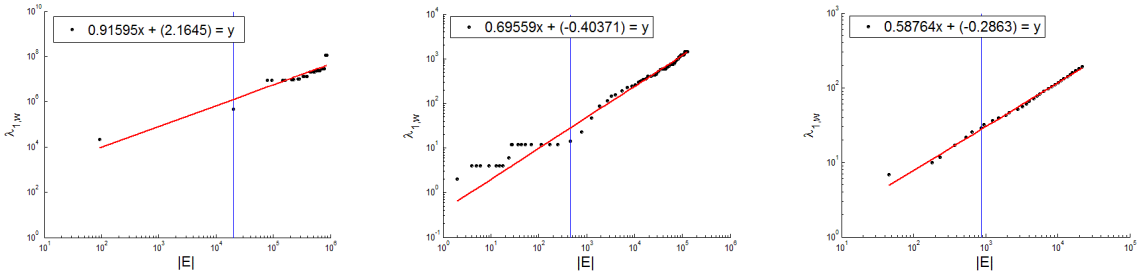


(e) *Auth-Conf* WPLs



(f) *Auth-Conf* entropy

Figure 3.11: Properties of weighted networks. Top: weight power laws for *CampIndiv*(W, E_d, N; vs E). The slopes for weight W and multi-edges E_d are above 1, indicating "fortification." Bottom: entropy plots for weight addition. Slope away from 1 indicates burstiness (eg., 0.88 for *CampIndiv*) The inset plot shows the corresponding time sequence ΔW versus time.



(a) Committee - Candidate (b) Blog Network (c) Author - Conference

Figure 3.12: Illustration of the LWPL. 1^{st} eigenvalue $\lambda_{1,w}(t)$ of the *weighted* adjacency matrix $\mathbf{A}_w$ versus number of edges $E(t)$ over time. The vertical lines indicate the gelling point.

In our experiments, β ranged from 0.5 to 1.6.

3.3 Summary of patterns and contributions

While previous work has revealed a number of interesting properties of how networks grow as nodes and edges are added, these discoveries shed light on how *existing* nodes behave in relation to each other. We give patterns for several spectral characteristics of real-world graphs, namely largest eigenvalues of both $0-1$ and weighted adjacency matrices, which are in connection with other parameters of the graphs, such as degrees of nodes and the edge density. We also provided explanations for small deviations from these laws, such as plateaus and jumps, especially those occurring before the gelling point.

In summary, our contributions are the following:

- We have analyzed several large data sets from a wide variety of domains, from political campaign contributions, to links in blogs, to computer network traffic, to bipartite actor-movie networks. We make several discoveries.
- *Oscillating secondary components and other component patterns.* We have showed the following *dynamic* patterns in *unweighted networks*, involving the rarely-studied non-giant components: that the sizes of the next-largest connected components appear to oscillate, that the *graph fractal dimension* of them is stable over time, and that the "rebel" probability—the probability that an incoming node will join a non-giant component—is exponential with the new node's degree.
- *Snapshot Power Laws on static, weighted networks*. We have shown several patterns in *static snapshots of weighted networks:* that the weight of a link between two nodes follows a function of the total weight of those nodes, that degree versus weight plots follow power laws.
- *Fortification, and other dynamic, weighted properties*. We have shown several *dynamic* patterns in *weighted networks:* That the total weight of the graph increases superlinearly over time with the number of edges, that edge weight additions are bursty, and that the weighted principal eigenvalue follows a power law with number of edges, over time.

Chapter 4

Models of network formation

PROBLEM STATEMENT: *Given knowledge of several established patterns (heavy-tailed degree distribution, shrinking diameter, densification), as well as newly discovered ones (oscillating sizes of next-largest connected components, power laws in edge weights) can we propose intuitive models that will generate these behaviors?*

Now that we have explored several new properties of networks, our next goal is to propose generative models to help explain these behaviors. Graph generators are vital for simulation of algorithms (like computer network routing algorithms), for simulation of rumor (or virus, or influence) propagation, and many other settings. In several such settings, real graphs may be difficult or even impossible to collect: for example a who-believes-whom graph is only in the mind of the human subjects; a who-mails-whom graph may be protected by privacy laws. They may be helpful in modeling for "what-if" scenarios and spotting anomalies.

We propose two generative models: the Butterfly Model and the Recursive Tensor Model (RTM). As we will detail, they both reproduce both previously-established and newly-discovered "laws" for networks. However, the Butterfly Model is agent-based, while RTM uses self-similarity-inducing tensor multiplication. The former has the advantage of being emergent, while the latter is more useful for theoretical analysis.

We aim to reproduce the following properties detailed in the previous two chapters:

- Established patterns: Power laws for in- and out-degree distribution, small, shrinking diameter, possibly after a "gelling point," and densification.
- New unweighted patterns: constant/oscillating DCs, power law growth of unweighted and weighted principal eigenvalues, stable graph fractal dimension among components, exponential "rebel" probability.
- New weighted patterns: Weight Power Law, Edge Weights Power Law, Snapshot Power Laws, bursty weight additions.

4.1 An emergent generator: Butterfly

The goal is to build a generative model for network topology. The generated topology should match properties observed in our work as well as properties observed in previous work.

Moreover, we want an *emergent* generator, that will follow a simple, local behavior, out of which these global patterns will naturally emerge. Thus, we plan to have nodes arriving one at a time, and we want to design the method with which newcomers link to existing nodes, analogously to the 'preferential attachment' of Barabasi et. al. [17], but without the pitfalls of preferential attachment.

To achieve a heavy-tailed in-degree distribution, some form of preferential attachment will suffice. In order to even *have* components disconnected from the GCC, we allow some newcomers to form zero links and be isolated. Other newcomers later become 'bridges', that can link the GCC with a DC, so the DC is absorbed.

To achieve a power-law in the out-degree distribution, we *vary* one of the parameters of our model, so that it takes values uniformly distributed.

4.1.1 Definition of proposed Butterfly model

With these considerations, we present the following model, which we call the Butterfly model. Incoming nodes may behave as "social butterflies" by choosing more than one starting point, or "host," in their interactions: meeting nodes in the vicinity of the host, out-linking to some of them, and flying away. The model uses three parameters. The first, p_{link}, determines how often a link is formed between two nodes, and it is the same for all newcomers. The others, p_{host} and p_{step} are "friendliness" parameters: p_{step} decides whether the 'butterfly' will take one more step in its random walk; p_{host} is the probability it will take one more host. We set p_{step} to be *different* for each newcomer, uniformly distributed in the range [0,1]. We set p_{host} to be the same for all newcomers.[1]

In the model, nodes join the network one at a time. With probability p_{host}, an arriving node, denoted $current$, picks a host at random, and is assigned a p_{step} probability from a uniform distribution. After choosing the host, $current$ travels in a random walk, recursively choosing at random one of the neighboring nodes (including both in- and out-links), taking each further step with probability p_{step}. The random walk may also be weighted, where instead of choosing uniformly among neighbors, the choice is weighted according to how many previous links there have been to a given neighbor. Each time $current$ visits (or re-visits) a node, it out-links to the visited node with p_{link} probability. Once the walk stops, $current$ flips a new coin, choosing a new host (and a subsequent random walk) with probability p_{host} and repeating, until no new hosts are chosen (host-choosing terminated with probability $1 - p_{\text{host}}$). Pseudocode for the model is shown in Fig. 4.1.

From these rules and parameters, we can calculate the expected number of hosts as $(1/(1 - p_{host}) - 1)$ and expected number of steps per host chosen is $(1/(1 - p_{step}) - 1)$.

We choose $p_{host} = 0.5$ so the expected number of hosts is 1. However, once in a while, an arriving node chooses multiple hosts, allowing the possibility of two formerly disconnected components joining. This joining will cause the GCC to absorb smaller components, to reproduce the oscillation of secondary components as observed in real data[2].

[1]Letting p_{host} vary uniformly, also performed well empirically.

[2]Note that the Butterfly model has been built incrementally in [153] and [114], . In this document we have refined the model further to created weighted graphs.

```
// generates a realistic looking graph
function butterfly
   global p_link = 0.3
   global p_host = 0.5
   global G = new_graph()
   for n = 1:N
       current=new_node()
       p_step = SampleUniform(0,1)
       G.add_node(current)
       while (SampleUniform() < p_host)
           host = G.random_node()
           visit(current, host)
   return(G)

// input: a newcomer, and host node to visit
// effect: it updates G, with the new edges,
//   after current links to existing nodes
function visit(current, host)
    // with probability p_step, visit and continue random walk
    if (SampleUniform() < current.p_step)
        // with prob. p_link, link to the contact_node
        if (rand() < p_link)
            G.add_directed_edge(current, host)
        next_visit = chooseNeighbor(host, isWeighted)
        visit(current, next_visit)

// Chooses next neighbor to visit in (possibly weighted) random walk
function chooseNeighbor(host, isWeighted)
    candidates = {}
    for nb in host.neighbors()
        if (isWeighted)
            for i = 1:G.weight(host, nb)
                candidates.append(nb)
        else
            candidates.append(nb)
    return chooseRandom(candidates)
```

Figure 4.1: Pseudocode for *Butterfly* model.

Unlike many previous models, this generator produces disconnected components due to the different number of hosts that are chosen. Some nodes may choose 0 hosts and form a component entirely disconnected from the others. Other nodes may choose multiple hosts in different components, merging components together.

We will next perform some justification for why the Butterfly model should reproduce the properties. We then demonstrate these properties empirically by measuring the network generated by the Butterfly model.

4.1.2 Analytical validation of Butterfly

We find that choosing the parameters as defined in the above table, the results are remarkably similar to what is displayed in real graphs. Note that the model displays a stable or shrinking diameter, and that after a burning off period the second and third components demonstrate a threshold at which they do not grow further without joining the GCC.

Theorem 1 *For a given host, the number of visits an arriving node forms follows power-law distribution with exponent* -2.

Proof 1 *Taking* p_{host} *constant, the expected number of steps* y *that an arriving node will take is* $\frac{1}{1-p_{step}} - 1$. *(*$1 - p_{step}$ *is the probability of stopping traveling at any time point, so the number of steps taken before stopping follows a geometric distribution with mean* $\frac{1}{1-p_{step}}$*, and the number of visits is the number of steps before deciding to stop, the mean minus one.) If* $p_{step} \sim \mathrm{Unif}(0, 1)$*, we can do a transformation to find the distribution of the expected number of steps* y *[42]:*

We represent $Y = g(X)$*, and the distribution over* X *is uniform,* $f_X(x) = 1$*. Since the function* $\frac{1}{x}$ *is strictly monotone decreasing, then* g *has inverse* $h = g^{-1}$*, specifically* $h(y) = \frac{1}{y}$*. So we have*

$$f_Y(y) \propto f_X(h(y)) * |\frac{dh(y)}{dy}| = h(y) * |-\frac{1}{x^2}| = f_Y(y) = x^{-2}$$

So, expected number of visits follows a power law with exponent -2.

Since we hold p_{link} constant, we believe that the number of visits will be closely related to the degree, so that empirically we have a power law distribution. Formally proving this would require taking into account the random effects of p_{link} as well as effects from multi-edges and multiple hosts.

We also note that empirically, we find that the in-degree distribution follows a power-law as well. We have no formal justification for this; however we believe that it follows from rich-get-richer behavior: the higher a degree of a node, the more chances it has to encounter a new node behaving in the random-walk cycle.

We show that the Exponential Rebel Probability pattern of the growth of connected components can be derived from our Butterfly model. In the model, the probability of choosing to connect to a given component is dependent on the number of nodes in the component, since the hosts are picked at uniform. In order for an arriving node to join a component, it must first choose a host within that component. Noting the random walk can access any node in the component, but not outside, we can derive the probability of a node "rebelling" represented as an event variable R, given the portion s of the graph's nodes in DCs and the degree d of the newcomer.

The algorithm gives us the following distributions: the number of hosts chosen has a geometric distribution with parameter $1 - p_{host}$ (the number of coin flips until "no new host"). The length of the random walk after a given host has a geometric distribution with parameter $1 - p_{step}$ (the number of links is being however many coin flips until "not-link"). So the total degree is the sum of the random walks. From these we can show the probability distribution of rebelling given degree.

Theorem 2 (Probability of "Rebelling")

$$P(R = true|s, D = d > 0) = \frac{\sum_{h=1}^{d} NBin(d,h,1-p_{step}) * Geom(h+1,1-p_{host}) * s^h}{\sum_{h=1}^{d} NBin(d,h,1-p_{step}) * Geom(h+1,1-p_{host})} \tag{4.1}$$

where NBin and Geom are the PDF of negative binomial and geometric distribution:
$NBin(y, r, p) = \binom{r+y-1}{y} p^r (1-p)^y$ *and*
$Geom(x, p) = (1-p)^{x-1} p$, *when* $p_{\text{link}} = 1$.

Proof 2 *We have the following probability distributions for the number of hosts, the degree (number of visits) given the number of hosts, and the probability of rebelling ("missing the GCC") given the number of hosts:*

- $P(H = h) \sim Geom(h + 1, 1 - p_{host}) = p_{host}^h (1 - p_{host})$
- $P(D = d|H = h) \sim NBin(d, h, 1 - p_{step}) = \binom{d+h-1}{h-1} (1 - p_{step})^h p_{step}^d$
- $P(R = true|H = h) = s^h$

Since D and R are conditionally independent given H, we can express the joint probability by $P(R, D, H) = P(D|H)P(R|H)P(H)$.
Therefore,

$$\begin{aligned} P(R = true|s, D = d) &= \frac{P(R, D)}{P(D)} \\ &= \frac{\sum_{h=1}^{d} P(D|H)P(H)P(R|H)}{\sum_{h=1}^{d} P(D|H)P(H)} \end{aligned}$$

yielding the above equality.

We can show numerically that, for degree $0 < d < 10$, the formula exhibits exponential decay for any values of p_{step} and p_{host}. In fact, we can give the intuitive explanation of the rebel probability under *Butterfly* :

$$P(R = true|s, D = d) = s^{(1-p_{step})d} = e^{(1-p_{step})d(logs)} \tag{4.2}$$

Justification: A rough approximation to the degree of a newcomer node under the *Butterfly* is the number of hosts h it chooses, times the typical length L of each random walk. Since $P(L = l) \sim Geom(l, 1 - p_{step})$ and $E(L) = \frac{1}{1-p_{step}}$, $d \approx h * E(L) = \frac{h}{1-p_{step}}$. Therefore, $P(R = true|s, D = d) = s^h \approx s^{(1-p_{step})d} = e^{(1-p_{step})d(logs)}$.

4.1.3 Empirical validation of Butterfly model

We simulated the model 10 times for $100,000$ nodes, with $p_{host} = 0.5$ and $p_{link} = 0.3$. One run's results are shown in Fig. 4.2. For each run the model exhibited power law in- and out-degree as well as weighted properties. Additionally, it displayed expected properties of the undirected graph: densification and stable NLCC sizes.

For $p_{link} = 0.3$, the densification exponent had range $(1.03, 1.17)$ All occurring values of the exponent are within the range observed in real graphs, and have a least-squares fit of $R^2 > 0.99$ in log-log scales. Moreover, contrary to the *Forest Fire* method [136], our generator is robust, producing realistic-looking results for a wide range of parameter values (plots omitted for space). In contrast, small deviations from recommended parameter values in *Forest Fire* led to unrealistic densification exponents (either 1 or 2), and the model only produced a single GCC.

Not only do we observe *densification*, but also *fortification*—that weights (or multi-edges) will outstrip the number of edges in a superlinear manner. We observed values between 1.1 and 1.2 in our simulations. One plot is shown in Figure 4.3.

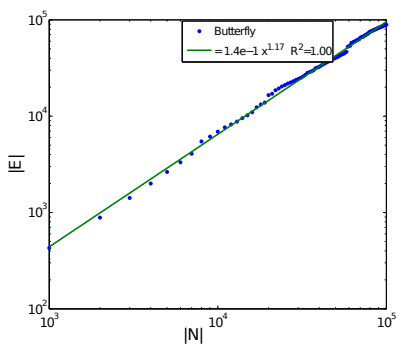
We use the *Butterfly* model to generate a series of edges, and then run analysis on those edges. We find that many properties are replicated under the *Butterfly* model. These plots were generated with $p_{host} = 0.5$; results for other values are similar. Figure 4.4 shows the empirical probability of "Rebelling" given degree. Since under the model the percentage of nodes in the GCC remains constant over time, we modified the p_{host} parameter to show varying behavior. Causing the model to change the percentage over time, without artificially changing the parameters over time, remains in future work; in the meantime modifying p_{host} appears to change this percentage.

4.2 A self-similar generator: RTM

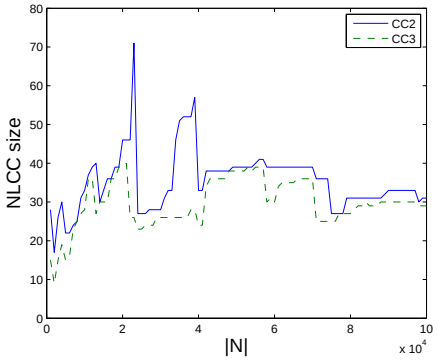
While the agent-based nature of the Butterfly model is intuitive, and its emergent behaviors exciting, making analytical guarantees of its behavior is difficult. We therefore propose an additional model which reproduces properties in a more predictable manner.

At the high level, our idea is to use recursion, in conjunction with tensors (n-dimensional extension of matrices). Recursion and self-similarity naturally lead to modular network behavior ("communities-within-communities"), power laws and bursty traffic. Earlier work used self-similarity to generate static snapshots of unweighted graphs [46].

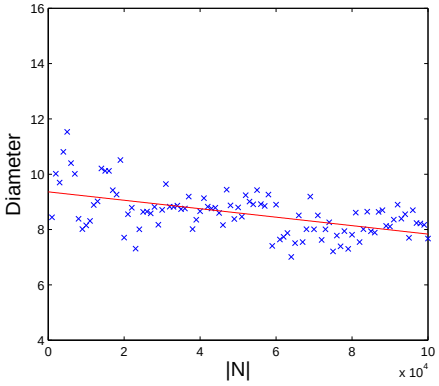
Here, we show how to build a generator that will match all of the properties listed. The idea is to use recursion not only on the adjacency matrix, but also on the *time* dimension. Specifically, we start with a small tensor $\mathcal{I}$ that has 3 sides ('modes'): (a) senders (b) recipients and (c) time. We call the graph represented by a tensor a 't-graph' that evolves over time (See Fig. 4.6(a-b)). Then, we recursively substitute every cell (i, j, t) of the original tensor $\mathcal{I}$, with a copy of itself, and multiply it with the value $a_{i,j,t}$ (See Fig. 4.6(c) for illustration and Definition 1 for full details). Thanks to the self-similarity of the construct, we expect the resulting tensor to have all the properties that we want.



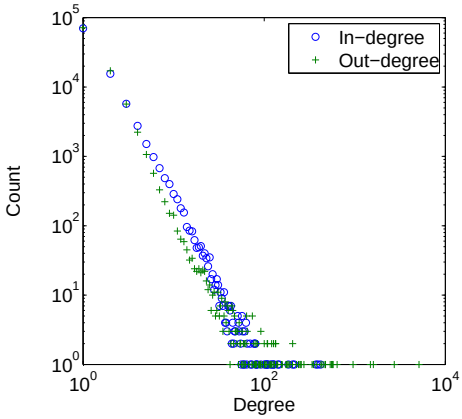
(a) N(t) vs E(t)



(b) CC sizes (log-lin)

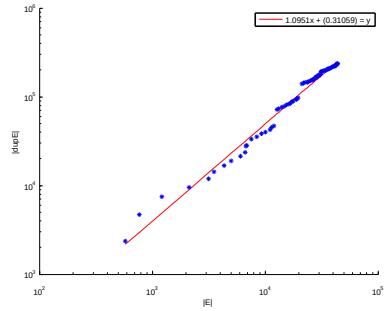


(c) Diameter(N)

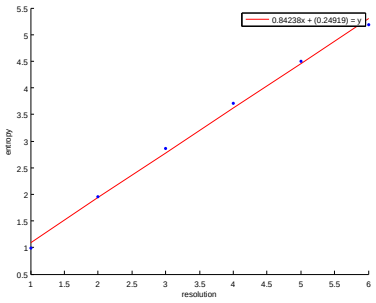


(d) Degree distribution

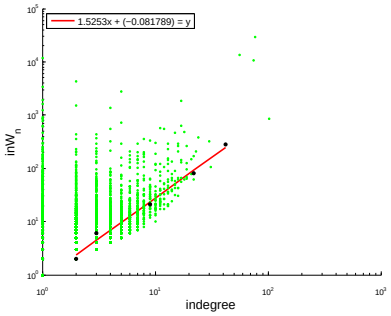
Figure 4.2: Results of proposed *Butterfly* model (p_{host}=0.5, p_{link}=0.3 p_{step} uniform.) (a) Densification power law (exponent: 1.17), (b) Stabilizing NLCCs (between 20 and 50), (c) Small/shrinking diameter, and (d) power laws in the PDF of in- and out- degree distributions.



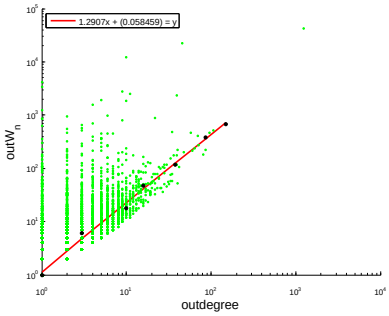
(a) "Fortification," Weights vs Edges



(b) Entropy plot of edge additions



(c) Snapshot, in-degree



(d) Snapshot, out-degree

Figure 4.3: Weighted properties of Butterfly model. (a) Plots the fortification law, of total weight vs. total number of edges, with power law slope of 1.10. (b) shows the entropy plot of edge additions, with bias factor of 0.84, indicating burstiness. (c) and (d) illustrate the Snapshot Power Laws, plotting in- and out- degree vs. in- and out- weight of nodes. The power law fits are 1.52 and 1.29, respectively.

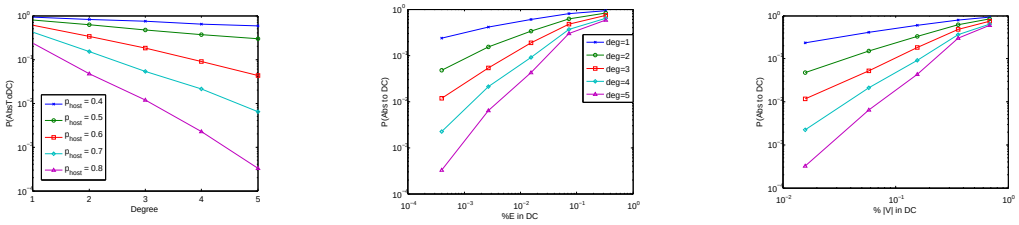


Figure 4.4: Under Butterfly model, (a) Probability of absorption into the DC given degree, for different parameter settings. (b) Probability of absorption into DC given the portion of edges in the DC. Notice the linear drop of the probability as the degree increases, as shown by real data in Figure 3.6.

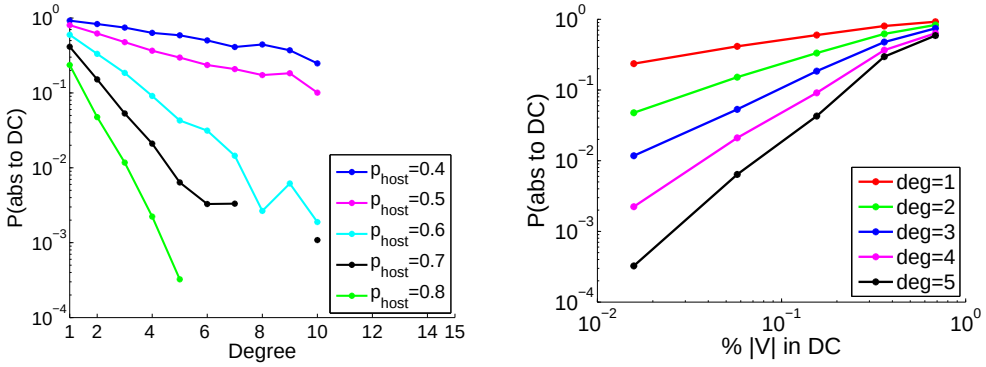
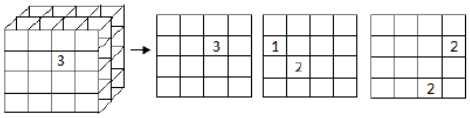


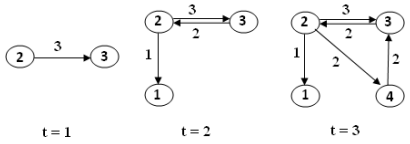
Figure 4.5: Under Butterfly model, (a) P(Absorption to DC) vs. Degree in log-lin scale. Notice the linear drop of the probability as the degree increases. (b) P(Absorption to DC) vs. Portion of Nodes in DC in log-log scale.

Symbol	Description
$\mathcal{A}, \mathcal{B}, \mathcal{C}$	Tensors used to illustrate recursive tensor product
$a_{i,j,k}$	Entry of a tensor
$\mathcal{I}$	Initial tensor in *RTM* model
$\mathcal{G}_{\mathcal{A}}$	t-graph (time-evolving graph) represented by tensor $\mathcal{A}$
$\mathbf{D}_t$	t^{th} slice of final tensor $\mathcal{D}$ in *RTM*
s_t	Total weight of $\mathbf{D}_t$
e_t	Number of edges of $\mathbf{D}_t$
$W_{\mathcal{D}}$	Total weight of a tensor $\mathcal{D}$, or $\sum_t s_t$
$p_{i,r}$	Proportion of weight/edges of the i^{th} (group of) slice(s) at resolution r
$\mathbf{s}_{\mathcal{D},r}$	Temporal profile of $\mathcal{D}$ at resolution r
$\mathbf{p}_{\mathcal{D},r}$	Normalized temporal profile of $\mathcal{D}$ at resolution r

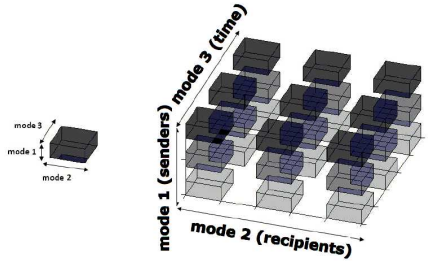
Table 4.1: Notation used for RTM.



(a) $(4 \times 4 \times 3)$ tensor $\rightarrow$ t-slices



(b) corresponding t-graph over time



(c) *RTM* of a tensor by itself

Figure 4.6: (a) An example for the initial tensor $\mathcal{I}$ of size $(4 \times 4 \times 3)$. The 't-slices' represent the changes on the adjacency matrix at every other time step. (b) The corresponding graph represented by the tensor in part (a). It changes according to the 't-slices' over time. (c) An example $(3 \times 3 \times 3)$ tensor $\mathcal{I}$ is given on the left. The recursive tensor product of $\mathcal{I}$ by itself, that is, the resulting $(3^2 \times 3^2 \times 3^2)$ tensor $\mathcal{D} = \mathcal{I} \textcircled{t} \mathcal{I}$ is given on the right.

4.2.1 Definition of proposed Recursive Tensor Model

For the construction, we choose an initial $(N \times N \times \tau)$ tensor $\mathcal{I}$ with nonzero cells (i, j, t) indicating an edge from node i to node j at time tick t. We initialize the cells so that the initial t-graph(t- for time-evolving) $\mathcal{G}_{\mathcal{I}}$ represented by $\mathcal{I}$ looks like a miniature real-world graph. $\mathcal{I}$ is chosen such that at each time tick the WPL is followed: for each link that occurs, enough weight is added so that there is a power law relationship between the total number of edges and total weight of the graph (with to some user-specified α exponent). One method of doing this is to simply use the *Butterfly* generator, which was shown to obey the WPL.

We propose to use *Recursive Tensor Multiplication* to produce a time-evolving graph. Our method extends Kronecker product [1] of two matrices by adding a third 'mode'. Kronecker product of two matrices is defined as follows: Given two matrices **A** and **B** of sizes $(N \times M)$ and $(N' \times M')$, respectively, the Kronecker product of **A** and **B**, namely matrix **C** of dimension $(N * N') \times (M * M')$ is given by

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \quad = \quad \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,M}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,M}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1}\mathbf{B} & a_{N,2}\mathbf{B} & \dots & a_{N,M}\mathbf{B} \end{pmatrix}$$

Definition 1 (Recursive Tensor Multiplication (RTM)) *Given two tensors $\mathcal{A}$ of size $(N \times M \times \tau)$ and $\mathcal{B}$ of size $(N' \times M' \times \tau')$, the Recursive Tensor Multiplication $\mathcal{C}$ of $\mathcal{A}$ and $\mathcal{B}$ is obtained by replacing each cell $a_{i,j,t}$ of tensor $\mathcal{A}$ with $a_{i,j,t} * \mathcal{B}$. The resulting tensor $\mathcal{C}$ is of size $(N * N') \times (M * M') \times (\tau * \tau')$ such that*

$$c_{((i-1)*N+i'),((j-1)*M+j'),((k-1)*\tau+k')} = a_{i,j,k} * b_{i',j',k'}.$$

An example of the Recursive Tensor Multiplication of a $(3 \times 3 \times 3)$ tensor by itself is given in Fig. 4.6(c).

To generate a growing graph over time, we get the *'Recursive Tensor Multiplication'* of the initial $(N \times N \times \tau)$ tensor $\mathcal{I}$ by itself k times as:

$$\mathcal{I}^k = \mathcal{D} = \underbrace{\mathcal{I} \textcircled{t} \mathcal{I} \textcircled{t} \dots \textcircled{t} \mathcal{I}}_{k \text{ times}}$$

and then we take the final tensor $\mathcal{D}$ to represent our data. The data spans τ^k number of time ticks with N^k nodes. At every time step t ($t = \{1, 2, ..., \tau^k\}$), we get the t-slice (See Definition 2 below) $\mathbf{D}_t$ of $\mathcal{D}$, and for each nonzero cell $a_{i,j}$ of $\mathbf{D}_t$, we add an edge between node i and node j with weight $a_{i,j}$. If the edge already exists, we increase the weight $w_{i,j}$ by the same amount.

Definition 2 (t-slice of a tensor $\mathcal{T}$) *Given a tensor $\mathcal{T}$ of size $(N \times M \times \tau)$, a t-slice of $\mathcal{T}$ is a matrix $\boldsymbol{T}_t$ such that*

$$T_t \equiv \mathcal{T}(i, j, t), \quad \forall i, \forall j, \quad 1 \leq i \leq N, 1 \leq j \leq M$$

[1] Unfortunately, Kronecker product **C** of two matrices **A** and **B** is also called Kronecker Tensor multiplication, despite **A**, **B**, **C** being matrices. To disambiguate, we use the name *RTM* where $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ are in fact tensors.

Definition 3 ((Normalized) temporal (t-) profile of $\mathcal{T}$) *Given a tensor $\mathcal{T}$ of size $(N \times M \times \tau)$, let s_t denote the total weight of its t-slice. Then, the t-profile (at resolution 0) of $\mathcal{T}$ is a $(1 \times \tau)$ vector, such that $\mathbf{s}_{\mathcal{T},0} \equiv (s_1, s_2, \ldots, s_\tau)$. The total weight $W_\mathcal{T}$ of $\mathcal{T}$ can be written as $\sum_{t=1}^{\tau} s_t$. Then, the normalized t-profile of $\mathcal{T}$ is a $(1 \times \tau)$ vector, such that $\mathbf{p}_{\mathcal{T},0} \equiv (\frac{s_1}{W_\mathcal{T}}, \frac{s_2}{W_\mathcal{T}}, \ldots, \frac{s_\tau}{W_\mathcal{T}})$.*

Having formally defined the model, we next prove some of the properties that our RTM generator displays.

4.2.2 Analytical validation of RTM

Theorem 3 (Self-similar and Bursty Edge/Weight Additions) *Let edge/weight additions for $\mathcal{I}$ with normalized t-profile $\mathbf{p}_{\mathcal{I},0}$ be self-similar and bursty so that the slope of the entropy plot is*

$$slope = H(\mathbf{p}_{\mathcal{I},0}) = -\sum_{i=1}^{\tau} \mathbf{p}_{\mathcal{I},0}(i) log_2(\mathbf{p}_{\mathcal{I},0}(i)),$$

After k iterations of RTM*, edge/weight arrivals over time for $\mathcal{D}$ are also self-similar and bursty. The slope of the entropy plot over all aggregation levels r of $\mathcal{D}$ is equal to*

$$slope = H(\mathbf{p}_{\mathcal{D},r}) = H(\mathbf{p}_{\mathcal{I},0}), \quad \forall r$$

where $H(\mathbf{p}_{\mathcal{D},r})$ is the slope of the entropy plot at aggregation level r. In other words, the slope does not change with the value of k, that is, burstiness is independent of scale.

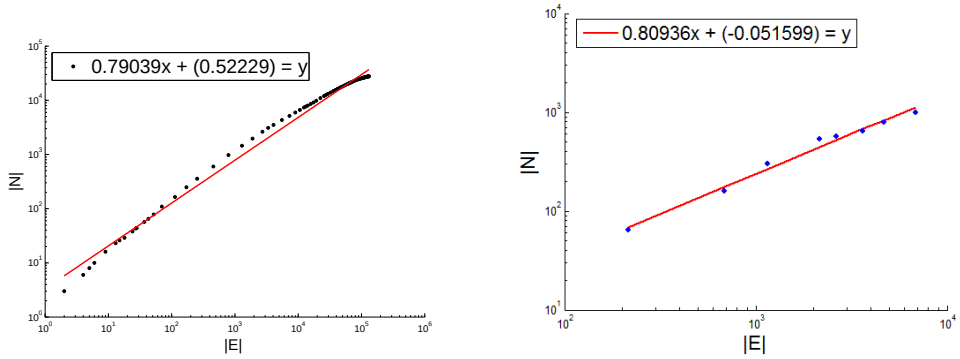
Proof See [5].

Theorem 4 (Weight Power Law (WPL)) *If the initial graph $\mathcal{G}_\mathcal{I}$ exhibits the WPL [153] at* all *time points, that is, number of edges $E(t)$ and total weight $W(t)$ over time follow a power law with exponent α, $\mathcal{G}_\mathcal{D}$ shows the same property at times $1, \tau^1, \tau^2, \ldots, \tau^k$ with exactly the same exponent α.*

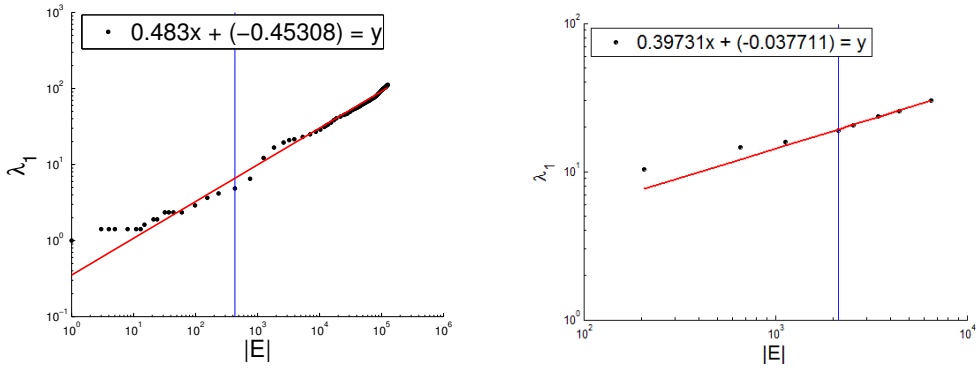
Proof See [5].

4.2.3 Empirical validation of RTM

As a comparison with real-world data, we give the plots showing reported laws for *BlogNet* and the plots our model generated for $N = 10$, $\tau = 2$ and $k = 3$ in Figure 4.7 and 4.8. In particular, we show (a) the Densification Power Law (DPL) and (b) the λ_1 Power Law (LPL) for unweighted graphs; and (a) the Weight Power Law (WPL); (b) the $\lambda_{1,w}$ Power Law (LWPL) and finally, (c) the Edge Weight Power Law (EWPL) for weighted graphs. Note that characteristics matched by *RTM* include both those from previous work as well as additional patterns discovered in this work. Interestingly, for the EWPL, we notice the same cutoff as observed in Figure 3.9 with the political campaigns data, this time around edge weight of 3: for small-weight edges, the adjacent nodes tended to be higher weight. This may also be attributed to the fact that few edges were of low-weight (5 percent were weight 3 or less.) Other desired characteristics such as small and shrinking diameter, the gelling point, etc. are also matched.

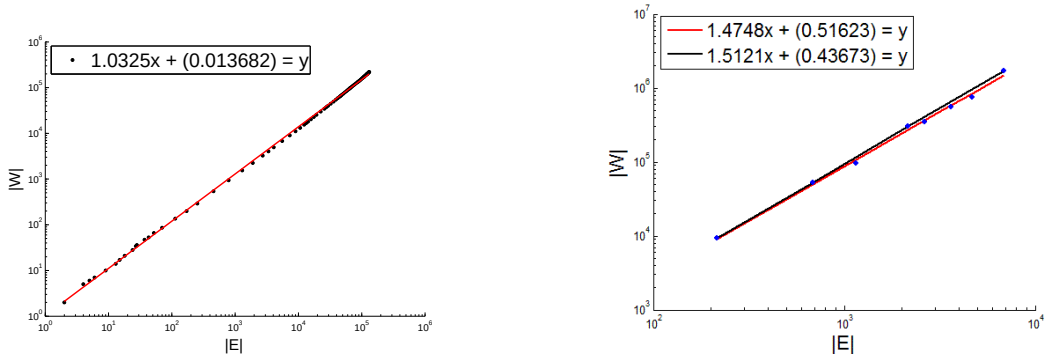


(a) Densification Power Law (DPL)

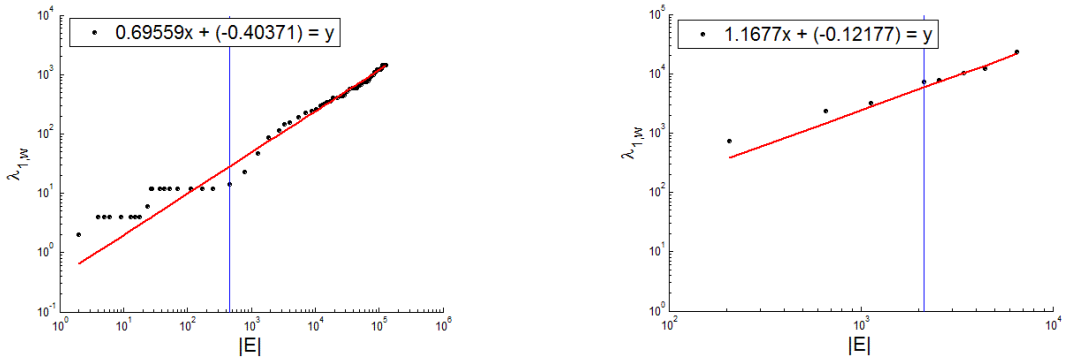


(b) λ_1 Power Law (LPL)

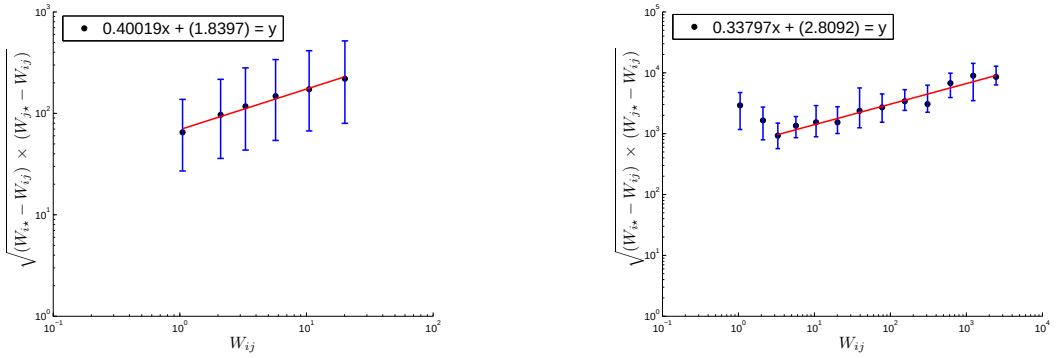
Figure 4.7: Plots showing unweighted laws that real-world graphs obey for *BlogNet* on the left and for our *RTM* generator on the right. Notice we reproduce the superlinear behavior between edges and nodes (more nodes implies even more edges), as well as the principal eigenvalue increasing in a power law over time.



(a) "Fortification" Weight Power Law (WPL)



(b) $\lambda_{1,w}$ Power Law (LWPL)



(c) Edge Weight Power Law (EWPL)

Figure 4.8: Plots showing weighted laws that real-world graphs obey for *BlogNet* on the left and for our *RTM* generator on the right. We successfully reproduce *fortification*, where superlinearly more weight is added per edge. We also reproduce the power-law increasing weighted principal component, and the "edge weights power law" for the proportion of weight given to an edge.

4.3 Discussion

In addition to providing answers about real graphs, this work introduces some interesting questions. The Butterfly model lends itself to some potential extensions in order to mimic the observations on weighted graphs. Currently a node does not increase its out-degree after it finishes its initial series of random walks, so weight between two existing edges cannot be added at later time. One might devise a scheme of "wake-ups" to allow this behavior. The precise choice of parameters to accommodate this (whether constant or randomized, as in p_{step}) is left for future work.

We validated *Butterfly* and *RTM* by showing that they obey properties observed in real graphs. A better fit to observed properties indicates a more realistic model. However, an open question is how to develop a more cohesive idea of graph properties and generative models, in order to classify and compare different generators.

It could be argued that certain properties in a network are more informative about the nature of a graph than others, and therefore more important for a generator to produce. It is probably the case that some properties follow from others: for instance, densification, gelling point, and shrinking diameters seem to be closely related. Small-world properties and heavy-tailed degree distributions also seem closely related, as a hub structure in a graph leads to small diameters. Analytically proving that some properties follow from others would be valuable contributions. Another direction would be to prove that different properties are *independent*: certainly, if one can generate a network for which one property is present and another is not. Both of these directions could be key steps in determining a sort of "network information gain" measurement to prioritize certain properties over others for a generator.

There are several other open questions with respect to the representation of these networks. The networks studied typically represent *one kind* of link, and inferring information about nodes using only that edge information can sometimes be problematic. For example, within an corporation, an email network would not necessarily represent all links, let alone the strength of ties. Other edge types, such as geographic proximity and other communication (phone calls, scheduled meetings) would add additional information about the nature of links between people. How to analyze multiple sources of data, whether multiple edge types, node features, or other network information, remains an important area of study.

4.4 Summary of models and contributions

The *Butterfly* model and the observation of constant NLCCs sheds light upon a recent, counter-intuitive discovery [141]: in several real graphs, it is shown that the GCC has *no* good cuts, so graph partitioning and graph clustering algorithms cannot help to identify communities. Our observations are useful for the monitoring of growing networks, like e-communities (say, in Yahoo, LinkedIn, Facebook, e-bay). The observations can help us spot 'anomalous' communities, like communities of potential abusers (spammers, fraudsters), or communities that are not coherent enough, and thus will probably disintegrate, if left alone. For instance, we should be surprised to find a large disconnected component in a real network that never joins to the GCC, and therefore may want to investigate it. The generative model produced is useful for what-if scenarios, to de-

Model \ Property	*Established*					*New unweighted*				*New weighted*					*Other*	
	Heavy-tailed degree distribution	Small, shrinking diameter	Densification Power Law	Triangle Power Law	Eigenvalue Power Law	Small disconnected components	Stable component graph fractal dimension	Exponential "rebel" probability	λ_1 Power Law	Weight Power Law	Edge Weights Power Law	Snapshot Power Law	Bursty weight additions	$\lambda_{1,w}$ Power Law	Agent-based	Simple to analyze
Butterfly	✓	✓	✓	?	?	✓	?	✓	✓	✓	✓	✓	✓	X	✓	-
Recursive Tensor	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-	✓

Table 4.2: A summary of properties exhibited by various models.

termine how a certain community may look one year from now, or perhaps how quickly a graph will become "gelled."

In summary, are contributions are as follows:

- We have proposed two models for generating the topology of real networks, the *Butterfly Model* and *Recursive Tensor Model*.
- We have shown that both follow many of the properties that are already established, as well as new ones, as summarized in Table 4.2.
- We have, for some properties *proved* that the models will generate such properties (Theorem 1, page 38; Theorem 2, page 39; Theorem 3, page 46; Theorem 4, page 46).

Part II

Conversation patterns in networks

Chapter 5

Preliminaries

Having explored some of the global properties of networks, we now aim to look deeper into the communication patterns that help form the links. To this end, we study *cascades*: tree-like structures formed by interactions between entities in a network. A cascade is typically formed by a root node (perhaps a blog post or post to a message board), and built up by replies in a tree-like fashion. Influence is implied in a cascade, as one node may invoke a response from a new node. (See cascades illustrated in Figures 5.1 and 5.2.) We say "tree-like" because, while most cascades are trees, a cascade need not be strictly a tree: a node within a cascade may have many parents. However, we do define cascades to have one root node, as we will discuss later.

The easiest way to visualize a cascade is to consider a threaded email. It begins with the first email, which forms the root of the tree, and continues as others respond. Threads in a message board-based online group are very easy to trace. However, cascades need not form within one centralized community, but may occur in other domains such as blogs. One blog may begin a "root" post, and another blog may respond by composing a new post and hyperlinking to the first post. In that context, a cascade is a representation of an "infection pattern," where entities in a network topology are activated one by one in a traceable path.

Thus, we explore two main modes of communication: blogs and online groups (where members of a group post to a message board or mailing list).

Our main questions include:

- What can we learn about temporal patterns of communication? After a topic becomes popular, how does interest die off: linearly, or exponentially? Is there periodicity?
- What are the topological properties of cascades? Do graphs of information cascades have common shapes? What are their properties? What are characteristic in-link patterns for different nodes in a cascade? What can we say about the size distribution of cascades?
- Can we use knowledge about cascades formed within a community to characterize that community? What similarities and differences can be observed between different groups?
- How does linking behavior vary between online communities, such as Usenet vs. blogs? How do the subjects of attention, coverage of subjects, and timeliness differ?
- How does information diffuse between communities? What tools can we use to measure this diffusion?
- What generative models are most realistic for reproducing cascading behavior?

In the next few sections we will provide some preliminary definitions, survey related work,

Figure 5.1: An example of a thread in an online group, and the corresponding cascade (with authors in color).

and describe the data we will analyze.

5.1 Definitions

We next address some of the major foundations of the work presented in this part.

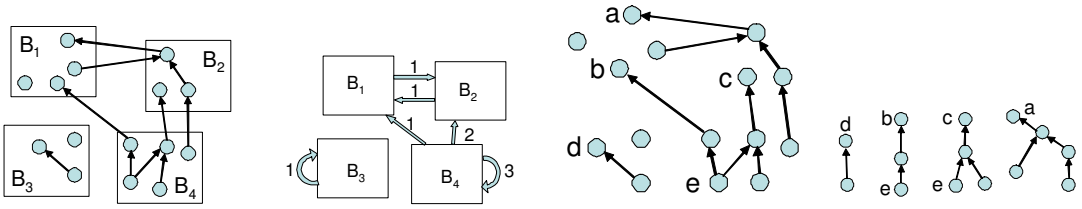
5.1.1 Cascades in online networks

We describe cascades in the context of blogs and in the context of groups. In groups, extracting cascades (known in this context as threads—we will use the terms interchangeably) is simple: the first post for a given subject is the root of the tree, and replies follow referencing that post. An example is shown in Figure 5.1.

Extracting cascades in blogs is a more complex process. We model two graph structures emergent from links in the blogosphere, which we call the *BlogNet* and the *PostNet*. Figure 5.2 illustrates these structures. The blogosphere is composed of blogs, and each blog has a set of posts. Hyperlinks occur from one post to another. From the blogosphere (a), we obtain the *BlogNet* (b) by collapsing all links between blog posts into weighted edges between blogs. A directed blog-to-blog edge is weighted with the total number of links occurring between posts in source blog pointing to posts in destination blog. From the *BlogNet* we can infer a social network structure, under the assumption that blogs that are "friends" link each other often.

In contrast, to obtain the *PostNet* (c), we ignore the posts' parent blogs and focus on the link structure. Associated with each post is also the time of the post, so we label the edges in *PostNet* with the time difference Δ between the source and the destination posts. Let t_u and t_v denote post times of posts u and v, where u links to v, then the link time $\Delta = t_u - t_v$. Note $\Delta > 0$, since a post can not link into the future and there are no self-edges.

From the *PostNet*, we can extract cascades (Figure 5.2(d)). A cascade has a single starting post called the *cascade initiator* with no out-links to other posts (e.g. nodes a, b, c, d in Figure 5.2(c)). Posts then join the cascade by linking to the initiator, and subsequently new posts join by linking to members within the cascade, where the links obey time order ($\Delta > 0$). Since



(a) Blogosphere (b) Blog network (c) Post network (d) Cascades

Figure 5.2: A graphical representation of the blogosphere (a). Squares represent blogs and circles blog posts. Each post belongs to a blog, and can contain hyper-links to other posts and resources on the web. We create two networks: a blog network (b) of citations (links) between blogs, and a post network (c) with time stamped links between blog posts. (d) are cascades extracted from (c). Cascades represent the flow of information through nodes in the network. To extract a cascade we begin with an initiator with no out-links to other posts, then add nodes with edges linking to the initiator, and subsequently nodes that link to any other nodes in the cascade.

a link points from the follow-up post to the existing (older) post, influence propagates following the reverse direction of the edges.

We define a *non-trivial* cascade to be a cascade containing at least two posts; a *trivial cascade* is an isolated post. (Figure 5.2(d) omits trivial cascades.) Non-trivial cascades form two main shapes, which we refer to as *stars* and *chains*. A star occurs when a single post is linked by several other posts, but the links do not propagate further. This produces a wide, shallow tree. Conversely, a chain occurs when a root is linked by a single post, which in turn is linked by another post. This creates a deep tree that has a small average branching factor. As we will later see, most cascades are somewhere between these two extreme points. Occasionally, separate cascades might be joined by a single post. For instance, a post may summarize a set of topics, or focus on a certain topic and provide links to different sources that are members of independent cascades. The post merging the cascades is called a *connector node*. Node e in Figure 5.2(c) is a connector node. It appears in two cascades by connecting cascades starting at nodes b and c. Connectors do not appear in groups data (as we observe them), as all messages form disjoint trees: detecting when a post has referenced multiple posts requires extensive preprocessing.

Cascades have *depth*, and posts within a cascade may have depth upward and downward. Posts in a cascade also have *conversation mass*, defined as follows: Let T be the set of all cascades, B be the set of all bloggers, and P be the set of all posts. Let $T(b)$ be the subset of all conversations in which blogger b (in B) contributes at least one post. Let $t \in T$ be a cascade. $t(p)$, for $t \in T$ and $p \in P$ is the subtree of the conversation t starting at post p. Define the conversation mass generated by post p as the number of posts in $t(p)$. Define the conversation mass for blogger B as the sum of the conversation mass of $t(p)$ over all t in $T(B)$, where p is the first post in t authored by blogger b. In other words, the conversation mass for a blogger equals: the total number of posts in all conversation trees below the point in which the blogger contributed, summed over all conversation trees in which the blogger appears.

5.1.2 Measuring self-similarity using power laws and burstiness

We covered these subjects in Chapter 2 (page 13) in detail. We will briefly revisit the topics of burstiness and power laws in terms of human behavior, in order to avoid dependency.

Self-similar behavior occurs often in nature. One example of self-similarity is the power law, a *scale-free* distribution. As a scale-free network grows, it will exhibit behavior similar to fractals, where the degree distribution of nodes maintains a distribution with constant slope, regardless of scale [17].

Two variables x and y are related by a power law when:

$$y(x) = Ax^{\alpha} \tag{5.1}$$

where A is positive and α is a negative constant. The constant α is often called the power law exponent.

A random variable is distributed according to a power law when the probability density function (pdf) is given by:

$$p(x) = Ax^{\alpha}, \quad \alpha < -1, x \geq x_{min} \tag{5.2}$$

The extra $\alpha < -1$ requirement ensures that $p(x)$ can be normalized. Power laws with $-1 < \alpha < 0$ rarely occur in nature, if ever [169].

Much human behavior is self-similar in time sequence, or bursty. Among the many methods that measure self-similarity (Hurst exponent, etc. [188]), we choose the *entropy plot* [204], which plots the entropy $H(r)$ versus the resolution r of an activity sequence $A = a_1, a_2, ..., a_T, a_t \in \mathbb{Z}*$. The resolution is the scale, that is, at resolution r, we divide our time interval into 2^r equal sub-intervals, sum the activity $A(k)$ in each sub-interval k ($k = 1 \ldots 2^r$). We then normalize into fractions $p_k = (A(k)/\sum_{j=1\ldots 2^r} A(k))$, and compute the Shannon entropy of the sequence p_k: $H(r) = -\sum_k p_k \log_2 p_k$. If the plot $H(r)$ is linear in some range of resolutions, the corresponding time sequence is said to be *fractal* in that range, and the slope of the plot s. Notice that a uniform activity distribution yields s=1; a lower value of s corresponds to a more bursty sequence like a Cantor dust [188], with a single burst having the lowest s=0: the intrinsic dimension of a point. The so-called 'b-model' [204], generates such self-similar traffic, suggesting that it may be a good model for blog traffic (as we will explore later on). Figure 5.3 shows bursty behavior and the corresponding entropy plot, for review.

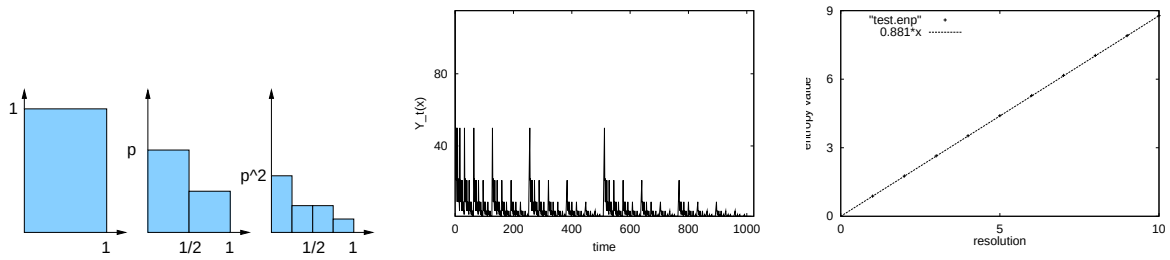
5.2 Related Work

We next survey other case studies in online communities, as well as network diffusion as applied to marketing and epidemiology.

5.2.1 Studies of online communities

Blogs

Related work on blogs has modeled link behavior in large-scale on-line data [1, 3, 128], and showed that information often propagates between blogs. Work on information diffusion based



(a) generation (b) synthetic data (c) entropy plot

Figure 5.3: Illustration of the *b-model*: *(*a*)* the recursive 80-20 procedure in its first three iterations *(*b*)* the generated synthetic activity *(e.g., number of posts, over time*) *(*c*)* its *entropy plot (*entropy versus resolution - see text*)* Because the synthetic input traffic is self-similar, the entropy plot is linear, that is, *scale free*. Its slope is 0.881, much different than 1.0, which would be the uniform distribution *(*50-50*)*

on topics [95] showed that for some topics, their popularity remains constant in time ("chatter") while for other topics the popularity is more volatile ("spikes"). Tangentially related work examined the relationship between blogs and mainstream media, setting up a system that identifies emotionally charged news articles for communities of different political orientation [78].

There has also been much work on the community structure of the blogosphere. The authors of [1] showed that sub-communities may assume different characteristics: in particular, for blogs during the 2004 election the liberal community was far less connected than the conservative one. In a related social network, the Usenet, Fiore *et al.* assigned roles that different users played based on a survey, and were able to identify some common network characteristics of these different roles [73]. Kumar et al. [128] analyze community-level behavior as inferred from blog-rolls, which indicate permanent links between "friend" blogs. In follow-up work, Kumar et. al. [129] studied several topological properties of link graphs in communities, discovering that "star" topologies are frequent.

Usenet and message boards

The Netscan project at Microsoft conducted a very thorough study of Usenet discussion patterns. In Turner et al., authors depicted the hierarchy of newsgroups, and changes between 2000 and 2004. They showed that between 2000 and 2004 most social and political groups declined in activity. In Fisher et al, authors studied the *social roles* of Usenet authors [74], paying attention to the degree distribution and determining whether authors in different newsgroups participated most in "questions," "answers," or "discussion." They suggested that the political group *alt.politics* had an "exclusive clique" in the core, where members get many replies to their posts but outsiders get few.

There have been a number of studies on online groups aside from Usenet. One study focused on online bulletin boards in a university, analyzing reply and membership networks, paying particular attention to hub members connecting the communities [88]. Other work has focused on rebuilding thread structures in bulletin board conversations that are not pre-labeled [207]. Blog comments can also serve as forums for a specific topic, and can be used to assess controversy

of blog posts [159]. A similar study focused on the Slashdot.org community, suggesting using a controversy measure based on the patterns in the threaded network [92]. Backstrom et al. studied Yahoo! Groups data, defining "thriving" groups and tracking engagement of core users in groups [14]; see also [51]. Leskovec et al. studied the edge arrivals of different online networks, proposing a generative model [141].

5.2.2 Cascades and viral marketing

In a broad sense, information cascades are phenomena in which an action or idea becomes widely adopted due to the influence of others, typically, neighbors in some network [32, 90, 93]. Cascades on random graphs using a threshold model have been theoretically analyzed [209]. Empirical analysis of the topological patterns of cascades in the context of a large product recommendation network is in [139] and [137].

Liben-Nowell and Kleinberg, studied the structure of chain letter cascades [142], and showed that the structure was characterized by a deep tree-like pattern, and proposed a probabilistic model to generate such trees. Golub and Jackson [91] built on this to show that a basic branching process model combined with the selection bias of observing only large diffusion can explain the results in [142].

There has also been some exploration into the dynamic processes of conversation and information propagation. Barabasi [16] postulates that the bursty nature of human behavior is a consequence of a queuing process and uses it to explain the heavy-tail activity patterns in e-mail communications; Vazquez et al. [62, 203] further explore this model.

Leskovec, Backstrom, and Kleinberg [133] considered the propagation of "memes" across the Web, in the context of news cycle. In course of studying this problem, they consider a model where they combine recency and the preferential attachment process. We will refocus these ideas to the graph generation process.

Viral marketing is a key application to studying interactions in networks, and there has been a great body of work in this domain. Kempe, Kleinberg, and Tardos [117] focused on finding the most influential nodes in a network, under the threshold-model of influence. Richardson and Domingos [185] introduced the concept of *network value* of a customer, which is valuable for viral marketing.

Rogers [186] studied how people adopt a new product: New adopters follow a Bell curve over time, therefore saturation follows an S-curve. There are the "innovators" at the beginning of the curve, who first adopt a product, followed by "early adopters." The majority of the people adopt the product after that, while there are a few laggards at the end of the cycle. The Bass Model for diffusion [21] fits this data to a model. The Bass model includes parameters for pricing and advertising effects, and matches product sales data for a wide variety of products.

Other models for product penetration and adoption include the Dirichlet model [202] (a model based on several consumer-based parameters); the "two-step flow" model of Lazarsfield and Katz [116], which includes both marketing effects and later agency effects; the trickle-down effect where products slowly become available to the less-wealthy masses [193]; and "crossing the chasm," which is based on the Bass model [164].

Other research has focused on network effects in particular for product adoption. Godes and Mayzlin studied the effectiveness of word-of-mouth communication about products, finding

that for a product with a low awareness level, word-of-mouth is particularly effective among not highly loyal customers, but exogeneously (firm)-created word-of-mouth is most effective among already loyal customers [86]. The authors also showed that online conversations are a successful medium in which to measure word-of-mouth communication about products [85]. Chevalier and Mayzlin showed that on-line reviews also affect purchasing rates [53].

However, without directly observing word-of-mouth communication, a key question remains in causality. Neighbors in a network are often similar, therefore have a propensity to adopt similar products even without word-of-mouth communication. Hill, Provost, and Volinsky compared adoption patterns of different products in a network, one that lends itself to word-of-mouth communication (VoIP service), and one that did not (pricing plans), and found that they had different adoption patterns across edges [103]. Using causal methods as described in [109] may also be useful in distinguishing between node similarity and true propagation.

5.2.3 Epidemiological Models and virus propagation

Epidemiology is a closely related area of study, with vast literature on models, immunization policies, and epidemic thresholds. Much of the theory may be found in [15]; see [101] for a recent survey. One of the main models is the *SIS* or flu-like model. SIS stands for *susceptible, infectious, susceptible*, and a node can become re-infected multiple times. The second major model is the *SIR* (*susceptible, infectious, removed*), and the node acquires immunity for life and is thus removed (or recovered), similar to chicken pox. Other extensions include "infection delay" and "user vigilance," as discussed in [205].

These ideas have been applied to real networks, under different assumptions. These include the Kephart-White model [118, 119], the Mean Field Assumption model [179, 180, 181, 182], correlated networks [33], and particle systems [97, 143]. A major topic of interest is the *epidemic threshold* of a graph, that is, the parameter τ of a virus at which a virus becomes endemic to a network, which is investigated in [44] and [206].

5.3 Data

We focus our cascade studies to the online domain. Time plays an important role in the growth of online networks, and links have a more or less uniform meaning: a link from a node u to another v means that the message corresponding to u is in reply to v. Cascades are therefore very easy to trace.

Our data sets fall into two categories: blog-like data and group-like data. Blog-like data includes data gathered from blogs and micro-blogs (Twitter), while group-like data includes that of Usenet and Yahoo! Groups, groups that essentially function as mailing lists. We can extract cascades from both kinds; however, the interfaces are different enough that we do not necessarily expect cascades to follow the same patterns in both data sets, due to the different interfaces. Online groups are centralized, and all content is easily accessed by any user; however, in blogs and micro-blogs, each user tends to only follow a few other users.

Dataset	Messages ($\times 10^6$)	Cascades ($\times 10^6$)	Users ($\times 10^6$)
Blogs	2.20	2.09	0.045
TWITTER	69.94	36.24	5.023
USENET	22.61	3.896	1.659
Y!GROUPS	5.869	1.558	0.690

Table 5.1: Synopsis of the datasets.

5.3.1 Blogs

Data description

We extracted our blog data set from a whitelist of blogs from August and September 2005 [84]. Our goal in this part is to study temporal and topological characteristics of information propagation, so we biased our dataset towards the more active part of the blogosphere. We collected our dataset using the following procedure. We started with a list of the most-cited blog posts in August 2005. For all posts we traversed the full conversation tree forward and backward following post's in- and out-links. For practical reasons we limited the depth of such conversation trees to 100 and the maximum number of links followed from a single post to 500. This process gave us a set of posts participating in conversations. From the posts we extracted a list of all blogs. This gave us a set of about $45,000$ active blogs. We went back to the original dataset and extracted all posts coming from this set of active blogs.

This process produced a dataset of $2,422,704$ posts from $44,362$ blogs gathered over a two-month period from beginning of August to end of September 2005. There are the total of $4,970,687$ links in the dataset out of which $245,404$ are among the posts of our dataset and the rest point to other resources (e.g. images, press, news, web-pages). For each post in the dataset we have the following information: unique Post ID, the URL of the parent blog, Permalink of the post, Date of the post, post content (html), and a list of all links that occur in the post's content. Notice these posts are not a random sample of all posts over the two month period, but are biased towards active blogs participating in conversations (by linking to other posts/blogs).

In Figure 5.4 we plot the number of posts per day over the span of our dataset (note weekly periodicity). Notice that our dataset has no "missing past" problem, i.e. the starting points of conversation are not missing due to the beginning of data collection, since we followed the conversation all the way to its starting point and thus obtained complete conversations. The posts span the period from July to September 2005 (90 days), while the majority of the data comes from August and September. The July posts in the dataset are parts of conversations that were still active in August and September.

Data preparation and cleaning

As we described previously, we represent blog data as a cluster graph (Figure 5.2(a)) where clusters correspond to blogs, nodes in the cluster are posts from the blog, and hyper-links between posts in the dataset are represented as directed edges. Before analysis, we cleaned the data to most clearly represent the structures of interest.

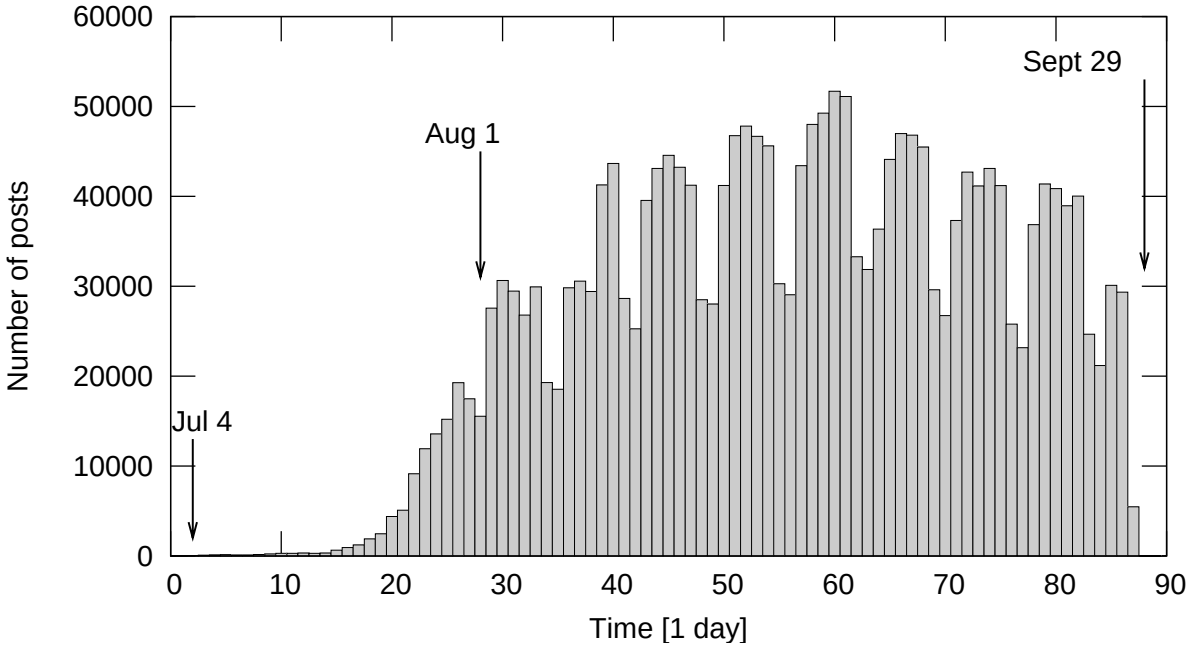


Figure 5.4: Number of posts by day over the three-month period.

Only consider out-links to posts in the dataset. We removed links that point to posts outside our dataset or other resources on the web (images, movies, other web-pages). The major reason for this is that we only have time-stamps for the posts in the dataset while we know nothing about creation time of URLs outside the dataset, and thus we cannot consider these links in our temporal analysis.

Use time resolution of one day. While posts in blogspace are often labeled with complete time-stamps, many posts in our dataset do not have a specific time stamp but only the date is known. Additionally, there are challenges in using time stamps to analyze emergent behaviors on an hourly basis, because posts are written in different time zones, and we do not normalize for this. Using a coarser resolution of one day serves to reduce the time zone effects. Thus, in our analysis the time differences are aggregated into 24-hour bins.

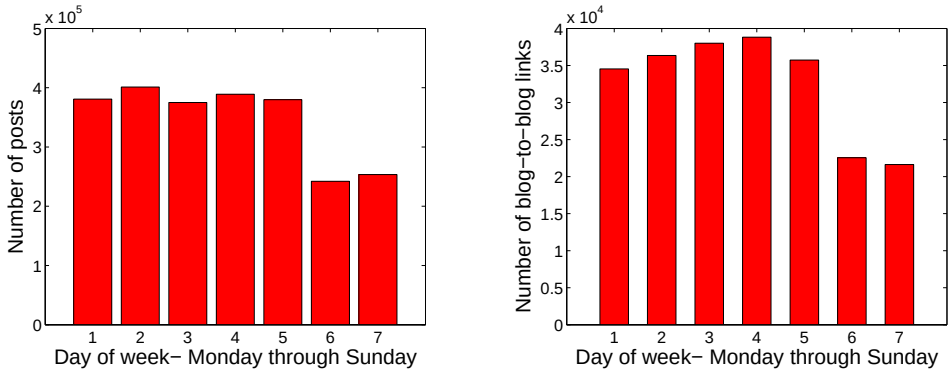
Remove edges pointing into the future. Since a post cannot link to another post that has not yet been written, we remove all edges pointing into the future. The cause may be human error, post update, an intentional back-post, or time zone effects; in any case, such links do not represent information diffusion.

Remove self edges. Again, self edges do not represent information diffusion. However, we do allow a post to link to another post in the same blog.

Properties

We next describe a few contributions in analyzing the data. They do not fit into cascade properties, but are worth mentioning for the sake of understanding the data.

Weekly periodicity.



(a) Posts (b) Blog-to-Blog links

Figure 5.5: Activity counts (number of posts and number of links) per day of week, from Monday to Sunday, summed over entire dataset.

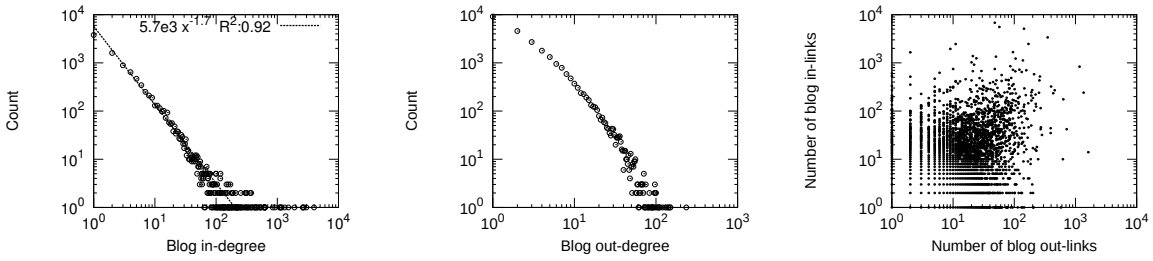


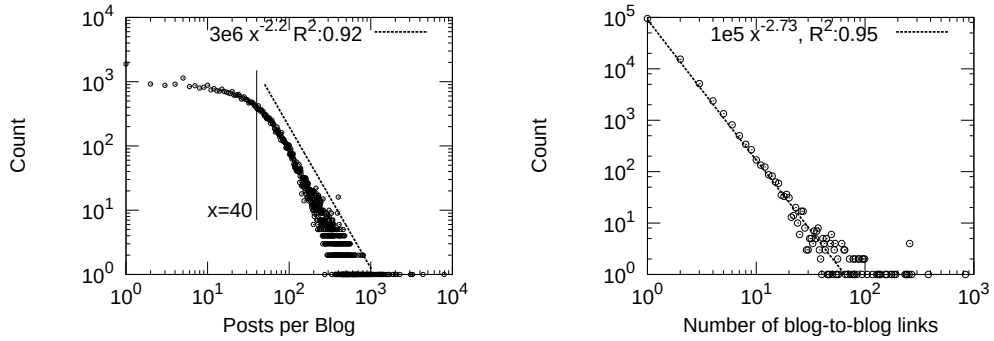
Figure 5.6: In- and out-degree distributions of the *BlogNet*, and the scatter plot of the number of in- and out-links of the blogs.

Traffic in blogs is not uniform; therefore, we consider traffic patterns when analyzing influence in the temporal sense. As Figure 5.4 illustrates, there is a seven-day periodicity. Further exploring the weekly patterns, Figure 5.5 shows the number of posts and the number of blog-to-blog links for different days of the week, aggregated over the entire dataset. Posting and blog-to-blog linking patterns tend to have a *weekend effect* of sharply dropping off at weekends.

***BlogNet* topology.**

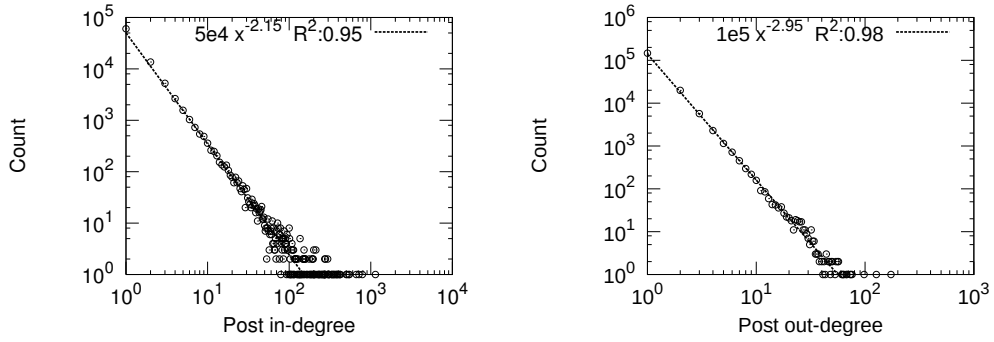
The first graph we consider is the *BlogNet*. As illustrated in Figure 5.2(b), every node represents a blog and there is a weighted directed edge between blogs u and v, where the weight of the edge corresponds to the number of posts from blog u linking to posts at blog v. The network contains $44,356$ nodes and $122,153$ edges. The sum of all edge weights is the number of all post to post links $(245,404)$. Connectivity-wise, half of the blogs belong to the largest connected component and the other half are isolated blogs.

We show the in- and out-degree distribution in Figure 5.6. Notice they both follow a heavy-tailed distribution. The in-degree distribution has a very shallow power-law exponent of -1.7, which suggests strong rich-get-richer phenomena. One would expect that popular active blogs that receive lots of in-links also sprout many out-links. Intuitively, the attention (number of in-links) a blog gets should be correlated with its activity (number of out-links). This does not seem



(a) (b)

Figure 5.7: Distribution of the number of posts per blog (a); Distribution of the number of blog-to-blog links, i.e. the distribution over the *BlogNet* edge weights (b).



(a) (b)

Figure 5.8: *PostNet* in- and out-degree distribution.

to be the case. The correlation coefficient between blog's number of in- and out-links is only 0.16, and the scatter plot in Figure 5.6 suggests the same.

The number of posts per blog, as shown in Figure 5.7(a), follows a heavy-tailed distribution. The deficit of blogs with low number of posts and the knee at around 40 posts per blog can be explained by the fact that we are using a dataset biased towards active blogs. However, our biased sample of the blogs still maintains the power law in the number of blog-to-blog links (edge weights of the *BlogNet*) as shown in 5.7(b). The power-law exponent is -2.7.

***PostNet* topology.**

In contrast to *BlogNet* the *PostNet* is very sparsely connected. It contains 2.2 million nodes and only $205,000$ edges. 98% of the posts are isolated, and the largest connected component accounts for $106,000$ nodes, while the second largest has only 153 nodes. Figure 5.8 shows the in- and out-degree distributions of the *PostNet* which follow a power law with exponents -2.1 and -2.9, respectively.

5.3.2 Discussion groups

We describe the two sources of groups data that will be used in our study, namely, messages from a set of Usenet groups and messages from a set of public Yahoo! groups.

Each dataset consists of records, where each record has the ID of message, the ID of its parent message (if applicable), the author of the message, and a timestamp. Notice that all the three datasets enable conversations among its users, i.e., messages can be posted in response to earlier messages.

Usenet

One of the first online forums, Usenet originated in 1979, preceding Web 2.0 by decades. While overall its activity is declining, Usenet is still in use and there are many very active communities [201], making it an excellent resource for social network analysis. We collected data from nearly 200 newsgroups with posts between 2004 and 2008, using a subscription service. In the interests of capturing a representative subset of data relating to political discussions, we selected all newsgroups available with the substring "polit" in the name[1]. We chose to focus on political newsgroups because politics is a topic that permeates most cultures, and can be used to compare cross-cultural groups. Indeed, there were many different regions of the world represented, including some groups for specific U.S. states. Around 70 were `alt.politics.*` subgroups, on topics such as political parties or regions, with another 20 topical groups under `talk.politics.*`. Others were devoted to regional discussion, either for local areas or topics. 22 were local United States (`va.politics`, `seattle.politics`, etc.), 6 were local Canadian groups, (`edm.politics`, `bc.politics`, etc). 3 from `de`, 4 from `dk`, 3 from `es`, 7 from `it`, 4 from `tw`, and 9 from `uk`. In addition there were several other international domains with one or two groups represented. Of these newsgroups, there were 19.6 million unique articles, and 6.2 million of these were cross-posted to multiple groups in the data set.

We also gathered a smaller but broader sample. We sampled Usenet based on groups posted to in early January 2010, according to `http://newsadmin.com/top100tmsgs.asp`. For a complete list of the groups crawled, refer to `http://www.cs.cmu.edu/~mmcgloho/pubs/groupthreads-list.txt`. This gave us a broad sample of newsgroups, including some on political discussion (`alt.politics`, `it.politica`), recreational activities and hobbies (`rec.outdoors.rv-travel`, `rec.music.beatles`), and general news or ads (`news.lists.filters`, `alt.marketplace.online.ebay`). This crawl produced around 10 million posts in total. Most groups had between 1,000 and 5,000 users, with some as few as 20.

Yahoo! groups.

Yahoo! groups is a popular online groups application. We chose public groups from Yahoo that had the following characteristics: (a) moderated, (b) active (not deleted or suspended), (c) at least ten messages, and (d) at least ten distinct users. This resulted in 13,102 groups in the dataset with over 14.9 million posts. The groups in our data included ones such as `WrestlingGear`, `cookbook-reviews`, `IndianaSPCA`, `welcometomorocco`, `neurosurgeonsclub`,

[1]While a number of other sampling methods were considered, we chose this one for simplicity; due to the structured nature of Usenet, this was a reasonable method. The complete list of newsgroups used may be found at `www.cs.cmu.edu/~mmcgloho/data/usenet.html`.

etc. These groups covered a broad set of topics and interests. Most groups contain 500 to 5,000 users, with some as few as ten (our minimum threshold for including in the dataset). The data was collected in January 2010.

Thread and author network construction

One method of looking at patterns of information diffusion is extracting *threads*, conversation trees of replies. The algorithm for thread induction in message boards is simple. Each post is labeled with a *message-ID* and *references*. References may be numerous: for our purposes we take the last one on the list, as it is the most recent and therefore the direct reply. Other references already occur further up in the tree. This forms several cascades, each one being one thread. Each message has at most one parent, and of the entire network of posts each connected component represents a thread, which may stretch across several groups (thanks to cross-posts).

From the post-reply trees one can induce a social network topology of authors. Every message has an e-mail address to identify the message author. The resultant social network is weighted for multiple links between two authors. This is similar in spirit to inducing a network of blogs based on citations of posts (as we described). As a point of reference, there are around 0.5 million authors total, and 4.7 million unique edges between them.

We next use the data described to explore patterns of conversations in networks and attempt to model the observed behaviors.

Chapter 6

Patterns of network conversation

PROBLEM STATEMENT: *Given a set of interactions in an online environment– from blogs, message boards, or other media– what structural and temporal patterns can we identify in the* cascades *formed by these interactions?*

In this part we take a more fine-grained approach to the structure of networks than in Part I. Specifically, we will address the interactions between entities in networks, in two online domains: blogs and online groups, as described in the previous chapter. We will extract *cascades* from the data, and also study the topological aspects of these cascades. We will aim to answer the following questions:

- What are the temporal patterns of conversations in online interactions? What is the distribution of inter-posting times in blogs? How quickly do in-links to a given blog post decay? What activities have bursty behavior?
- What are typical shapes of cascades? Do most conversations follow "star" patterns, longer "chains," or something in-between?
- What is the pattern of degree distributions within cascades? How does this vary by level?
- What are patterns of authorship in conversations?

We begin with patterns found in blogs, and follow with patterns found in online groups such as Usenet and Yahoo! Groups.

6.1 Blogs

We analyze properties of cascades formed by hyperlinks between blogs.

The cascade extraction procedure was described in Section 5.1.1 on page 54.

6.1.1 Pattern 1: Popularity decay power law

First we examine temporal properties, such as how a post's popularity grows and declines over time. We collect all in-links to a post and plot the number of links occurring after each day following the post. This creates a curve that indicates the rise and fall of popularity. By aggregating over a large set of posts we obtain a more general pattern.

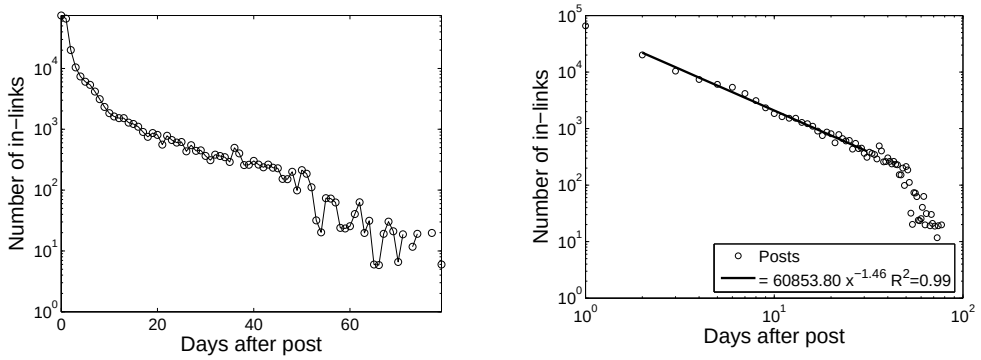


Figure 6.1: Number of in-links vs. the days after the post (a) linear scales, (b) log-log scales. Power law fit to the data has exponent -1.46.

Figure 6.1(a) shows the post dropoff on linear scales, and Figure 6.1(b) shows the post dropoff on log-log scale (normalized for weekly periodicity, see details in [140]).

We fit the power-law distribution with a cut-off in the tail (due to the fact that most posts have complete in-links only for 30 days following publication). Again for the purposes of normalization, we performed the fitting for all days of the week separately (Figure 6.1 shows dropoff for posts first appearing on Monday), and found a stable power-law exponent of around -1.5, which is exactly the value predicted by the model where the bursty nature of human behavior is a consequence of a queuing process [19]. Thus,

Observation 6.1.1 (Blog in-link decay) *The probability that a post written at time t_p acquires a link at time $t_p + \Delta$ is:*

$$p(t_p + \Delta) \propto \Delta^{-1.5}$$

6.1.2 Pattern 2: Inter-Posting Time

How often does a blogger write posts? Is there a distribution that will help predict the length of a hiatus between posts in a blog? Through further analysis we discovered the following temporal pattern, relating to how often bloggers pause between two posts (see Figure 6.2).

Observation 6.1.2 (Inter-posting Time for blog authors) *The PDF of the Inter-Posting-Time follows a power law of exponent -2.7. The inter-posting time is defined as the time between two consecutive posts of the same blogger.*

6.1.3 Pattern 3: Burstiness in blogs

For a given blogger, do we find periodicities or uniform behavior, or is the activity bursty? We use the b-model as described in Section 5.1.2 to measure whether we can characterize blog behavior as bursty. We focus on three time sequences: $p(t)$ (posts over time), in-links $i(t)$, out-links $o(t)$, downward conversation mass $md(t)$, etc. Using the bursty view point and the "bias factor," we have the following observation:

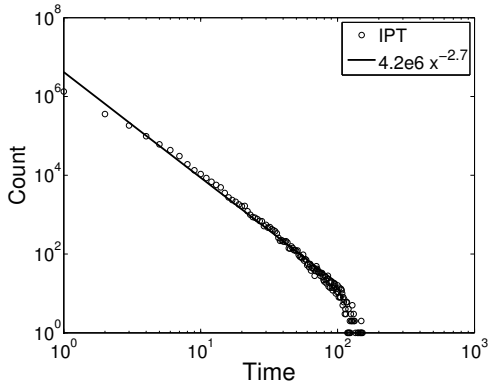


Figure 6.2: Inter-posting time in blogs.

Observation 6.1.3 (Burstiness in blogs) *Most of the time series of interest are self-similar. Most of the bias factors are in the 70% range, that is, much more bursty than uniform (Poisson).*

See Figure 6.3 for sequences and entropy plots. Self-similarity in these cases is surprising: one might expect the entropy plots to be parabolic, or piece-wise linear. Yet, most of them are indeed self-similar! The uniform distribution (or Poisson arrivals) would lead to bias factors around 50% (fifty-fifty splits), but the bias factors we measured are much larger. This burstiness can perhaps be explained as a few posts "hitting a nerve" and attracting a lot of interest compared to others.

6.1.4 Pattern 4: Common cascade shapes

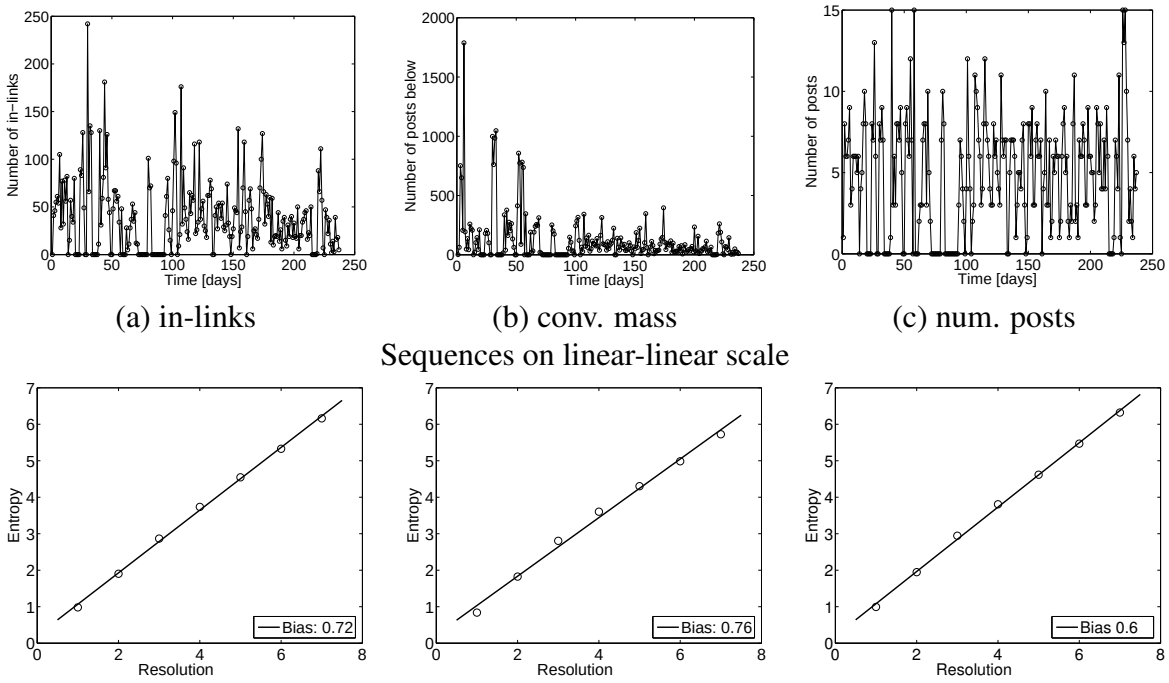
Having examined temporal patterns, we shift our focus to structural patterns in cascades. To obtain the examples of the common shapes and count their frequency we used the algorithms as described in [139]. We give examples of common *PostNet* cascade shapes in Figure 6.4. Graphs are ordered by frequency and the subscript of the label gives frequency rank. Thus, G_{124} is 124^{th} most frequent cascade with 11 occurrences.

Of the $2,092,418$ cascades, 97% are trivial cascades (isolated posts), 1.8% are smallest non-trivial cascades (G_2), and the remaining 1.2% of the cascades are topologically more complex.

Most cascades can essentially be constructed from instances of stars and trees, which can model more complicated behavior like that shown in Figure 6.4. Cascades tend to be wide, and not too deep. Structure G_{107}, which we call a *cite-all chain*, is especially interesting. Each post in a chain refers to every post before it in the chain.

Observation 6.1.4 (Cascade shapes) *We find that the cascades found in the graph tend to take certain shapes preferentially, stars being more common than chains.*

Also notice that cascade frequency rank does not simply decrease as a function of the cascade size. For example, as shown on Figure 6.4, a 4-star (G_4) is more common than a chain of 3 nodes (G_5). In general stars and shallow bursty cascades are the most common type of cascades.



(a) in-links (b) conv. mass (c) num. posts

Sequences on linear-linear scale

(d) in-links (e) conv. mass (f) num. posts

Entropy plots

Figure 6.3: Blogging behaviors are bursty: in-links, conversation mass and number of posts, over time, for the `www.MichelleMalkin.com` blog. The top row shows the data sequences, and the bottom row shows the *entropy plots (*see text - entropy versus resolution r'*)*: they are all linear, which means that the time sequences are self-similar. (Uniform behavior would have bias factor of 0.5, while the observed bias factor is higher.)

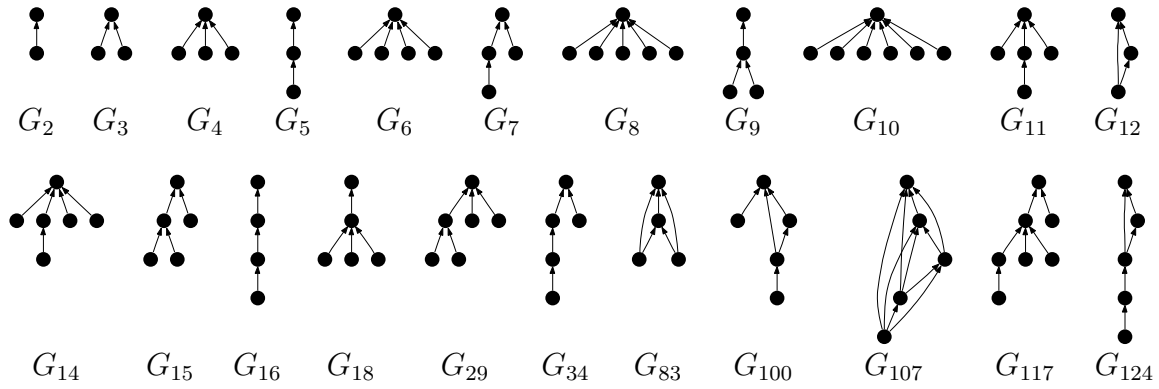
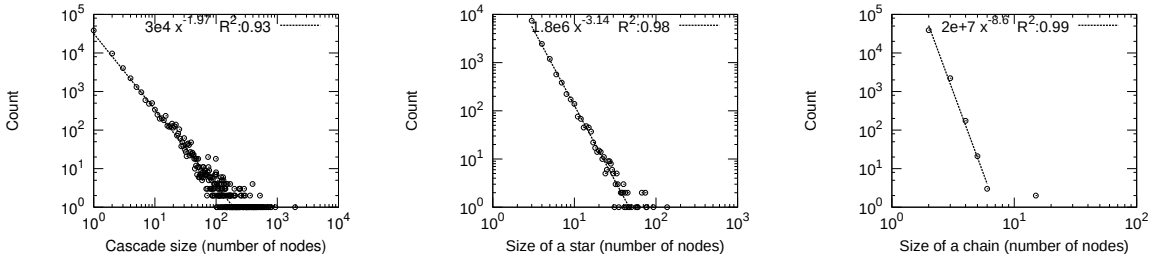


Figure 6.4: Common cascade shapes ordered by the frequency. Cascade with label G_r has the frequency rank r.



(a) All cascades (b) Star cascade (c) Chain cascade

Figure 6.5: Size distribution over all cascades (a), only stars (b), and chains (c). They all follow heavy tailed distributions with increasingly steeper slopes.

6.1.5 Pattern 5: Cascade Size Distribution

What distribution do cascade sizes follow? Does the probability of observing a cascade on n nodes decreases exponentially with n? We examine the *Cascade Size Distributions* over the bag of cascades extracted from the *PostNet*. We consider three different distributions: over all cascade size distribution, and separate size distributions of star and chain cascades. We chose stars and chains since they are well defined, and given the number of nodes in the cascade, there is no ambiguity in the topology of a star or a chain.

Figure 6.5 gives the Cascade Size Distribution plots. Notice all follow a heavy-tailed distribution. We fit a power-law distribution and observe that overall cascade size distribution has power-law exponent of ≈ -2 (Figure 6.5(a)), stars have ≈ -3.1 (Figure 6.5(b)), and chains are small and rare and decay with exponent ≈ -8.5 (Fig. 6.5(c)).

Observation 6.1.5 (Cascade size power law) *Probability of observing a cascade on n nodes follows a Zipf distribution:*

$$p(n) \propto n^{-2}$$

We also observe power laws with the sizes of particular shapes, such as stars and chains.

6.1.6 Pattern 6: Collisions of cascades

By the definition we adopt for blogs, the cascade has a single initiator node, but in real life one would also expect that cascades collide and merge. As illustrated in Figure 5.2(c) on page 55, there are connector nodes which are the first to bring together separate cascades. As the cascades merge, all the nodes bellow the connector node now belong to multiple cascades. We measure the distribution over the connector nodes and the nodes that belong to multiple cascades.

First, we consider only the connector nodes and plot the distribution over how many cascades a connector joins (Figure 6.6(a)). We only consider nodes with out-degree greater than 1, since nodes with out-degree 1 are not connecting multiple cascades. There are still posts that have out-degree greater than 1, and connect only one cascade. These are the posts that point multiple out-links inside the same cascade (e.g. G_{12} and G_{107} of Figure 6.4). The dip the at the number of joined cascades equal to 1 in Figure 6.6(a) gives the number of such nodes.

As cascades merge, all the nodes that follow belong to multiple cascades. Figure 6.6(b) gives the distribution over the number of cascades a node belongs to. Here we consider all the nodes and find out that 98% of all nodes belong to a single cascade, and the rest of distribution follows a power-law with exponent -2.2.

Observation 6.1.6 (Collisions in blog cascades) *The number of cascades a post in a blog joins follows a power law, with exponent -2.2. Collision nodes (nodes with out-degree greater than 1) also follow a power law in number of cascades joined, with exponent -3.1.*

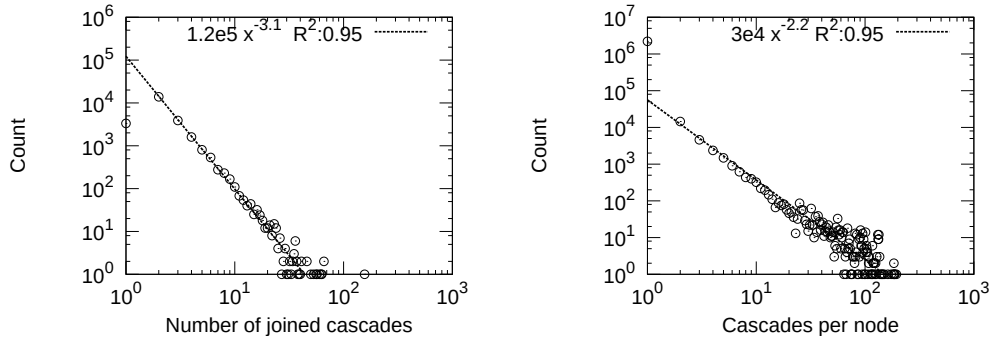


Figure 6.6: Distribution of joined cascades by the connector nodes (a). We only consider nodes with out-degree greater than 1. Distribution of a number of cascades a post belongs to (b); 98% of posts belong to a single cascade.

6.2 Both blogs and groups

While some patterns in blogs (collisions) are not relevant for groups, some patterns were observed in both blog and groups data.

6.2.1 Pattern 7: Cascade size vs. depth and breadth

As suggested by Figure 6.4, most blog cascades follow tree-like shapes. To further verify this we examine how the diameter, defined as the length of the longest undirected path in the cascade, and the relation between the number of nodes and the number of edges in the cascade change with the cascade size in Figure 6.7.

This gives further evidence that the cascades are mostly tree-like. We plot the number of nodes in the cascade vs. the number of edges in the cascade in Figure 6.7(a). Notice the number of edges e in the cascade increases almost linearly with the number of nodes n ($e \propto n^{1.03}$). This suggests that the average degree in the cascade remains constant as the cascade grows, which is a property of full m-ary trees, such as stars. Next, we also measure cascade diameter vs. cascade size (Figure 6.7(b)). We plot on linear-log scales and fit a logarithmic function. Notice the diameter increases logarithmically with the size of the cascade, which means the cascade needs to grow exponentially to gain linear increase in diameter, which is again a property of the balanced trees and very sparse graphs.

In groups data, we study the distribution of cascade sizes and depth (which is the length of the maximum path to a leaf from the root in a thread). Figure 6.7(c) shows the size and the depth distribution in USENET. As we note, not surprisingly, these are both heavy-tailed.

Next, we consider the relationship between size and depth: what is the average depth of a thread of a given size? Figure 6.7(c-d) plots this data. It somewhat surprising that there is a power law relationship between size and depth: the size is roughly quadratic in depth. This observations hints that traditional models such as preferential attachment are probably insufficient to model conversation threads, since such models generate graphs with logarithmic diameter.

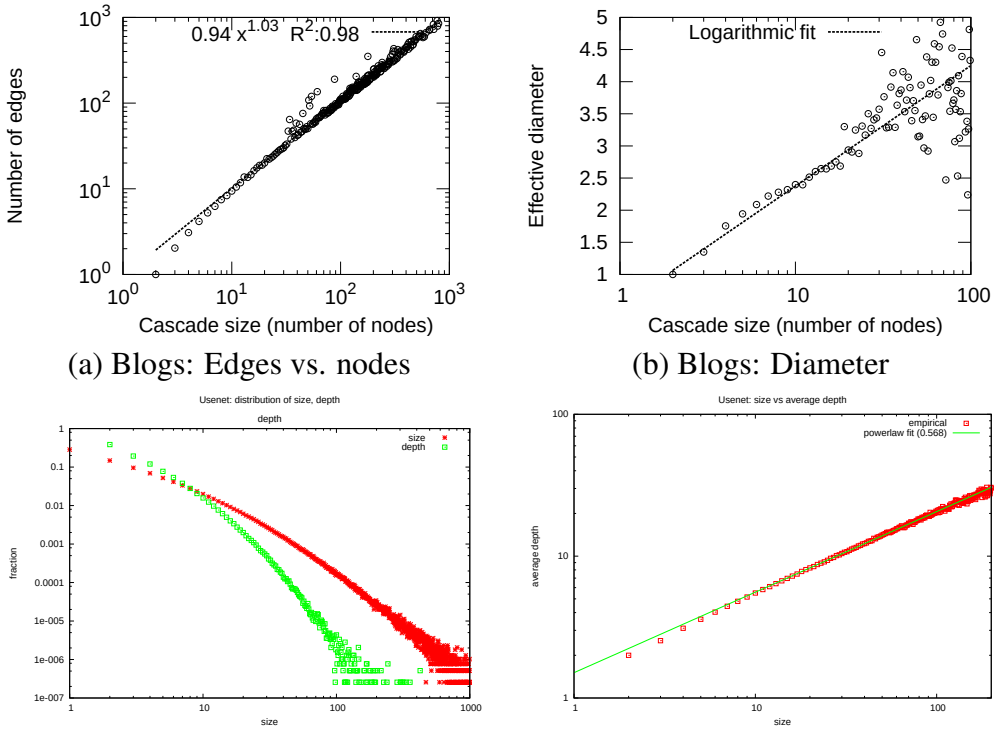
Observation 6.2.1 (Sizes and dimensions of cascades) *Size vs. depth appears to follow a power law, while size vs. breadth appears logarithmic.*

6.2.2 Pattern 8: Cascade degree

We also observe the degree distributions of the cascades. This means that, for blogs, from the *PostNet* we extract all the cascades and measure the overall degree distribution. Essentially we work with a *bag of cascades*, where we treat a cascade as separate disconnected sub-graph in a large network. For groups, only in-degree is considered (as all posts have out-degree 1).

Figure 6.8(a) plots the out-degree distribution of the blog cascades. Notice the cascade out-degree distribution is truncated, which is the result of not perfect link extraction algorithm and the upper bound on the post out-degree (500). Figure 6.8(b) shows the in-degree distribution of the bag of cascades.

Observation 6.2.2 (Cascade degree distribution) *Cascade degree distribution follows a power law, with exponent about -2 in blogs for in-degree and -2.2 for out-degree.*



(a) Blogs: Edges vs. nodes

(b) Blogs: Diameter

(c) Usenet: Size and depth

(d) Usenet: Size vs. depth

Figure 6.7: (a-b): Diameter and the number of edges vs. the cascade size. Notice that diameter increases logarithmically with the cascade size, while the number of edges basically grows linearly with the cascade size. This suggests cascades are mostly tree-like structures. (c-d): Usenet size and depth distributions. The size is superlinear with depth, suggesting that cascades are neither complete chains nor complete stars.

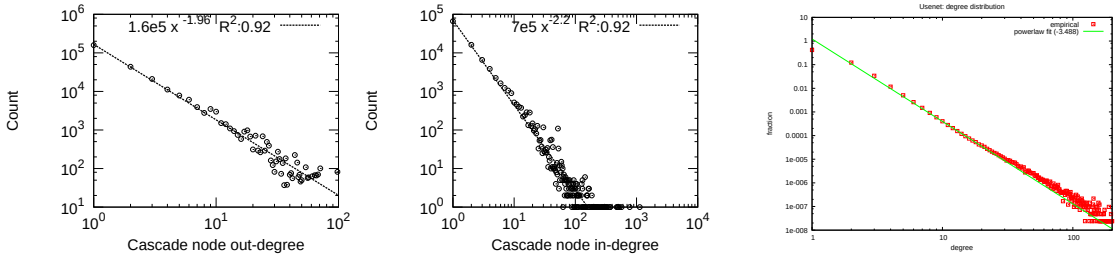
From Figure 6.9(a), it is arguable that the degree distribution for Usenet is close to a power law, i.e., $p(k) \propto k^{-\alpha}$ for some $\alpha > 2$.

6.2.3 Pattern 9: Per-level degree distribution

Is the degree distribution independent of the level of a thread? Figure 6.9(a) shows the degree distribution at each level of a Usenet thread (the root is assumed to be at level 1). The distribution becomes "steeper" with the level since having more children becomes less likely at higher levels.

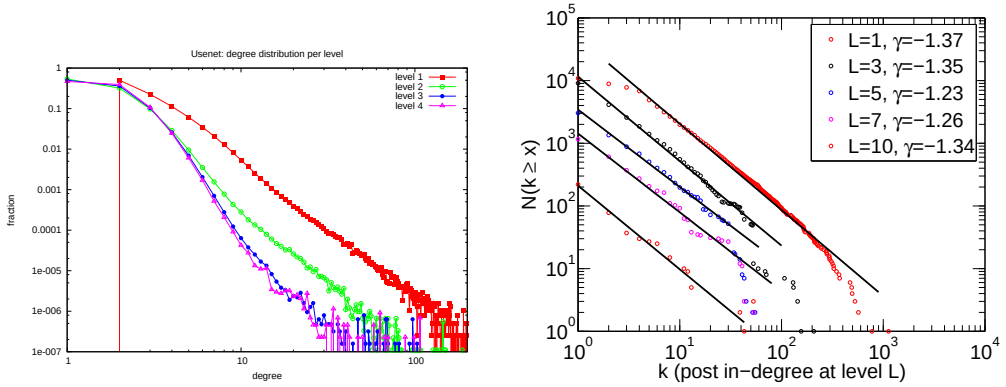
Interestingly, we do not observe this pattern in blogs: there, the steepness appears constant. Figure 6.9(b) plots the in-degree distribution of nodes at level L of the cascade. A node is at level L if it is L hops away from the root (cascade initiator) node. Notice that the in-degree exponent is stable and does not change much given the level in the cascade. This means that posts still attract attention (get linked) even if they are somewhat late in the cascade and appear towards the bottom of it.

Observation 6.2.3 (Per-level degree distribution in cascades) *While by-level degree distribution is heavy-tailed, the slope is constant in blogs but not for groups.*



(a) Blogs: Out-degree (b) Blogs: In-degree (c) Usenet: In-degree

Figure 6.8: (a-b): Out- and in-degree distribution over all cascades extracted from the *PostNet*(c): Degree distribution of threads in USENET.



(a) Usenet: In-degree by level (b) Blogs: In-degree by level

Figure 6.9: Per-level degree distribution in (a) USENET and (b) blogs. Note all distributions are heavy tailed.

6.3 Usenet and discussion groups

In groups we have also begun to study authorship patterns. Most of the observations are illustrated for USENET; the Y! groups dataset, as well as Twitter, follow mostly similar qualitative patterns, although the actual parameters vary.

6.3.1 Pattern 10: Authorship in cascades

What are properties of authors of messages in a thread (that is, a cascade in an online group)? We first consider the size of a thread and the average number of distinct authors in the thread. We also consider the average of the most number of times an author occurs in a thread. Figure 6.10 shows these plots. We find that there is a polynomial relationship between the size of a thread and the number of authors participating in the thread. In fact, this relationship is very reminiscent of the *Heap's Law* in information retrieval [100], which relates the vocabulary size to the document collection size.

Observation 6.3.1 (Authorship in cascades) *Super-linear behavior characterizes authorship in groups. As the number of authors increases, the thread becomes super-linearly larger.*

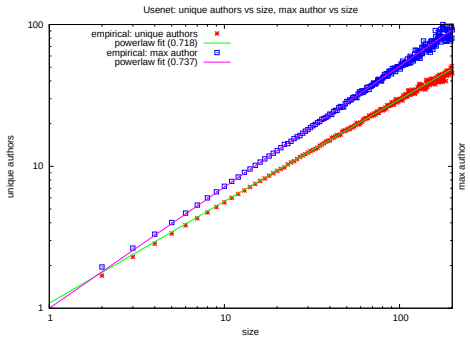


Figure 6.10: Average number of unique authors and maximum author activity vs thread size in USENET. As the number of authors increases (or one author becomes more active), the thread becomes super-linearly larger.

6.4 Summary of patterns and contributions

In this chapter we have studied cascades extracted from various online media. We have looked at temporal as well as structural patterns.

We found that the popularity of posts drops off with a power law distribution. Intuition might lead one to believe that people would lose interest in a post topic in an exponential pattern. However, since linking patterns are based on the behaviors of individuals over several instances, much like other real-world patterns that follow power laws such as traffic to Web pages and scientists' response times to letters [203], it is reasonable to believe that a high number of individuals link posts quickly, and later linkers fall off with a heavy-tailed pattern. In fact, the slope of the dropoff in log-log scale had a value of 1.59, which is reasonably close to the value of 1.5 of correspondence response times in the work of Vázquez et. al.

In this work we have focused on in-link behaviors of blog posts. However, we believe that connector nodes also play an important part in the blogosphere, and would like to develop a plausible model to explain behavior of different out-linking patterns among blogs. We will address this in the next chapter.

In summary, the patterns are as follows:

- Temporal Patterns: Inter-posting time and in-link decay both follow power laws. Almost all activity in blogs is characterized by burstiness.
- Several power laws in the structure of cascades. Cascade shapes in blogs tend toward stars, more than chains, and follow power laws in size. The degree distribution in cascades follows a power law; however, the degree per-level in cascades follow different distributions.
- Number of authors per cascade has *fortification* behavior: adding more authors requires superlinearly more nodes to be added.

Additional case studies in blogs and online groups may be found in the Appendix.

Chapter 7

Models of network conversation

PROBLEM STATEMENT: *Given knowledge of several temporal and cascade patterns in online social networks, (power law in in-links, cascade shapes and sizes, etc.) can we propose intuitive models that will generate this behavior?*

Having explored patterns of cascades, we seek to determine generative models to mimic this behavior. Generative models help explain some of the behavior observed, suggesting certain human behavioral processes over others and allowing for better forecasting. As we approached analyzing the patterns differently for groups and blogs, we also will use different generative models. The nature of linking blogs and groups is different: groups are centralized, which makes it easier for a reader to view all postings, rather than following a few blogs. Therefore, a model successful for generating groups behavior will not necessarily be realistic for blogs, and vice versa.

We begin by modeling cascades in groups. To set up a baseline, we begin by exploring a branching process model to generate cascades. We notice that it fails to produce any realistic behaviors. We improve upon this baseline by proposing an alternate model, the Time-Identity model, which generates cascades by a modified form of preferential attachment (which also takes into account recency), assigns identities by a Polya urn process, and obeys the patterns found in online groups.

We then propose models for generating blog behavior. The first proposed model for blogs is the Cascade Generation Model, which uses the induced social network and shows how an epidemiological model (SIS) on the network will produce realistic cascade properties. This gives credence to epidemiological models for diffusion in networks. Finally, we attempt to build the blog network from scratch as well as generate cascades, which we successfully do with the BlogModel.

7.1 Models for online groups

Can we develop a process to produce realistic cascades found in groups? We will first propose a baseline model, show that it is unrealistic, and then propose a more realistic model.

7.1.1 A baseline: Branching processes

The Galton-Watson branching process [98] is a classical model for generating a random tree in probability theory, making it an intuitive starting point. This models many phenomena like the growth of a population [34]. We study this model as a generative model for threads, and discuss properties of the real conversations that it does or does not satisfy. This is perhaps the most basic tree generation model, and serves as a benchmark for us, similar to the role the Erdös-Renyi graph plays in graph generation models.

Definition of baseline BP-MODEL

Recall that in branching processes, each individual in generation i produces a random number of individuals in generation $i+1$ according to a probability distribution. These random numbers are drawn independently for different nodes.

Let p be a fixed probability distribution on non-negative integers. The messages in a thread are generated by the following process. Each thread starts with a root node and proceeds in discrete steps. At the ith step of the process, each leaf at the ith level of the thread constructed so far independently generates a certain number of children according to the distribution p, i.e., a leaf u has k children with probability $p(k)$. If $k = 0$, then u is a leaf. If $k > 0$, then the children of u participate in the $(i+1)$st step. The process terminates when there are no more new children.

Notice that the only parameter of the model is the distribution p. We can fit the real dataset to BP-MODEL and compute the maximum likelihood estimate for this parameter: $p(k)$ is estimated to be the fraction of nodes with k children in the data; it can be easily shown that this is indeed the maximum-likelihood estimator. BP-MODEL can simulate the inferred distribution in order to generate the threads.

Analysis of BP-MODEL

Let Z_i be the random variable denoting the number of children at the ith level of the threads, and μ be the mean number of children of a node. Let $Z = \sum_i Z_i$ be the random variable denoting the size of the thread. From the definition of a branching process, the mean size of a thread is given by the recurrence

$$E[Z] = 1 + \sum_{i=1}^{\infty} ip(i)E[Z] \Longrightarrow E[Z] = (1-\mu)^{-1}.$$

In our case, from $\mu < 1$ for all datasets tested (Y! Groups, Usenet, and Twitter), the branching process dies out almost surely.

We now analyze the tails of two properties of the threads generated by the model, namely, their size and their depth. We first show that the tail of the size distribution is qualitatively similar to that of the degree distribution. Let $X \sim p$ be a random variable distributed according to p.

Lemma 2 *For any $i > 0$ and $k > 0$, $E[X^k] < \infty$ if and only if $E[Z_i^k] < \infty$.*

Proof 3 *It is easy to see that the size distribution stochastically dominates the degree distribution. Therefore, if the degree distribution does not have a finite kth moment, then the size distribution also does not have a finite kth moment.*

Conversely, we show that if the degree distribution has a finite kth moment, then the kth moment of the size distribution is also finite. For simplicity, we illustrate this for $k = 2$. From the basic theory of branching processes [98], the generating function for Z_i is given by the ith iterate f_i of the generating function f of p. The second moment of Z_i is given by $f_i''(1)$. We know that $f_1'(1) = f'(1) = \mu$ and let $f_1''(1) = f''(1) = \nu < \infty$ by assumption. It is also easy to see that $f_i''(1) = \mu^i$. By simple calculations, one can obtain the recurrence

$$f_i''(1) = f'(1)f_{i-1}''(1) + f''(1)(f_{i-1}'(1))^2 = \mu f_{i-1}''(1) + \nu\mu^{2(i-1)},$$

from which

$$f_i''(1) = i\nu\mu^{i+1}\frac{\mu^i - 1}{\mu - 1} < \infty.$$

An important corollary of the above lemma is the following:

Theorem 5 *The distribution of the size of the tree generated using a branching process follows a power-law distribution, if and only if the distribution of the number of children is a power law.*

Proof 4 *If the Z's tail follows a power law with exponent $-k$, then its kth moment is infinite. So, in order to have power-law tails, some $E(Z_i^k)$ must be infinite. This is the case if and only if the corresponding $E(X^k)$ is infinite (X is a power law).*

Next, we analyze the depth of threads generated by the model. We show that the depth has an exponential vanishing tail.

Lemma 3 *If $\mu < 0$, the probability that the tree generated by the branching process has depth at least i is exponentially small in i.*

Proof 5 *The expected number of children in the ith generation is given by $\mathrm{E}[Z_i] = \mu^i$. For a tree to have depth at least i, this number must be at least 1. By the Markov inequality, the probability of this event is at most $\Pr[Z_i \geq 1] \leq \mathrm{E}[Z_i] = \mu^i$.*

From this, we see that the distribution of depths of threads generated by BP-MODEL does not have a power law tail.

Drawbacks of BP-MODEL

The main advantage of BP-MODEL is its conceptual simplicity. Furthermore, it is also easy to estimate the parameters of the model, and as we observed, the parameter (i.e., the degree distribution) can be succinctly approximated by a power law. By Lemma 2, it also leads to a power-law size distribution, provided the degree distribution is a power law (see Figure 6.7).

The main drawbacks of BP-MODEL are the following.

(1) The model, while generative, is not on-line. Furthermore, it does not naturally produce many properties: the degree distribution is stipulated and the messages are created according to this distribution. In this sense, this model is similar to the configuration model [161] in random graph theory, where a random graph with a specified degree sequence is generated. The model does not try to abstract the social processes behind the creation of messages and the growth of threads.

(2) This model cannot capture the depth distributions of threads that are observed in reality (Figure 6.7(c)). From Lemma 3, we know that the depth cannot be a power law; this is seen in

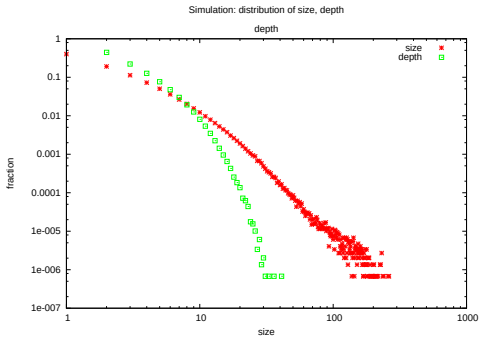


Figure 7.1: Size and depth distribution of threads using BP-MODEL (with p estimated from USENET).

Figure 7.1. BP-MODEL also cannot capture the quadratic relationship between size and depth in Figure 6.7(d).

Moreover, the size distribution generated by the model has a tail that is quantitatively similar to that of the degree distribution. However, in reality, the size distribution has a flatter tail than the degree distribution.

(3) In the branching process model, the number of children at each node is determined by a single distribution. However, this is not realistic for groups, as was shown in Section 6.2. A vanilla branching process model cannot capture this phenomenon (although varying the branching factor at different depths would).

(4) The branching process model does not capture the order in which the messages are created, i.e., the timestamps associated with the messages are left out. Furthermore, the model does not capture the author of messages. These are two critical parameters that distill the essence of conversations in social settings, so we attempt to improve upon this model.

7.1.2 TI-Model

Having rejected BP-MODEL, we propose a new model for the growth of threads of conversation. Our proposed TI-MODEL (for "time-identity model") has two mechanisms: generating cascades and assigning authorship.

Our intuition is as follows. Under preferential attachment, messages that have already received many replies are more likely to receive a new reply, which would lead to older posts gaining the most followers. Paradoxically, as suggested by the Popularity Decay power law (Section 6.1.1, page 67), new messages receive more attention than old ones. This effect might not be very pronounced in the growth of networks such as the Web where the nodes (webpages) have a relatively long "lifespan." On discussion forums and in blogs, however, messages quickly become outdated, and there is a clearly observable tendency that a new message added to a thread is in response to a relatively recent message. Our model captures this fact by attaching new nodes to posts in a preferential process that is based not only on high degree, but also on recency. Furthermore, we assign authorship based on intuitive behaviors of responses.

Definition of proposed TI-MODEL

We now give a formal definition of TI-MODEL in two phases: cascade generation and author assignment.

TI-MODEL **cascade (thread) generation.** We assume a thread grows in discrete time steps. Each time, either a decision is made to stop the thread (i.e., no more message will be added to it), or to add a message in reply to one of the current messages in the thread denoted by v (i.e., the new vertex will be added as a child of v). The probability of the latter decision depends on two parameters of the vertex v. One parameter is the current degree of v. We denote this by $\deg_v$. The other parameter, called the *recency* of v and denoted by r_v, is the number of time steps since v was added to the thread.

In general, we take the probability of the decision to add a child to v to be proportional to some function $h(\deg_v, r_v)$ of the degree and recency of v, and the probability of death to be proportional to a constant δ. That is, the probability of adding a child to v is $\frac{h(\deg_v, r_v)}{\sum_u h(\deg_u, r_u)+\delta}$ and the probability of termination is $\frac{\delta}{\sum_u h(\deg_u, r_u)+\delta}$, where the summation in the denominator is over all nodes u currently in the thread.

We focus on a particular form of the function h: when h is a linear combination of $\deg_v$ and and an exponentially decreasing function in r_v. That is, $h(\deg_v, r_v) = \alpha \deg_v + \tau^{r_v}$ for constants $\alpha \geq 0$ and $\tau \in (0, 1)$. We choose this form of function because of the following reasons.

(1) An exponential "discounting" function like τ^{r_v} is the simplest way to model dependence on time.

(2) A linear combination is perhaps the simplest and most natural way to combine the recency and degree[1].

(3) Considering a linear combination (as opposed to, e.g., the square root of the degree plus the exponential discount) allows us to compute the denominator of the probability expressions independent of the current degrees, and this makes this model particularly amenable to mathematical analysis.

Note that both the degree and recency components play a role in generating different types of threads. If the former plays a prominent role, then we get "bushy" threads, where many messages are in response to a single earlier message. If the latter plays a prominent role, then we get "skinny" threads, where the thread is essentially a path and messages appear in succession as a cascade of responses.

TI-MODEL **author assignment.** In addition to understanding the process by which the thread structures are generated, we also want to understand *who* is responsible for generating the reply message. Therefore, we extend the model to author identity. The motivation is that authors tend to respond to responses to their own earlier messages. Thus, when a new message v arrives as a child of message u in a thread t, the author $a(v)$ is likely to be chosen from the set $\{a(w)\}$ for

[1]Another natural alternative that we considered is the product of the degree with the exponential discounting term ($h(\deg_v, r_v) = \tau^{r_v} \deg_v$). While this formulation might makes sense intuitively, it does not generate graphs similar to what we see in practice. In particular, the exponential discounting factor does not let the degrees of the vertices to grow to a heavy-tailed distribution. We have also done simulations with a few other reasonable choices of h, and did not observe fundamentally different results.

some w along the path from u to root(t). (There is a slight caveat that w is unlikely to be u since $a(v)$ is most likely not the same as $a(u)$.)

The above observations, combined with the empirical evidence of Heap's law (Figure 6.10), suggests a modified Polya urn process in order to reproduce author identity patterns. When a new message v arrives with $u = \text{parent}(v)$, then $a(v)$ is chosen according to the following process. Let $A'(v) = \text{path}(\text{parent}(v))$.

$$a(v) = \begin{cases} a(w), w \in'_A (v) & \text{wp. } \gamma \\ u & \text{wp. } \epsilon \\ a \in_A & \text{wp. } 1-\gamma-\epsilon \end{cases}$$

Note that this can also be viewed as a variant of the *copying* model [127]: with probability $\gamma > 0$, we copy one of the authors from $\text{path}(\text{parent}(u))$; with probability $\epsilon \ll \min(\gamma, 1-\gamma)$, we copy u itself; and with the remaining probability, we choose a random author from A. By this process, the probability that an author is chosen is dependent on the number of times he/she already authored a message in the path to the root (as well as the position of those messages).

From data, it is easy to statistically learn the parameters γ and ϵ of TI-MODEL. It is possible to show that the above modified Polya urn process generates a heavy tail for the number of occurrences of an author on a path (proof omitted). However, it seems much harder to analyze the number of occurrences in a tree, since different paths share nodes.

Analytical validation of TI-MODEL

In this section we show that the degree distribution of graphs generated from TI-MODEL defined above has a heavy tail.

Theorem 6 *Let G be a thread with n nodes generated from the model in the above section with $h(\deg_v, r_v) = \alpha\deg_v + \tau^{r_v}$. Then for every d, the fraction of nodes of G that have at least d children is at least $\Omega(d^{-1})$.*

Proof 6 (Sketch) *We start by observing that with $h(\deg_v, r_v) = \alpha\deg_v + \tau^{r_v}$, at the time that the thread has k nodes we have*

$$\sum_u h(\deg_u, r_u) = \alpha(k-1) + \sum_{j=1}^{k} \tau^j < \alpha(2k-2) + \tau/(1-\tau).$$

Now, we consider the ith vertex added to the thread, and study the growth of the degree of this node at time t, as t grows. We denote the degree of this node at time t by $d_i(t)$. Note that $d_i(t)$ is a random variable, and $d_i(t+1) - d_i(t)$ is either one (if the $(t+1)$'st node connects to i) or zero (if it doesn't). The probability that $d_i(t+1) - d_i(t) = 1$ is

$$\frac{h(\deg_v, r_v)}{\sum_u h(\deg_u, r_u) + \delta} = \frac{\alpha d_i(t) + \tau^{t+1-i}}{\sum_u h(\deg_u, r_u) + \delta} > \frac{\alpha d_i(t)}{\alpha t + \tau/(1-\tau)}.$$

Therefore, we have

$$\mathrm{E}[d_i(i+1)] \geq 1 \tag{7.1}$$

Dataset	α	τ	δ
USENET	0.1	0.94	0.4
Y!GROUPS	0.7	0.95	0.8

Table 7.1: Parameters of TI-MODEL.

and

$$\mathrm{E}[d_i(t+1)] - \mathrm{E}[d_i(t)] > \frac{\alpha \mathrm{E}[d_i(t)]}{\alpha t + \tau/(1-\tau)}. \tag{7.2}$$

We couple the sequence of random variables $d_i(i+1), d_i(i+2), \ldots$ with another sequence which instead of the inequalities (7.1) and (7.2), satisfies the corresponding equalities. We call these random variables $d'_i(t)$. By coupling, $d_i(t)$ stochastically dominates $d'_i(t)$. Therefore, it is enough to prove the desired lower bounds on $d'_i(t)$ instead of $d_i(t)$. To do this, we first calculate the expected value of $d'_i(t)$, which we denote by $ED_i(t)$. This can be calculated from the recurrence relations given by (7.1) and (7.2). The solution of these recurrences is

$$ED_i(t) = \frac{\alpha t + \tau/(1-\tau)}{\alpha(i+1) + \tau/(1-\tau)}.$$

The above equation can be proved easily by induction on t using recurrences given by (7.1) and (7.2). This means that for every i, the expected degree of the ith node of the thread grows at least linearly with time. Furthermore, the sequence of random variables $d'_i(t)$ defines a martingale, and therefore by standard martingale concentration inequalities [165], if $t - i$ is large enough, the value of $d'_i(t)$ is concentrated around its expectation. Putting these together, we obtain that for $t = n$ large enough and $i < n - O(1)$, with a large probability, we have

$$d_i(n) > \frac{\alpha t}{2(\alpha i + \tau/(1-\tau))}.$$

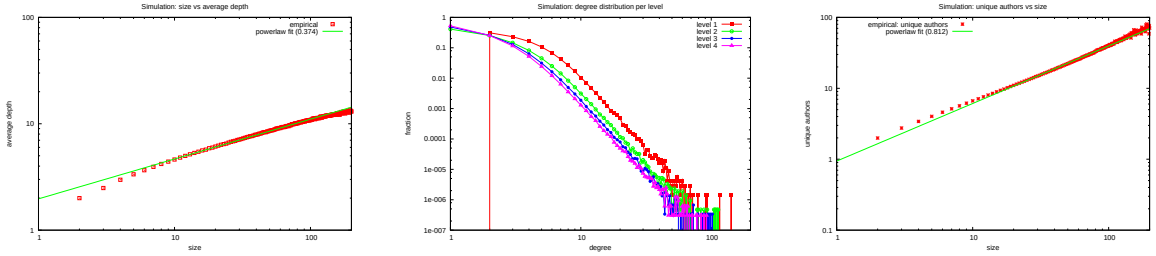
This means that the number of nodes that have degree at least d is bounded from below by the number of i's satisfying $\alpha i + \tau/(1-\tau) < 0.5\alpha n/d$, which is $\Theta(n/d)$. Thus, the fraction of nodes having degree at least d is at least $\Theta(d^{-1})$ ***QED***

Empirical validation of TI-MODEL

We next estimate the parameters of TI-MODEL from the data and simulate the model to see if the statistics match the empirical ones. The parameters are estimated through a simple grid search and maximum likelihood computation. Table 7.1 shows the parameters of TI-MODEL estimated from the data.

We consider the size vs. depth relationship and the degree distribution conditioned at each level, to see if these resemble the empirical observations. Figure 7.2 shows these plots for USENET, simulated using the parameters from Table 7.1. These show that TI-MODEL is able to reasonably capture the empirical observations.

Finally, we consider the number of unique authors as a function of thread size, by using TI-MODEL. Figure 7.2(c) shows the plot. We can see that this is reasonably consistent with the observation we made in Section 6.3.



(a) (b) (c)

Figure 7.2: (a) Size vs. depth. Notice that the model successfully reproduces a power law relationship between size and depth. (b) Per-level degree distribution for TI-MODEL simulation of USENET. (c) Unique authors vs. thread size in TI-MODEL. Notice we successfully reproduce the power law observed in real data.

Using EM, we fit the TI model to various data, and used the fitted parameters to help characterize the groups. This application is shown in Appendix A.

7.2 Models for blogs

Blogs are less centralized than groups– groups tend to form communities around a certain topic, while community detection in blogs is less obvious. Perhaps because of this, diffusion between blogs is more easily understood: there is a more clear social network formed by the hyperlinks between posts, while in groups the induced social network formed by replies is less visible to the average user. Therefore, we seek to find an appropriate model for blog conversations.

What is the underlying process that generates cascades in blogs? Our goal here is to find a generative model that generates cascades with properties observed in Section 6.1. Here our focus is to model blog behavior, rather than groups behavior, and to test whether an epidemiology-based model produces reasonable behavior.

7.2.1 Cascade Generation Model for blogs

We present a conceptual model for generating information cascades that produces cascade graphs matching several properties of real cascades. The model is intuitive and requires only a single parameter that corresponds to how interesting (easy spreading) are the conversations in general in blogs.

Intuitively, cascades are generated by the following principle. Using the observed *BlogNet*, a post is posted at some blog, other bloggers read the post, some create new posts, and link the source post. This process continues and creates a cascade. One can think of cascades being a graph created by the spread of the virus over the *BlogNet*. This means that the initial post corresponds to infecting a blog. As the cascade unveils, the virus (information) spreads over the network and leaves a trail. To model this process we use a single parameter β that measures how infectious are the posts on the blogosphere. Our model is very similar to the SIS (susceptible-infected-susceptible) model from the epidemiology [101]. We next formally describe Cascade

Generation model.

Definition of proposed Cascade Generation model

We begin with the induced *BlogNet*formed by already-observed behavior. Each blog is in one of two states: *infected* or *susceptible*. If a blog is in the infected state this means that the blogger just posted a post, and the blog now has a chance to spread its influence. Only blogs in the susceptible (not infected) state can get infected. When a blog successfully infects another blog, a new node is added to the cascade, and an edge is created between the node and the source of infection. The source immediately recovers, i.e. a node remains in the infected state only for one time step. This gives the model ability to infect a blog multiple times, which corresponds to multiple posts from the blog participating in the same cascade.

More precisely, a single cascade of the *Cascade Generation model* is generated by the following process.

(i) Uniformly at random pick blog u in the *BlogNet* as a starting point of the cascade, set its state to *infected*, and add a new node u to the cascade graph.

(ii) Blog u that is now in infected state, infects each of its uninfected directed neighbors in the *BlogNet* independently with probability β. Let $\{v_1, \ldots, v_n\}$ denote the set of infected neighbors.

(iii) Add new nodes $\{v_1, \ldots, v_n\}$ to the cascade and link them to node u in the cascade.

(iv) Set state of node u to not infected. Continue recursively with step (ii) until no nodes are infected.

We make a few observations about the proposed model. First, note that the blog immediately recovers and thus can get infected multiple times. Every time a blog gets infected a new node is added to the cascade. This accounts for multiple posts from the blog participating in the same cascade. Second, we note that in this version of the model we do not try to account for topics or model the influence of particular blogs. We assume that all blogs and all conversations have the same value of the parameter β. Third, the process as described above generates cascades that are trees. This is not big limitation since we observed that most of the cascades are trees or tree-like. In the spirit of our notion of cascade we assume that cascades have a single starting point, and do not model the collisions of the cascades.

Empirical validation of Cascade Generation model

We validate our model with extensive numerical simulations. We compare the obtained cascades towards the real cascades extracted from the *PostNet*. We find that the model matches the cascade size and degree distributions.

We use the real *BlogNet* over which we propagate the cascades. Using the Cascade Generation model we also generate the same number of cascades as we found in *PostNet* (≈ 2 million). We tried several values of β parameter, and at the end decided to use $\beta = 0.025$. This means that the probability of cascade spreading from the infected to an uninfected blog is 2.5%. We simulated our model 10 times, each time with a different random seed, and report the average.

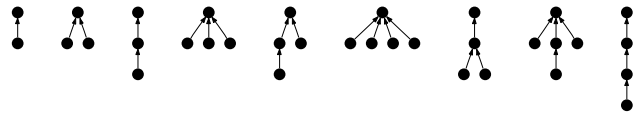


Figure 7.3: Top 10 most frequent cascades as generated by the Cascade Generation model. Notice similar shapes and frequency ranks as in Figure 6.4.

First, we show the top 10 most frequent cascades (ordered by frequency rank) as generated by the Cascade Generation model in Figure 7.3. Comparing them to most frequent cascades from Figure 6.4 we notice that top 7 cascades are matched exactly (with an exception of ranks of G_4 and G_5 swapped), and rest of cascades can also be found in real data.

Next, we show the results on matching the cascade size and degree distributions in Figure 7.4. We plot the true distributions of the cascades extracted from the *PostNet* with dots, and the results of our model are plotted with a dashed line. We compare four properties of cascades: (a) overall cascade size distribution, (b) size distribution of chain cascades, (c) size distribution of stars, and (d) in-degree distribution over all cascades.

Notice a very good agreement between the reality and simulated cascades in all plots. The distribution over of cascade sizes is matched best. Chains and stars are slightly underrepresented, especially in the tail of the distribution where the variance is high. The in-degree distribution is also matched nicely, with an exception of a spike that can be attributed to a set of outlier blogs all with in-degree 52. Note that cascades generated by the Cascade Generation model are all trees, and thus the out-degree for every node is 1.

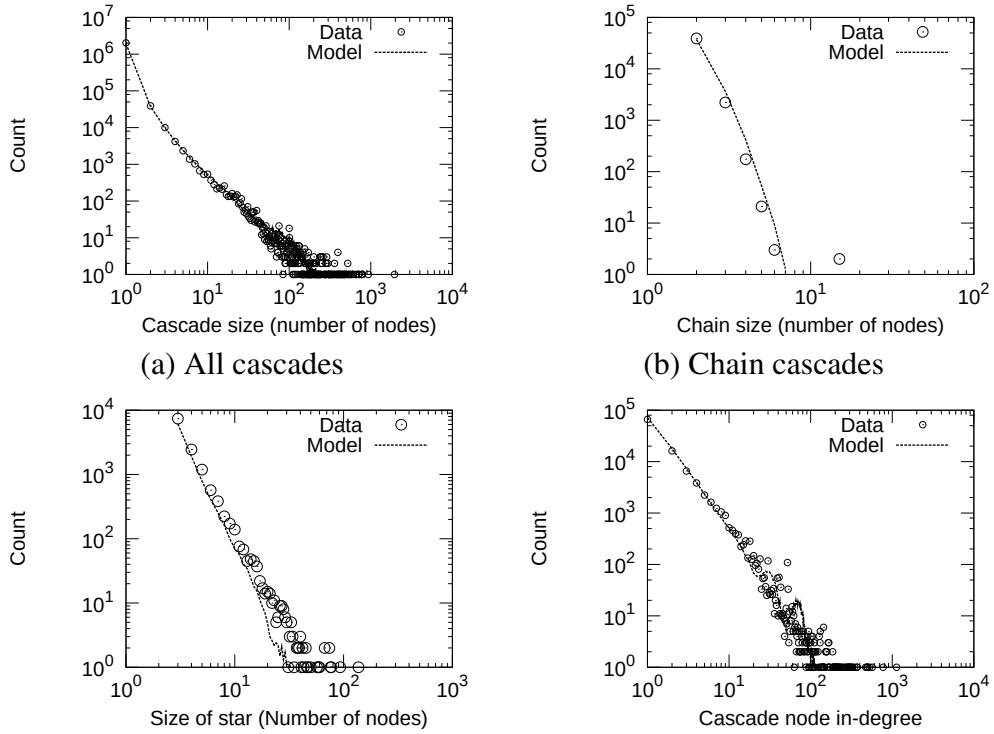
Discussion of Cascade Generation model

We also experimented with other, more sophisticated versions of the model. Namely, we investigated various strategies of selecting a starting point of the cascade, and using edge weights (number of blog-to-blog links) to further boost cascades.

We considered selecting a cascade starting blog based on the blog in-degree, in-weight or the number of posts. We experimented variants where the probability β of propagating via a link is not constant but also depends on the weight of the link (number of references between the blogs). We also considered versions of the model where the probability β exponentially decays as the cascade spreads away from the origin.

We found out that choosing a cascade starting blog in a biased way results in too large cascades and non-heavy tailed distributions of cascade sizes. Intuitively, this can be explained by the fact that popular blogs are in the core of the *BlogNet*, and it is very easy to create large cascades when starting in the core. A similar problem arises when scaling β with the edge weight. This can also be explained by the fact that we are not considering specific topics and associate each edge with a topic (some blog-to-blog edges may be very topic-specific) and thus we allow the cascade to spread over all edges regardless of the particular reason (the topic) that the edge between the blogs exists. This is especially true for blogs like BoingBoing that are very general and just a collection of "wonderful things."

It is surprising that the Cascade Generation model behaves so well despite its simplicity. It



(a) All cascades (b) Chain cascades

(c) Star cascades (d) In-degree distribution

Figure 7.4: Comparison of the true data and the model. We plotted the distribution of the true cascades with circles and the estimate of our model with dashed line. Notice remarkable agreement between the data and the prediction of our simple model.

estimates cascade behavior without taking into account the numerical weight between links. That we were able to obtain such realistic cascades with a constant value of β was also surprising: it would seem intuitive that some topics are more "trendy" than others, and posts based on these topics might gain more links by assuming a higher value of β.

7.2.2 Zero-crossing Model for blogs

An limitation of Cascade Generation model is that it requires the network to be pre-defined. We would like to generate realistic behavior without this assumption—to intuitively model all aspects of blogging behavior. Much like the Butterfly model of Chapter 4, we would like to devise a set of principles or local rules that lead to emerging, macroscopic behavior that matches patterns of Chapter 6.

Motivation and intuition of $\mathcal{ZC}$

Realistic patterns are difficult to create naturally. While models such as *RTM* or the *b-model* reproduce self-similar behavior, these model do not comply with our intuition of the blogger behavior, which is generated online rather than planned in advance. Furthermore, we simulta-

neously model topological and temporal behavior rather than one at a time. Some models fail to do this in an emergent manner, such as the growth function in [107, 115], or the exponential distribution in [121]; or require special assumptions such as a constant rate of answering emails [203].

Baseline model: $\mathcal{EXP}$ model. Our proposed Zero-Crossing model puts together two very different aspects of the blogosphere, time and topology, properties that are much more difficult to model jointly than when considered separately. As existing models usually consider modeling single aspect of the blogosphere such as the mortality of blogs or the information propagation there is no natural model to compare our model to. However, in order to have a baseline comparison, we devised a nontrivial model based on conventional wisdom of exponential post inter-arrival times [121] and "rich get richer" linking behavior. We refer to it as the $\mathcal{EXP}$ model which we define as follows. The inter-posting times for each blog are sampled from an exponential distribution with parameter λ. A blog then creates a post and links to another post that is chosen by the "preferential attachment" rule [17]: the probability of linking to a post is proportional to its current in-degree, which is a measure of its current popularity.

We will later compare patterns generated by our proposed model to the ones that $\mathcal{EXP}$ generates. Next we describe our zero-crossing model ($\mathcal{ZC}$) based on a random walk on a line, which is sketched in Figure 7.5.

$\mathcal{ZC}$ principles. Our model involves three major mechanisms:

- *When would a blogger write a post?* We propose a model based on zero-crossing of a random walk on a discrete line.
- *What does a blogger read and cite?* Once a blogger has decided to blog, which other blogs (if any) will he choose to read, and which posts inside those chosen blogs will he choose to cite? Our idea here is related to the "exploit and explore" strategy: usually, the blogger will choose one of the blogs he has chosen in the past ("exploit"), but occasionally he will read a completely new blog ("explore").
- *How does a blogger follow-up?*: Once a blogger decides to cite post Q, he may follow up on it, and also cite some of the posts that Q is citing; the blogger may do that recursively. We will refer to this mechanism as *link expansion*.

Next we describe the details of each of the mechanisms.

Definition of proposed $\mathcal{ZC}$model

We detail the three main mechanisms, then combine them into one list of rules.

When a blogger writes a post. Our goal here is to generate realistic power-law inter-posting times as well as bursty behavior, which will form the heart of our model. We propose the following mechanism: the blogger does a random walk on a line, and decides to post whenever he is at state **0** (e.g., at his computer). At each time point, a blogger is in a state represented by an integer. There are two possible transitions: with equal probability the blogger either adds or subtracts 1 from his current state. The blogger publishes a post when his state is **0**. In that sense, the state of a blogger describes how far away he is from his computer (or equivalently, how far

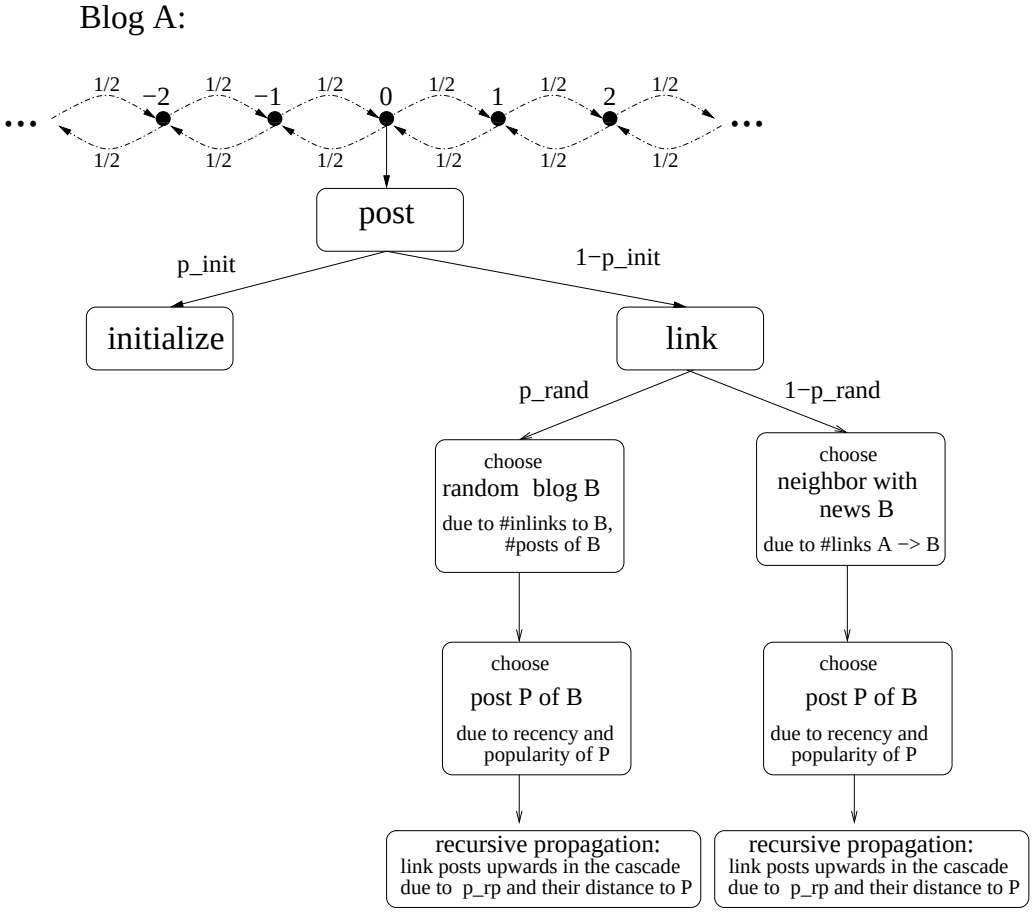


Figure 7.5: Our zero-crossing model $\mathcal{ZC}$. Each blog behaves according to this model. Numbers correspond to the steps of our $\mathcal{ZC}$ generative model.

he is mentally away from the blogging mode). The idea is that random events may distract him to some other, nearby state; if there are too many successive distractions away from state **0**, the blogger will be away from his computer for a long time. This mechanism *provably* generates bursty blogging activity: the blogging time-stamps are exactly the zero-crossings of a random walk (Brownian motion), and it is known that their intrinsic ("fractal") dimension is $f = 0.5$ [148, 188]. See Fig. 7.6 for examples of random walks.

Random walks have also been considered to model and explain how human make decisions in uncertain environments, for instance see [41].

What a blogger reads and cites. In constructing a solution to this part of the problem, our goal is to cause "rich get richer" behavior to generate power laws of in- and out-degree in the blog network.

Once the blogger is ready to post, he may choose to initialize a new cascade (with probability $1 - p_L$), i.e., a new post without any outlinks that others can then cite to create new information cascade. The interesting modeling aspects arise in the opposite case, when the blogger decides

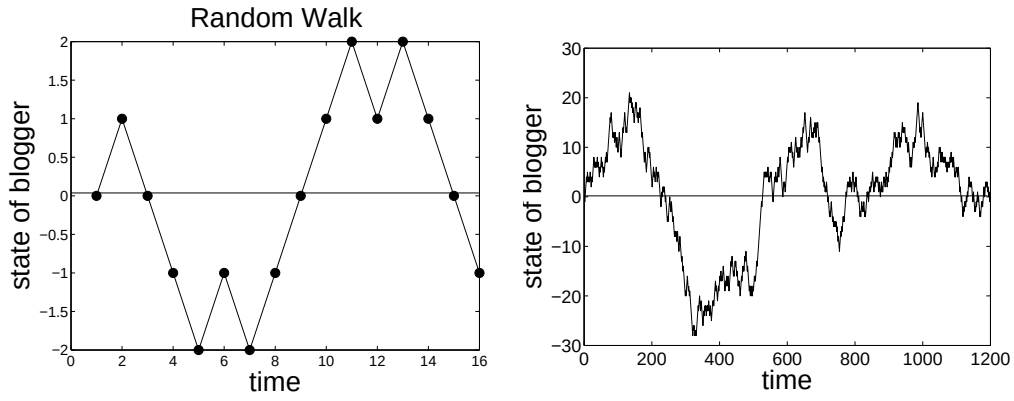


Figure 7.6: Random walk over the states of a blogger. Left, a blogger posts at times 1, 3, 9, 15 when the random walk crosses horizontal axis which gives inter-posting times 2, 6, 6. Right, a longer walk demonstrates the burstiness.

to comment on some other posts and join to an existing cascade.

How does he choose a post to comment on? We propose the following "exploit vs. explore" mechanism, which reflects how humans act: the chosen post will usually belong to one of his favorite blogs. However, once in a while, the blogger may want to cite a post on a completely new blog. Thus, first the blogger decides whether to pick a post of a neighbor ('favorite blog') or a post of a non-neighbor in the blog network. (This will create an emerging network as new links are formed, unlike the CGM model which required a predefined social network.) Among the neighboring blogs, possible candidates are blogs that have published a post since his last visit. He prefers candidates that he preferred in the past, i.e. he chooses a blog proportionately to the number of past links he has made to that blog. We call this the "exploit" mode, where blogger visits favorite blogs that he found valuable/interesting in the past.

In the opposite case, with probability p_E, the blogger goes into the "explore" mode and chooses a blog he has never linked to before. In that case, he trusts the taste of the majority and chooses a blog B proportionally with the total number in-links of B times the number of posts of B. We expect a rich-get-richer setting, because blogs with many in-links probably have higher quality and/or better word-of-mouth ratings, and thus will naturally attract the attention of bloggers.

After choosing a blog, the blogger has to determine on which post to comment. He therefore judges the posts based on their recency and their popularity, i.e., the probability of linking to a post is proportional to the ratio of the number of in-links and the time since the publication of the post.

How a blogger follows up. Now, our blogger can publish his post with a link to the chosen post. He will consider to link to other posts that participated in the same conversation tree, in the same way that scientific papers point to an earlier article A, and often point to the citations of A, and so on recursively. Posts that are many hops away from the chosen post are less likely to be linked: for each post and each path Π from the chosen post to that post he flips a biased coin and with probability $\Pi_{LE}^{|\Pi|}$ he links it.

Notice that our proposed $\mathcal{ZC}$ model heavily relies on how the information flows through the blogosphere. We exploit this both in a topological sense to model how bloggers create links and in a temporal sense to model the dynamics at which new posts are being written.

This completes the description of our artificial blogger. After that, the blogger transitions out of the active blogging state, and the next blogger begins the simulation, in a round-robin fashion. Notice that all the three major steps in our blogger model have very simple, local behavior, with no sophisticated distributions or constraints to guide our blogger. Yet, as we show next, this simple model, repeated over all bloggers, leads to emerging behavior that matches the properties and patterns found on the real blogosphere.

Full list of $\mathcal{ZC}$ rules.

Each blogger has 3 parameters: p_L (prob. of a post creating an out-link), p_E (prob. of exploration mode), and p_{LE} (prob. of expanding a link). All blogs start at position **0** and publish a post in the first round. In each next round each blog A follows the 6 steps of Fig. 7.5:

1. Change state: With probability 1/2 add one to current state of A, and with probability 1/2 subtract one A's state.

2. Create post: If A's current state is not **0** then stop else continue with next step.

3. Initiate cascade: A creates a post P. With probability $1-p_L$, A initializes a new conversation tree (P has no out-links) and stop else continue with next step.

4. Choose mode: With probability p_E blog A is in "exploration" mode and with $1-p_E$ it is in "exploitation" mode.

4.1. "exploitation" mode: Let $N(A)$ be the set of neighboring blogs, blogs A previously linked to. Then the probability of A choosing a neighboring blog B is: $\Pr[A \text{ chooses } B] \propto \#\text{links}(A \to B)$

4.2: "exploration" mode: A chooses a non-neighbor blog. Let $\bar{N}(A)$ be the set of blogs with no in-links from A. The probability of choosing a non-neighbor B is: $\Pr[A \text{ chooses } B] \propto (\#\text{inlinks}(B)+1)(\#\text{posts}(B)+1)$

5. Choose post: The probability of choosing a post Q in blog B is: $\Pr[A \text{ chooses } Q] \propto \frac{\#\text{inlinks}(Q)+1}{\#\text{rounds passed since publication}+1}$. A creates a link from its post P to the post Q of blog B.

6. Link Expansion: For each post R reachable from post P, for each path Π from P to R with probability $\Pi_{LE}^{|\Pi|}$ create a link from post P to post R.

Analytical validation of $\mathcal{ZC}$ Model

Theorem 7 *The inter-posting times in $\mathcal{ZC}$ follow a power law distribution with exponent -1.5.*

***Proof* 1 (Sketch)** *[168]* We first note that the probability of posting at time t in our model (denoted by $u_{t'}$) is zero for odd t' and $2^{-t'}\binom{t'}{t'/2}$ otherwise. We can relate $u_{t'}$ (for even $t'>0$) to the probability of the inter-posting being t (denoted by p_t) as follows:

$$u_{t'} = \sum_{1 \le t \le t'/2} p_{2t} u_{t'-2t}$$

Solving for p_{2t} we obtain $p_{2t} = \frac{\binom{2t}{t}}{(2t-1)2^{2t}}$. Using Sterling's formula in a limit analysis ($t \to \infty$) we obtain the result:

$$p_t \propto t^{-3/2}$$

Theorem 8 *The blogging activity in our $\mathcal{ZC}$ Model is self-similar and bursty.*

***Proof* 2** The fractal dimension of the zero-crossings of Brownian motion is is f = 0.5, see for example [148, 188]. This result extends to our random walk which is a discrete version of Brownian motion.

Empirical validation of $\mathcal{ZC}$

We compare distributions of properties in the real blog cascades with those in the synthetic data produced by $\mathcal{ZC}$Ḟor comparison we also employ the baseline $\mathcal{EXP}$ model. We consider a model to be good if it intrinsically produces patterns and properties similar to those found in the real data. Note that statistical properties of conversations and blog behavior intrinsically emerge from the model and were not in any way "forced."

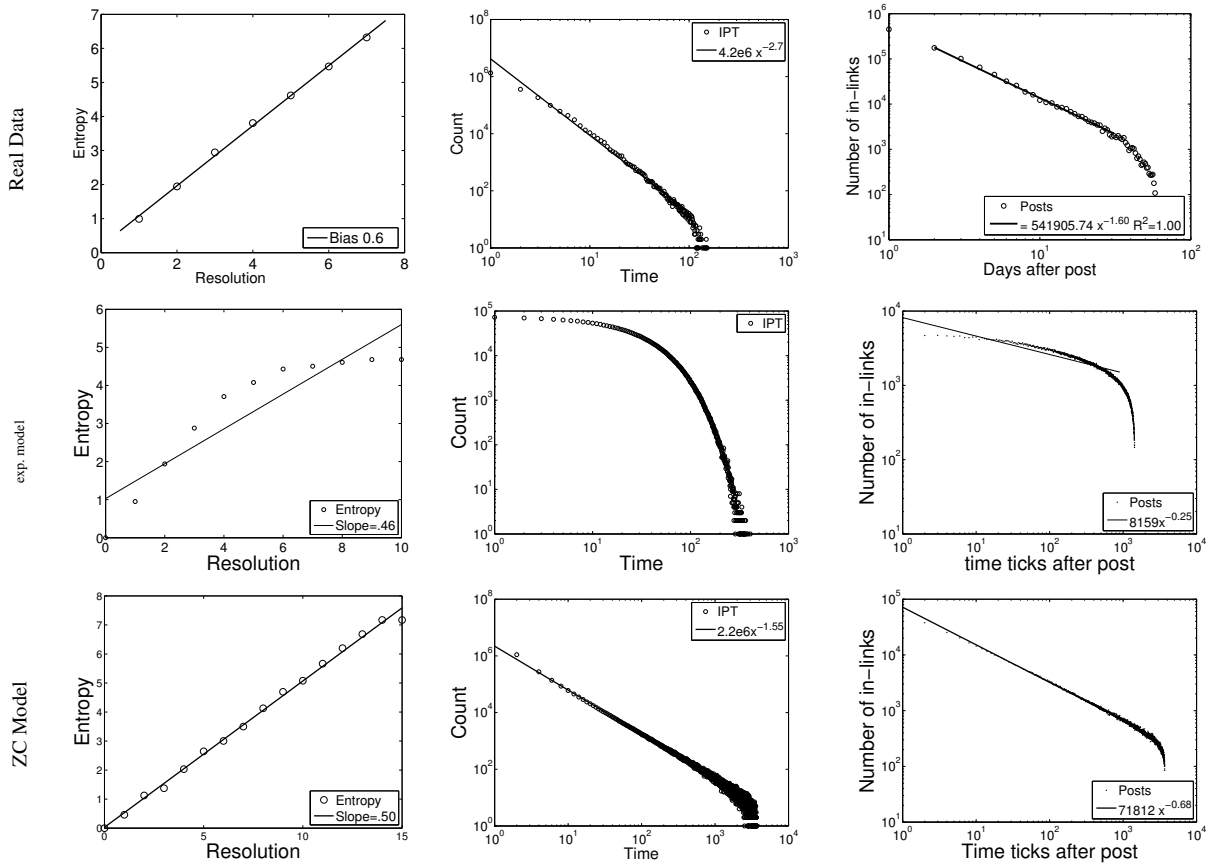
In our experiments, we chose the parameters p_L, p_E and p_{LE} independently and uniformly at random in $[0, 1]$. So, in the $\mathcal{ZC}$ model there are no parameters to set or tune. In order to achieve a good basis of comparison between the real data and the synthetic data, we chose the number of blogs in the simulation to be 45,000 and run it until 2.2 million posts are created.

Temporal Patterns. The first temporal property we measure is burstiness of time sequences. From entropy plots of [152] we observe that the activity of most blogs is self-similar and bursty. Our model $\mathcal{ZC}$ also creates bursty and self-similar activity, as can be seen in Figure 7.7(a). The entropy plots plot the entropy versus resolution, that is $H(W)$, vs. $\log_2(W)$. The plots of the real data and the synthetic data generated by $\mathcal{ZC}$ model are both linear which implies that the activity is self-similar as discussed in section on information fractal dimension. Furthermore, both plots have a slope different from 1, which implies that the activity is bursty and not uniform. Similar plots can be found for most of the blogs [152] in the real data. In fact, our model provably creates self-similar and bursty activity, see Thm. 8. In contrast, the $\mathcal{EXP}$ model does not create self-similar activity (left middle plot of Figure 7.7). Moreover, we can extend the $\mathcal{ZC}$ model to match the slope of the real data more accurate by modifying in an ad-hoc fashion the random walk into a more general form of Brownian motion, a.k.a., anomalous diffusion [63].

We also successfully generate realistic inter-posting times, as shown in figure Figure 7.7(b) (in log-log scales). Our model $\mathcal{ZC}$ matches the shape of the power law distribution perfectly. In fact, the first return times (in our $\mathcal{ZC}$ model: the inter-posting times) follow a power law distribution with exponent -1.5, as we showed in Thm. 7. In contrast, the inter-posting times of the $\mathcal{EXP}$ model follow an exponential distribution.

The final temporal property is the blogosphere is the Popularity Decay power law: the number of in-links a post published at time t obtains at time $t + \delta$. The plot basically measures how quickly does the popularity (number of on-links) of a post decay with its age. Figure 7.7(c) depicts δ on the horizontal axis and it depicts the overall number of links that were created δ time-ticks after the publication of the post it links to on the vertical axis. Again, note that the power law discovered in [138] is matched more closely by our model $\mathcal{ZC}$ than by the $\mathcal{EXP}$ model.

Where does this power law come from in our model $\mathcal{ZC}$? A blogger chooses a post of a blog by its recency and its number of in-links, that is, the probability is given by normalized ratio of number of in-links and the time difference since the publication of the post. Since a



(a) IFD: Entropy Plot (b) IPT: Inter-Posting Times (c) PP: Popularity over Time

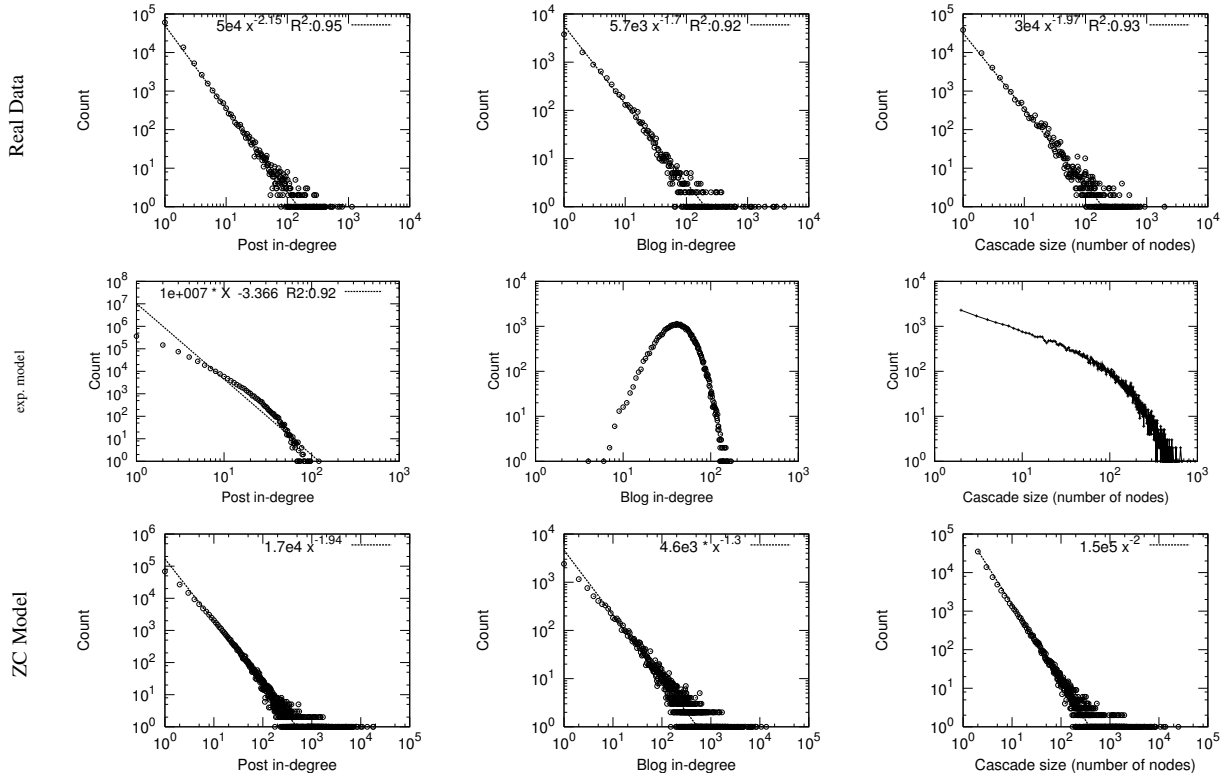
Figure 7.7: Temporal patterns of the blogosphere: (a) real data, (b) $\mathcal{EXP}$ model, and (c) the blogosphere as modeled by the $\mathcal{ZC}$ model. Notice $\mathcal{ZC}$ model outperforms $\mathcal{EXP}$ model and matches the temporal characteristics of real blogosphere.

blog publishes at most one post per time-tick it follows that the PDF of the time differences that occur in that selection of posts is the time difference multiplied by the number of in-links of the corresponding post. Globally, a power law distribution of time differences emerges that matches the real data.

Topological Patterns. Figure 7.8 shows that the power laws in the distribution of the blog and post in-degree, and the Cascade Size power law, found by [138] are matched closely by our $\mathcal{ZC}$ model. Not only $\mathcal{ZC}$ matches the shape perfectly, but it also matches the power law exponents well: -1.94 versus -2.15 for the BID in Fig. 7.8(a); -1.3 versus -1.7 for the PID in Figure 7.8(b); and -2 versus -1.97 for the Cascade Size power law in Figure 7.8(c). In contrast $\mathcal{EXP}$ model only somewhat mimics the PID power law.[2]

Where do the power laws come from in our model $\mathcal{ZC}$? The power laws of the in-degree distributions can be explained by the fact that a blog A in the "exploration" mode chooses another

[2]The power law comes out more clearly if the model is run for longer time.



(a) PID: Post In-Degree (b) BID: Blog In-Degree (c) SCT: Size of Cascades

Figure 7.8: Topological patterns of the blogosphere. Top: real blogosphere; Middle: $\mathcal{EXP}$ model; Bottom: blogosphere as modeled by the $\mathcal{ZC}$ model. Notice $\mathcal{ZC}$ model outperforms $\mathcal{EXP}$ model and matches the properties of real blogosphere.

blog B in order to link to a post published by B based on the number of B's in-links, which causes a rich-get-richer phenomenon. This phenomenon leads to a power law distribution. Similarly, a blog A publishing a post chooses another post P to create a link to P based on number of P's in-links. Again, the resulting rich-get-richer phenomenon leads to a power law distribution.

Moreover, the $\mathcal{ZC}$ model also matches the power law of the distribution of the cascade sizes (SCT) which is more surprising. Our model $\mathcal{ZC}$ is the first blog model that matches this power law. The power law exponents are almost the same (-2 versus -1.97).

7.3 Summary of models and contributions

Each of the proposed models, TI-MODEL, Cascade Generation model, and $\mathcal{ZC}$ have their uses.

The Time-Identity model is useful for producing patterns found in online groups, and is also easy to fit to real data with EM algorithm. This fitting will help allow us to characterize different groups and compare the data.

We use the Cascade Generation Model to show that, given a network, an epidemiological model does well in reproducing realistic cascade patterns. This paves the way toward the $\mathcal{ZC}$

Model \ Property	*Temporal properties*			*Cascade properties*								*Other*		
	Inter-posting time	In-link decay	Burstiness	Realistic shapes	Realistic sizes	Realistic degree	Realistic per-level degree	Realistic size vs. depth	Collisions	Unique Authors Power Law	Author Activity Power Law	Blog-like	Group-like	Builds network
Branching Process	X	X	X	?	?	X	?	X	?	X	X	X	X	X
Time-Identity	X	?	?	?	?	✓	✓	X	?	✓	?	X	✓	X
Cascade Generation	?	X	X	✓	✓	✓	?	?	?	?	?	✓	X	X
BlogModel	✓	✓	✓	?	✓	✓	?	?	?	?	?	✓	X	✓

Table 7.2: A summary of properties exhibited by various models. A "?" indicates that experiments have not been performed.

model. $\mathcal{ZC}$ uses random walks and recursive propagation, and reproduces not only the cascade patterns, but also temporal patterns, and does not require a pre-existing network structure.

The reproduction of properties is summarized in Table 7.2.

Part III

Network effects in action

Having explored structural patterns in networks and interactions, what are some real-world effects of network behavior? We next explore three applications. The first is a direct application of network topology, with the goal of finding anomalous nodes in networks. Based on findings in Chapter 3, we apply patterns to the local networks around nodes, or "ego-nets," to pinpoint anomalous nodes.

The second application uses network information and ideas of propagation to label nodes in a network. We have a particular application of finding risky accounts in the general ledger of a company. We propose SNARE, a method that uses flagging and belief propagation to produce a ranked list of the "riskiest" nodes. We also show that this method is generalizable to other labeling tasks, such as genres of blogs.

Finally, we address another ranking problem, that of ranking products and merchants in a data set of online reviews. The reviews essentially produce a labeled bipartite graph between review authors and review objects. We explore several statistical methods for ranking the products in an attempt to provide a ranking that works better than average rating. We also propose a unique evaluation method.

Chapter 8

Oddball: Anomaly detection

PROBLEM STATEMENT: *Given a knowledge of patterns in real networks, how can we identify anomalous nodes in a tractable, accurate, and explainable manner?*

Given a large, weighted graph, how can we find anomalies? Which rules should be violated, before we label a node as an anomaly? We propose the `OddBall` algorithm, to find such nodes. The contributions are the following: (a) we discover several new rules (power laws) in density, weights, ranks and eigenvalues that seem to govern the so-called "neighborhood sub-graphs" and we show how to use them for anomaly detection; (b) we carefully choose features, and design `OddBall`, so that it is scalable and it can work un-supervised (no user-defined constants) and (c) we report experiments on many real graphs with up to *1.6 million* nodes, where `OddBall` indeed spots unusual nodes that agree with intuition.

8.1 Introduction

Given a real graph, with weighted edges, which nodes should we consider as "strange"? Applications of this setting abound: For example, in network intrusion detection, we have computers sending packets to each other, and we want to know which nodes misbehave (e.g., spammers, port-scanners). In a who-calls-whom network, strange behavior may indicate defecting customers, or telemarketers, or even faulty equipment dropping connections too often. In a social network, like Facebook and LinkedIn, again we want to spot users whose behavior deviates from the usual behavior, such as people adding friends indiscriminately, in "popularity contests."

The list of applications continues: Anomalous behavior could signify irregularities, like credit card fraud, calling card fraud, campaign donation irregularities, accounting inefficiencies or fraud [22], extremely cross-disciplinary authors in an author-paper graph [195], network intrusion detection [190], electronic auction fraud [49], and many others.

In addition to revealing suspicious, illegal and/or dangerous behavior, anomaly detection is useful for spotting rare events, as well as for the thankless, but absolutely vital task of data cleansing [58]. Moreover, anomaly detection is intimately related with the pattern and law discovery: unless the majority of our nodes closely obey a pattern (say, a power law), only then can we confidently consider as outliers the few nodes that deviate.

Most anomaly detection algorithms focus on clouds of multi-dimensional points, as we will describe in the survey section. Our goal, on the other hand, is to spot strange nodes in a *graph*, with weighted edges. What patterns and laws do such graphs obey? What features should we extract from each node?

We propose to focus on neighborhoods, that is, a sphere, or a ball (hence the name `OddBall`) around each node(the *ego*): that is, for each node, we consider the induced sub-graph of its neighboring nodes, which is referred to as the *egonet*. Out of the huge number of numerical features one could extract from the egonet of a given node, we give a carefully chosen list, with features that are effective in revealing outliers. Thus, every node becomes a point in a low-dimensional feature space.

Main contributions in this chapter are:

1. *Egonet patterns*: We show that egonets obey some surprising patterns (like the *Egonet Density Power Law* (*EDPL*), *EWPL*, *ELWPL*, and *ERWPL*), which gives us confidence to declare as outliers the ones that deviate. We support our observations by showing that the *ERWPL* yields the *EWPL*.
2. *Scalable algorithm*: Based on those patterns, we propose `OddBall`, a scalable, unsupervised method for anomalous node detection.
3. *Application on real data*: We apply `OddBall` to numerous real graphs (DBLP, political donations, and other domains) and we show that it indeed spots nodes that a human would agree are strange and/or extreme.

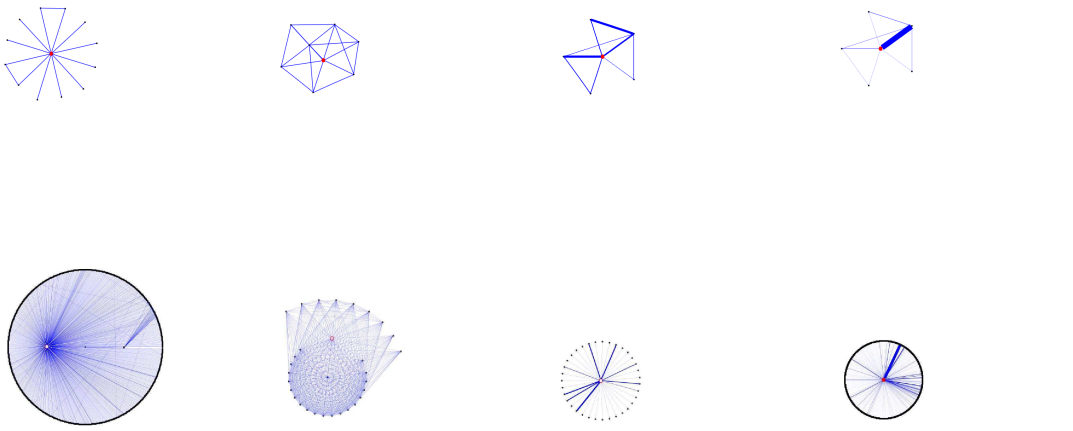
Of course, there are numerous types of anomalies. For a full list, see [7]; here we focus on only the following major types (see Figure 8.1 for examples and Section 8.3 for the dataset description):

1. *Near-cliques* and *stars*: Those nodes whose neighbors are very well connected (near-cliques) or not connected (stars) turn out to be "strange": in most social networks, friends of friends are often friends, but either extreme (clique/star) is suspicious.
2. *Heavy vicinities*: If person i has contacted n distinct people in a who-calls-whom network, we would expect that the number of phone calls (weight) would be proportional to n (say, $3n$ or $5n$). Extreme total weight would be suspicious, indicating, e.g., faulty equipment that forces redialing.
3. *Dominant heavy links*: In the who-calls-whom scenario above, a very heavy single link in the 1-step neighborhood of person i is also suspicious, indicating, e.g., a stalker that keeps on calling only one of his/her contacts an excessive count of times.

The upcoming sections are as follows: We describe the datasets; the proposed method and observed patterns; the experimental results; prior work; and finally the conclusions.

8.2 Related Work

We will be using several ideas based on work in Chapter 3, but two more areas of interest are also highly relevant to this work: outlier detection and graph anomalies.



(a) Near-star (b) Near-clique (c) Heavy vicinity (d) Dominant edge

Figure 8.1: Types of anomalies that `OddBall` detects. Top row: toy sketches of egonets (ego shown in larger, red circle). Bottom row: actual anomalies spotted in real datasets. (a) A near-star in *PostNet*: `instapundit.com/archives/025235.php`, an extremely long post on Hurricane Katrina relief agencies with numerous links to diverse other posts about donations. (b) A near-clique in *PostNet*: `sizemore.co.uk`, who often linked to its own posts, as well as to its own posts in other blogs. (c) A heavy vicinity in *PostNet*: `blog.searchenginewatch.com/blog` has abnormally high weight w.r.t. the number of edges in its egonet. (d) Dominant edge(s) in *Com2Cand*: In FEC 2004, George W. Bush received a huge donation from a single committee: Democratic National Committee (~$87M)(!) - in fact, this amount was *spent against* him; next heaviest link (~$25M): from Republican National Committee.

8.2.1 Outlier Detection

Outlier detection has attracted wide interest, being a difficult problem, despite its apparent simplicity. Even the definition of the outlier is hard to give: For instance, Hawkins [99] defines an outlier as "an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism." Similar, but not identical, definitions have been given by Barnett and Lewis [20], and Johnson [112].

Outlier detection methods form two classes, *parametric* (statistical) and *non-parametric* (model-free). The former includes statistical methods that assume prior knowledge of the underlying data distribution [20, 99]. The latter class includes *distance-based* and *density-based* data mining methods. These methods typically define as an outlier the (n-D) point that is too far away from the rest, and thus lives in a low-density area [125]. Typical methods include LOF [40] and LOCI [177]. These methods not only flag a point as an outlier but they also give outlierness

scores; thus, they can sort the points according to their "strangeness." Many other *density-based* methods especially for large high-dimensional data sets are proposed in [4, 13, 50, 81]. Finally, most clustering algorithms [48, 106, 171] reveal outliers as a by-product.

8.2.2 Anomaly Detection in Graph Data

Noble and Cook [172] detect anomalous sub-graphs using variants of the *Minimum Description Length* (MDL) principle. Eberle and Holder [67] use MDL as well as other probabilistic measures to detect several types of anomalies (e.g. unexpected/missing nodes/edges). Frequent subgraph mining [110, 212] is used to detect non-crashing bugs in software flow graphs [145]. Chakrabarti [43] uses MDL to spot anomalous edges. Sun et al. [195] use proximity and random walks, to assess the normality of nodes in bipartite graphs. OutRank and LOADED [80, 163] use similarity graphs of objects to detect outliers.

In contrast to the above, we work with *unlabeled* graphs. We explicitly focus on nodes, while interactions are also considered implicitly as we study *neighborhood sub-graphs*. Finally, we consider both bipartite and unipartite graphs as well as edge *weights*.

8.3 Data Description

We studied several unipartite/bipartite, weighted/unweighted large real-world graphs in a variety of domains, described in detail in Table 8.1, as well as Section 2.3. Particularly, unipartite networks include the following: *PostNet* contains post-to-post links in a set of blogs[140], *Enron* contains emails at Enron collected from about 1998 to 2002 (made public by the Federal Energy Regulatory Commission during its investigation), and *Oregon* contains AS peering information inferred from Oregon route-views BGP data. Bipartite networks include the following: *Auth-Conf* contains the publication records of authors to conferences from DBLP, and *Don2Com* and *Com2Cand* are from the U.S. Federal Election Commission in 2004[1], a public record of donations between donors and committees and between committees and political candidates, respectively.

For *Don2Com* and *Com2Cand*, the weights on the edges are actual weights representing donation amounts in dollars. For the remaining weighted datasets, the edge weights are simply the number of occurrences of the edges. For instance, if post i contains k links to another post j, the weight of the edge $e_{i,j}$ is set to k.

In our study, we specifically focused on undirected graphs, but the ideas can easily be generalized to directed graphs.

8.4 Proposed Method

Borrowing terminology from social network analysis, "ego" is an individual node.

[1]Parsed dataset from all cycles can be found at `www.cs.cmu.edu/~mmcgloho/fec/data/fec_data.html`

Name	N	E	Weights	Structure	Description
PostNet	223K	217K	Yes	Unipartite	Network of posts based on citations
Auth-Conf	421K	1M	Yes	Bipartite	DBLP Author/Conference associations
Com2Cand	6K	125K	Yes	Bipartite	2004 US FEC Committee to Candidate donations
Don2Com	1,6M	2M	Yes	Bipartite	2004 US FEC Donor to Committee donations
Enron	36K	183K	No	Unipartite	Email associations at Enron
Oregon	11K	38K	No	Unipartite	AS peering connections

Table 8.1: Datasets studied in this work.

Informally, an ego (node) of a given network is anomalous if its neighborhood significantly differs from those of others. The basic research questions are: (a) *what features should we use to characterize a neighborhood?* and (b) *what does a 'normal' neighborhood look like?*

Both questions are open-ended, but we give some answers below. First, let's define terminology: the "k-step neighborhood" of node i is the collection of node i, all its k-step-away nodes, and all the connections among all of these nodes—formally, this is the "*induced sub-graph*." In SNA, the 1-step neighborhood of a node is specifically known as its "*egonet*".

How should we choose the value of k steps to study neighborhoods? Given that real-world graphs have small diameter [10], we need to stay with small values of k, and specifically, we recommend k=1. We report our findings only for k=1, because using $k > 1$ does not provide any more intuitive or revealing information, while it has heavy computational overhead, possibly intractable for very large graphs.

8.4.1 Feature Extraction

The first of our two inter-twined questions is *which statistics/features to extract* from a neighborhood.

There is an infinite set of functions/features that we could use to characterize a neighborhood (number of nodes, one or more eigenvalues, number of triangles, effective radius of the central node, number of neighbors of degree 1, etc etc). Which of all should we use?

Intuitively, we want to select features that (a) are fast-to-compute and (b) will lead us to patterns/laws that most nodes obey, except for a few anomalous nodes. We spend a lot of time experimenting with about a dozen features, trying to see whether the nodes of real graphs obey any patterns with respect to those features (see our technical report [7]). The majority of features lead to no obvious patterns, and thus we do not present them.

The trimmed-down set of features that *are very* successful in spotting patterns, are the following:

1. N_i: number of neighbors (degree) of ego i,
2. E_i: number of edges in egonet i,

3. W_i: total weight of egonet i,
4. $\lambda_{w,i}$: principal eigenvalue of the *weighted* adjacency matrix of egonet i.

The next question is how to look for outliers, in such an n-dimensional feature space, with one point for each node of the graph. In our case, n=4, but one might have more features depending on the application and types of anomalies one wants to detect. A quick answer to this would be to use traditional outlier detection methods for clouds of points using all the features.

In our setting, we can *do better*. As we show next, we group features into carefully chosen pairs, where we show that there are patterns of normal behavior (typically, power-laws). We flag those points that significantly deviate from the discovered patterns as anomalous. Among the numerous pairs of features we studied, the successful pairs and the corresponding type of anomaly are the following:

- E vs N: *CliqueStar*: detects near-cliques and stars
- W vs E: *HeavyVicinity*: detects many recurrences of interactions
- λ_w vs W: *DominantPair*: detects single dominating heavy edge (strongly connected pair)

8.4.2 Laws and Observations

The second of our research questions is *what do normal neighborhoods look like*. Thus, it is important to find patterns ("laws") for neighborhoods of real graphs, and then report the deviations, if any. In this work, we report some new, surprising patterns:

Observation 8.4.1 (*EDPL*: Egonet Density Power Law) *For a given graph $\mathcal{G}$, node $i \in V(\mathcal{G})$, and the egonet $\mathcal{G}_i$ of node i, the number of nodes N_i and the number of edges E_i of $\mathcal{G}_i$ follow a power law.*

$$E_i \propto N_i^{\alpha}, \;\; 1 \leq \alpha \leq 2.$$

In our experiments the *EDPL* exponent α ranged from 1.10 to 1.66. Fig. 8.2 illustrates this observation, for several of our datasets. Plots show E_i versus N_i for every node (green points); the black circles are the median values for each bucket of points (separated by vertical dotted lines) after applying logarithmic binning on the x-axis as in [153]; the red line is the least squares(LS) fit on the median points. The plots also show a blue line of slope 2, that corresponds to cliques, and a black line of slope 1, that corresponds to stars. All the plots are in log-log scales.

Observation 8.4.2 (*EWPL*: Egonet Weight Power Law) *For a given graph $\mathcal{G}$, node $i \in V(\mathcal{G})$, and the egonet $\mathcal{G}_i$ of node i, the total weight W_i and the number of edges E_i of $\mathcal{G}_i$ follow a power law.*

$$W_i \propto E_i^{\beta}, \;\; \beta \geq 1.$$

Fig. 8.3 shows the *EWPL* for (only a subset of) our datasets (due to space limit). In our experiments the *EWPL* exponent β ranged up to 1.29. Values of $\beta > 1$ indicate super-linear growth in the total weight with respect to increasing total edge count in the egonet.

Observation 8.4.3 (*ELWPL*: Egonet λ_w Power Law) *the principal eigenvalue $\lambda_{w,i}$ of the weighted adjacency matrix and the total weight W_i of $\mathcal{G}_i$ follow a power law.*

$$\lambda_{w,i} \propto W_i^{\gamma}, \;\; 0.5 \leq \gamma \leq 1.$$

Fig. 8.4 shows the *ELWPL* for a subset of our datasets. In our experiments the *ELWPL* exponent γ ranged from 0.53 to 0.98. γ=0.5 indicates uniform weight distribution whereas γ~1 indicates a dominant heavy edge, in which case the weighted eigenvalue follows the maximum edge weight. γ=1 if the egonet contains only one edge.

Observation 8.4.4 (*ERWPL*: Egonet Rank Power Law) *the rank $R_{i,j}$ and the weight $W_{i,j}$ of edge j in $\mathcal{G}_i$ follow a power law.*

$$W_{i,j} \propto R_{i,j}^{\theta}, \;\; \theta \leq 0.$$

The *ERWPL* suggests that edge weights in the egonet have a skewed distribution. This is intuitive; for example in a friendship network, a person could have many not-so-close friends (light links), but only a few close friends (heavy links).

Next we show that if the *ERWPL* holds, then the *EWPL* also holds. Given an egonet graph $\mathcal{G}_i$, the total weight W_i and the number of edges E_i of $\mathcal{G}_i$, let $\mathcal{W}_i$ denote the ordered set of weights of the edges, $W_{i,j}$ denote the weight of edge j, and $R_{i,j}$ denote the rank of weight $W_{i,j}$ in set $\mathcal{W}_i$. Then,

Lemma 4 *ERWPL implies EWPL, that is: If $W_{i,j} \propto R_{i,j}^{\theta}$, $\theta \leq 0$, then*

$$W_i \propto E_i^{\beta} \begin{cases} \beta = 1, \text{ if } -1 \leq \theta \leq 0 \\ \beta > 1, \text{ if } \theta < -1 \end{cases}$$

Proof 7 *For brevity, we give the proof for $\theta < -1$. Other cases are similar. Given that $W_{i,j} = cR_{i,j}^{\theta}$, $W_{min} = cE_i^{\theta}$, i.e. $c = W_{min}E_i^{-\theta}$. Then we can write W_i as*

$$\begin{aligned} W_i &= W_{min}E_i^{-\theta}\left(\sum_{j=1}^{E_i} j^{\theta}\right) \approx W_{min}E_i^{-\theta}\left(\int_{j=1}^{E_i} j^{\theta} dj\right) \\ &= W_{min}E_i^{-\theta}\left(\frac{j^{\theta+1}}{\theta+1}\Big|_{j=1}^{E_i}\right) = W_{min}E_i^{-\theta}\left(\frac{1}{-\theta-1} - \frac{1}{(-\theta-1)E_i^{-\theta-1}}\right) \end{aligned}$$

For large E_i and considering $\theta < -1$, the second term in parenthesis goes to 0. Therefore; $W_i \approx c'E_i^{-\theta}$, where $c' = \frac{W_{min}}{-\theta-1}$, and since $\theta < -1$, $\beta > 1$.

8.4.3 Anomaly Detection

We can easily use the observations since anomalous nodes would behave away from the normal pattern. Let us define the y-value of a node i as y_i and similarly, let x_i denote the x-value of node i for a particular feature pair $f(x, y)$. Given the power law equation $y = Cx^{\theta}$ for $f(x, y)$, we define the outlierness score of node i to be

$$\textit{out-line(i)} = \frac{max(y_i, Cx_i^{\theta})}{min(y_i, Cx_i^{\theta})} * log(|y_i - Cx_i^{\theta}| + 1)$$

Intuitively, the above measure is the "distance to fitting line." Here we penalize each node with both the *number of times* that y_i deviates from its *expected* value Cx_i^{θ} given x_i, and with the logarithm of the *amount* of deviation. This way, the minimum outlierness score becomes 0, for which the actual value y_i is equal to the expected value Cx_i^{θ}.

This simple and easy-to-compute method not only helps in detecting outliers, but also provides a way to sort the nodes according to their outlierness scores. However, this method is prone to yield false positives for the following reason: Assume that there exists some points that are far away from the remaining points but that are still located close to the fitting line. In our experiments with real data, we observe that usually happens for high values of x and y. For example, in Fig. 8.2(a), the points marked with left-triangles ($\triangleleft$) are almost on the fitting line even though they are far away from the rest of the points.

We want to flag both types of points as outliers, and thus we propose to combine our heuristic with a density-based outlier detection technique. We used LOF [40], which also assigns outlierness scores *out-lof(i)* to data points; but any other outlier detection method would do, as long as it gives such a score. To obtain the final outlierness score of a data point i, one might use several methods such as taking a linear function of both scores and ranking the nodes according to the new score, or merging the two ranked lists of nodes, each sorted on a different score. In our work, we simply used the sum of the two normalized(by dividing by the maximum) scores, that is, *out-score(i)* $=$ *out-line(i)*$+$*out-lof(i)*.

8.5 Experimental Results

CliqueStar

Here, we are interested in the communities that the neighbors of a node form. In particular, *CliqueStar* detects anomalies having to do with near-cliques and near-stars. Using *CliqueStar*, we were successful in detecting many anomalies over the unipartite datasets (although it is irrelevant for bipartite graphs since by nature the egonet forms a "star").

In social media data *PostNet*, we detected posts or blogs that had either all their neighbors connected (cliques) or mostly disconnected (stars). We show some illustrative examples along with descriptions from *PostNet* in Fig. 8.1. See Fig.8.2a for the detected outliers on the scatterplot from the same dataset.

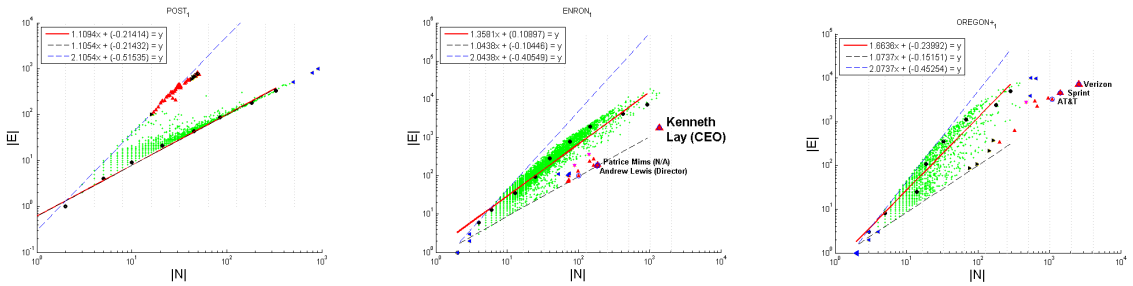
In *Enron*(Fig.8.2b), the node with the highest anomaly score turns out to be Kenneth Lay, who was the CEO and is best known for his role in the Enron scandal in 2001. Our method reveals that none of his over 1K contacts ever sent emails to each other.

In *Oregon* (Fig.8.2c), the top outliers are the three large ISPs (Verizon, Sprint and AT&T).

HeavyVicinity

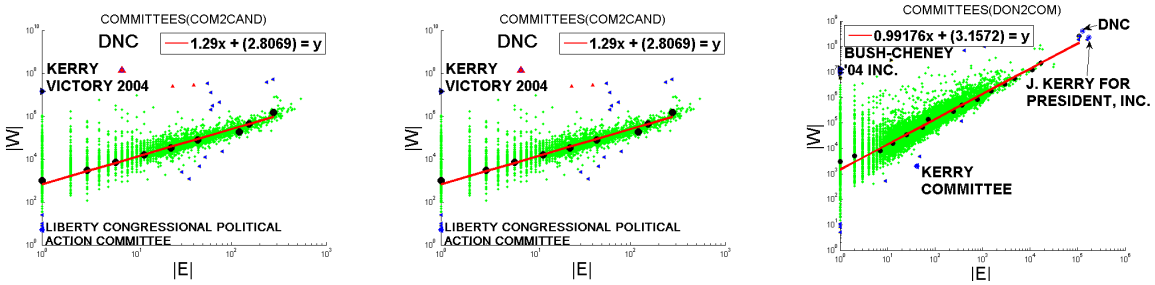
In our datasets, *HeavyVicinity* detected "heavy egonets," with considerably high total edge weight compared to the number of edges. We mark the anomalies in Fig.8.3 for several of our datasets. See [7] for results on all the datasets and further discussions.

In *Com2Cand*(Fig.8.3a), we see that Democratic National Committee gave away a lot of money compared to the number of candidates that it donated to. In addition, (John) Kerry Victory 2004 donated a large amount to a single candidate, whereas Liberty Congressional Political Action Committee donated a very small amount ($5), again to a single candidate. Looking at the *Candidates* plot for the same bipartite graph (Fig.8.3b), we also flagged Aaron Russo, the lone



(a) *PostNet* (b) *Enron* (c) *Oregon*

Figure 8.2: Illustration of the Egonet Density Power Law (*EDPL*), and the corresponding anomaly *CliqueStar*, with outliers marked with triangles. Edge count versus node count (log-log scale); red line is the LS fit on the median values (black circles); dashed black and blue lines have slopes 1 and 2 respectively, corresponding to stars and cliques. Most striking outlier: Ken Lay (CEO of Enron), with a star-like neighborhood. See Section 5.1.1 for more discussion and Fig.1 for example illustrations from *PostNet*.



(a) *Com2Cand* (b) *Com2Cand* (c) *Don2Com*

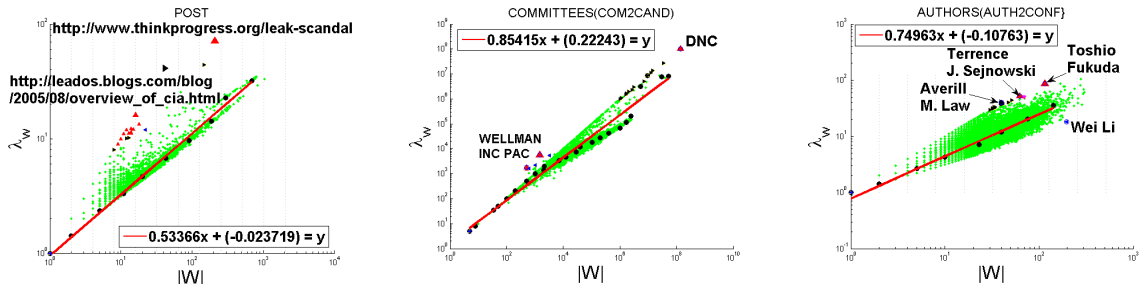
Figure 8.3: Illustration of the Egonet Weight Power Law (*EWPL*) and the weight-edge anomaly *HeavyVicinity*. Plots show total weight vs. total count of edges in the egonet for all nodes (in log-log scales). Detected outliers include Democratic National Committee and John F. Kerry (in FEC campaign donations). See Section 5.2.1 for more discussions.

recipient of that PAC. (In fact, Aaron Russo is the founder of the Constitution Party which never ran any candidates, and Russo shut it down after 18 months.)

In *Don2Com*(Fig.8.3c), we see that Bush-Cheney '04 Inc. received a lot of money from a single donor. On the other hand, we notice that the Kerry Committee received less money than would be expected looking at the number of checks it received in total. Further analysis shows that most of the edges in its egonet are of weight 0, showing that most of the donations to that committee have actually been returned.

DominantPair

Here, we find out whether there is a single dominant heavy edge in the egonet. In other words, this method detected "bursty" if not exclusive edges.



(a) *PostNet* (b) *Com2Cand* (c) *Auth2Conf*

Figure 8.4: Illustration of the Egonet λ_w Power Law (*ELWPL*) and the dominant heavy link anomaly *DominantPair*. Top anomalies are marked with triangles and labeled. See Section 5.2.2 for detailed discussions for each dataset and Fig.1 for an illustrative example from *Com2Cand*.

In *PostNet*(Fig.8.4a) nodes such as ThinkProgress's post on a leak scandal[2] and A Freethinker's Paradise post[3] linking several times to the ThinkProgress post were both flagged. On another note, the slope of the fitting line is close to 0.5, pointing to uniform weight distribution in egonets overall. This is expected as most posts link to other posts only once.

In *Com2Cand*(Fig.8.4b), Democratic National Committee is one of the top outliers. We would guess that the single large amount of donation was made to John F. Kerry. Counterintuitively, however, we see that that amount was spent for an opposing advertisement against George W. Bush.

DominantPair flagged extremely focused authors (those publish heavily to one conference) in the DBLP data, shown in Fig.8.3c. For instance, Toshio Fukuda has 115 papers in 17 conferences (at the time of data collection), with more than half (87) of his papers in one particular conference (ICRA). In addition, Averill M. Law has 40 papers published to the Winter Simulation Conference and nowhere else. On the other extreme, another interesting point is Wei Li, with many papers, who gets them published to as many distinct conferences, probably once or twice to each conference (uniform rather than 'bursty' distribution).

See [7] for results on all the datasets and further discussions.

8.5.1 Scalability

Major computational cost of our method is in feature extraction. In particular, computing those features, such as total number of edges and total weight, for the egonets is the bottleneck as one needs to find the induced 1-step neighborhood subgraphs for all nodes in the network.

The problem of finding the number of edges in the egonet of a given node can be reduced to the problem of triangle counting. One straightforward *listing* method for local triangle counting is the *Node-Iterator* algorithm. *Node-Iterator* considers each one of the N nodes and examines which pairs of its neighbors are connected. Time complexity of the algorithm is $O(Nd_{max}^2)$. Approximate streaming algorithms for local triangle counting can be applied to reduce the time complexity to $O(E \log N)$ with space complexity $O(N)$ [24]. Another recent method *Eigen-*

[2] `www.thinkprogress.org/leak-scandal`
[3] `leados.blogs.com/blog/2005/08/overview_of_cia.html`

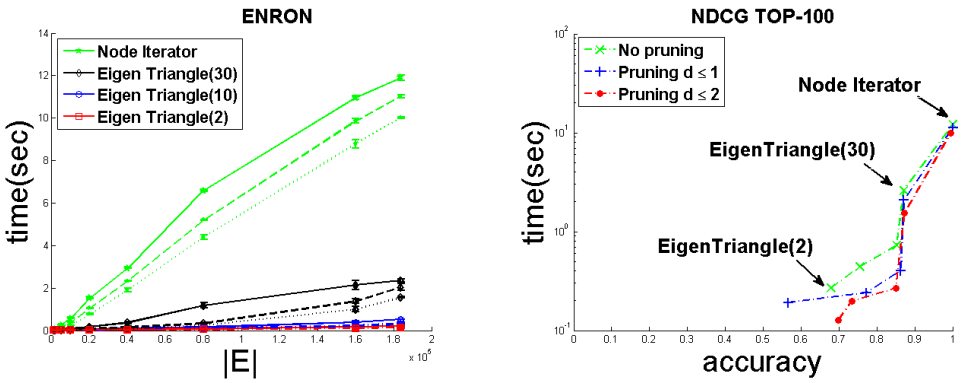


Figure 8.5: (a) Time vs. number of edges. Effect of pruning on computation time of counting triangles. Solid(–): no pruning, dashed(−−): pruning $d \leq 1$, and dotted(...): pruning $d \leq 2$ nodes. Computation time increases linearly with increasing number of edges, while decreasing with pruning. (b) Time vs. accuracy. Effect of pruning on accuracy of finding top anomalies as in the original ranking before pruning. New rankings are scored using Normalized Cumulative Discounted Gain. Pruning reduces time for both *Node-Iterator* and *Eigen-Triangle* for different number of eigenvalues while keeping accuracy at as high as~1 and~.9, respectively.

Triangle [199] uses eigenvalues/vectors to approximate paths of length three, i.e. local triangle counts, without actual counting.

To improve speed more, we propose pruning low degree nodes. In Fig.8.5a, we show computation time for *Node-Iterator* (green), and for *Eigen-Triangle* using 2(red), 10(blue), and 30(black) eigenvalues vs. graph size in terms of number of edges for Enron($E \simeq 180K$). Solid(–), dashed(−−), and dotted(...) lines are for no pruning, after pruning $d \leq 1$, and $d \leq 2$ nodes, respectively. We empirically note that time grows linearly with increasing graph size and also reduces with pruning. (Experiments ran on a Pentium class workstation, with 16GB of RAM, running Linux Fedora Core. To account for possible variability due to system state, each run is repeated 10 times and execution time results are averaged. Error bars show the variance across repeated runs.)

While increasing speed, one might wonder how pruning affects accuracy. To measure how rankings changed after pruning compared to the rankings without pruning, we used Normalized Discounted Cumulative Gain(NDCG) which is prevailingly used in Information Retrieval for measuring the effectiveness of search engines. Fig.8.5b shows time vs. NDCG scores for *Eigen-Triangle* using 2, 5, 10 and 30 eigenvalues, and also *Node-Iterator* for top k anomalies. For brevity, we only show ranking scores for k=100. *, +, and o symbols represent no pruning, pruning $d \leq 1$, and $d \leq 2$ nodes, respectively. Notice that pruning low degree nodes decreases computation time, while keeping the accuracy at as high as~.9 for *Eigen-Triangle*(30), and~1 for *Node-Iterator*.

8.6 Summary of Contributions

This is one of the few studies that focus on anomaly detection in graph data, including weighted graphs. We propose to use "egonets," that is, the induced sub-graph of the node of interest and its neighbors; and we give a small, carefully designed list of numerical features for egonets. The major contributions are the following:

1. Discovery of new patterns that egonets follow, such as patterns in density (*EDPL*), weights (*EWPL*), principal eigenvalues (*ELWPL*), and ranks (*ERWPL*). Proof of Lemma 4, linking the *ERWPL* to the *EWPL*.
2. `OddBall`, a fast, un-supervised method to detect abnormal nodes in weighted graphs. Our method does not require any user-defined constants. It also assigns an "outlierness" *score* to each node.
3. Experiments on real graphs of over 1M nodes, where `OddBall` reveals nodes that indeed have strange or extreme behavior.

Chapter 9

SNARE: Detecting misstatements in accounting data

PROBLEM STATEMENT: *Given a network of interactions, and some (noisy) knowledge of the labels of a few of the nodes, can we label other nodes in the network? Can we apply this to a real problem of risk detection in accounting data?*

9.1 Introduction

Accounting irregularities, in which data are intentionally or unintentionally misrepresented, raise significant risk for corporations and investors. Settlement amounts awarded in investor lawsuits have been increasing [65], and so has the number of financial restatements in recent years [194]. Auditors undertake a variety of procedures to determine whether there is reasonable assurance that financial statements are fairly stated, so automated assistance for detecting risks of misstatement has the potential for making the audit process more efficient.

Most of the well-known techniques for detecting accounting irregularities, such as ratio analysis, operate at the financial statement level, a highly aggregated summary of a company's financial activity, and generally offer little useful guidance to an auditor beyond a broad indicator of risk at a company. We have been investigating analytics that operate at a much more detailed level, on the transactions recorded in a company's general ledger. Past methods in this domain [22] explored the potential of different classification methods, such as logistic regression, expectation-maximization, and naive Bayes, on individual accounts and transactions. In this chapter we show how exploiting the link structure between accounts has the potential to greatly increase the accuracy of classification methods while making only a few assumptions. We will be applying belief propagation algorithms and link analysis to identify the risk of irregularities in corporate accounting.

Furthermore, we will show that this method is highly flexible to other tasks. Different domains will have different sources of knowledge about nodes in a network; however, our method allows a simple setting for domain experts to input this information without an understanding of the details of the algorithm.

Our contributions are the following: We introduce SNARE (Social Network Analysis for Risk Evaluation), which detects related entities that may be overlooked by using individual risk scores, it extends a well-known algorithm for graphical models into a useful application, and it may be flexibly applied to different domains. We show how it can be applied to the detection of fraud risk in general ledger accounting data as well as typical graph-labeling tasks in other domains such as web data and social networks.

9.2 Related Work

Social networks have become more important as practitioners become increasingly aware of the significance of relations between entities in a network. It has been demonstrated that knowledge of social structure can allow one to help make inferences about an organization [25, 147], to identify individuals [102], or to predict adopters of consumer products [103]. Related work has used knowledge of social structures for detecting securities fraud [167]. The authors later improved the approach by showing that one can often infer links that are not explicitly stated [77], and successfully extended the methods using inferred knowledge [71].

Semi-supervised learning methods may also be useful for graph labeling, as addressed in [216]. Finding authority of a node is one specific labeling task addressed in the literature. One way of defining the authority of a node in a network is its "reputation for knowledge"; that is, how reliable the source is. Guha et al. extend many of these ideas for reputation networks applied to eBay or Epinions [96]: rather than simply trusting someone's knowledge of a topic, one may also trust another's reliability as a seller on eBay or a recommender on Epinions. The authors use matrix methods and model a "web of trust," where both trust and distrust are propagated over edges (with different patterns of propagation). They were able to predict trust between individuals given a small amount of labeled data.

HITS[123] and Pagerank[174] address reputation for webpages. Other methods of propagation of trust and distrust are discussed in Ziegler et al. [217], particularly in relation to trust on the semantic web.

Other work identifies particular anomalous patterns and seeks to spot them in large graphs. Pandit et al. introduce *NETPROBE*, which uses belief propagation to model eBay as a tripartite network of "fraudsters," "honest users," and "accomplices." Upon deciding on this model, they then use loopy belief propagation to assign probabilities of each node being in the three states [176], by detecting bipartite cores. (Our work extends this idea by using node features to bias the initial beliefs for each node, rather than relying only on the graph structure to classify the nodes.)

Many risk detection methods approach the problem by attempting to detect suspicious behavior in users. This approach has been successful for cellular phone fraud, where a caller's patterns are often disrupted by periods of inactivity. Here, most fraud schemes follow certain signatures, such that a rule-based system have lead to some successes [72]. Rule-based approaches have also been applied to the detection of money laundering[38]. A survey of related methods can be found in [37].

The literature contains many methods for detecting accounting irregularities which typically use a model-based approach [27, 28, 37, 60, 94]. However, many of these traditional approaches are limited by factors such as the diversity of fraud schemes, errors present in the training data,

and access only to aggregated financial statement data instead of detailed transactions. To counter this problem, in previous work, authors of [22] set up a system called Sherlock[1] for detecting errors and fraudulent behavior in general ledger data. Sherlock used classification methods for identifying suspicious accounts, by evaluating a set of features measuring different types of unusual activity. Methods such as naive Bayes, expectation-maximization, and logistic regression were used and compared. This work will approach the same problem of identifying accounts with high fraud risk from a social network analytic perspective.

This is the first work, to our knowledge, that has adapted generalized belief propagation to the accounting domain, and provided a framework to extend it into other domains for node labeling, incorporating both node and edge information. In this work, we are using data where all true labels are unknown from the start, and our results are verified by human investigation.

9.3 Proposed Method

We will address the following problem:

Given:

- A graph $G = (V, E)$, where entities (persons, accounts, blogs, etc.) are represented as vertices, or nodes, in the graph, and interactions (phone calls, account transactions, hyperlinks) between them are represented as edges.
- Binary class (state) labels $X = \{x_1, x_2\}$ defined on V.
- A set of features for each node $v_i \in V$, based on node attributes (geographic location, name, etc.)

Output: A mapping $V \rightarrow X$ from nodes to class labels.

The labels X are binary categorical variables derived from the context (normal or irregular, conservative or liberal, etc.). We also note that while nodes and links can be related to social entities such as persons and relations or actions, the proposed methods can be applied to any sort of entities, such as accounts or webpages.

The basic premise of SNARE is to use neighboring labels to classify a given node. This premise has proved effective for many graph labeling tasks [108]. However, we also take into account domain knowledge, by assigning an initial risk scores to nodes prior to evaluating neighborhood associations between them. To measure risk by association, we then use *belief propagation* for passing risk to connected nodes. A detailed tutorial of belief propagation may be found in work by Yedidia [213].

Let us summarize the procedure. In a network for a given task, the true label for each node v_i is unknown. We are, however, given some local observations about the node, which we use as a local estimation of its risk, or *node potential* $\phi_i(x_c)$ of v_i for class x_c (the procedure for determining this will be described shortly). Information about this node is inferred from the surrounding nodes. This is obtained through iterative message passing to and from v_i to each neighbor v_j, where a message from v_i to v_j with its own assessment of v_j's believed class is denoted by m_{ij}. At the end of the procedure, the *belief* of a node v_i belonging to in class x_c is

[1]Sherlock is research in progress. As such, the methods we describe should not be interpreted as descriptive of PwC's current standard practice in analyzing general ledger data.

determined. The belief is an estimated probability, which can be thresholded into the classes (e.g. a $b_i(x_c) > .5$ implies v_i belongs to class x_c), or used relatively to compare risk scores between nodes (e.g. $b_i(x_c) > b_j(x_c)$ implies v_i is more likely to belong to x_c than v_j).

In more detail, messages are obtained the following way. Each edge e_{ij} has associated messages $m_{ij}(x_c)$ and $m_{ji}(x_c)$ for each possible class. $m_{ij}(x_c)$ is a message that v_i sends to v_j about v_j believed likelihood of belonging to x_c. Iteratively, messages are updated using the sum-product algorithm. Each outgoing message from a node to a neighbor is updated according to incoming messages from the node's other neighbors. Formally, the message-update equation is as follows:

$$m_{ij}(x_c) \leftarrow \sum_{x_d \in X} \phi_i(x_d)\psi_{ij}(x_d, x_c) \prod_{k \in N(i) \setminus j} m_{ki}(x_d) \tag{9.1}$$

where $N(v_i)$ is the set of neighboring nodes to v_i. $\psi_{ij}(x_c, x_d)$ is the *edge potential* of an edge between two nodes i, j of classes x_c and x_d. $\psi_{ij}(x_c, x_d)$ is generally large if edges between x_c and x_d occur often, and small if not. Order of message-passing does not matter, provided all messages are passed in each iteration. We also normalize $m_{ij}(x_c)$ to avoid numerical underflow, as discussed in [56], so each edge's message vector sums to one: $\sum_c m_{ij}(x_c) = 1$.

Convergence occurs when the maximum change between any message between iterations is less than some value (in our experiments 10^{-6}). Convergence is not guaranteed in general graphs (only for trees), but typically occurs in practice. Upon convergence, belief scores are determined by the following equation:

$$b_i(x_c) = k\phi_c(v_i) \prod_{v_j \in N(v_i)} m_{ji}(x_c) \tag{9.2}$$

where k is a normalizing constant (beliefs for each class must sum to 1).

Adapting the message passing algorithm to our purposes has the following challenge: Find an effective yet intuitive way to choose node and edge potentials. We use two main concepts, *homophily* over edges and *node attributes* to influence probability of different classes.

For purposes of explanation, we will have two classes, x_R for "risky" and x_{NR} for "non-risky." We will subsequently refer to $b_i(x_R)$ is the end probability of a node being risky after completion of the algorithm. A node with $b_i(x_R) = 1$ is certainly suspect, and $b_i(x_R) = 0$ is not suspect; most nodes will fall somewhere in between, on the continuum. SNARE will then produce a ranked list of the "risky" nodes, as candidates for further investigation.

For the edge potential term $\psi_{ij}(x_c, x_d)$ in the message-passing equations, we chose an identity function with a noise parameter ϵ. That is, if v_i is risky, v_j has a high probability of being risky, while allowing for some variance. The transition matrix is shown formally in Table 1.

$\psi_{ij}(x_d, x_c)$	$v_i = x_{NR}$	$v_i = x_R$
$v_j = x_{NR}$	$1 - \epsilon$	ϵ
$v_j = x_R$	ϵ	$1 - \epsilon$

Table 9.1: Transition matrix, or edge potentials for belief propagation.

Before beginning the message passing procedure, however, we must also assign a *node potential* to each individual node. The node potential represents the risk of a node without considering information from its neighbors. The initial node potential depends on the assumed distribution of class labels. When classes are evenly divided, default values $(\phi(x_{NR}), \phi(x_R)) = (0.5, 0.5)$ may be appropriate, while in cases where risk is sparse (as in most anomaly-detection domains) more skewed values such as $(\phi(x_{NR}), \phi(x_R)) = (0.9, 0.1)$ may be more reasonable.

However, a key component of SNARE is that the initial node potential is determined for each individual node by an process that can incorporate prior knowledge into the algorithm, for example in form of domain knowledge. In most domains where fraud is a challenge, there is rich information available about the potential fraudsters, such as geographic location, patterns of activity, or other features that suggest suspicious behavior. Therefore, we adjust node potential by assessing the risk to each individual node. There are many ways of using node features to adjust the initial potential: in fact one can treat assigning initial potential as a classification problem in itself. For our purposes we chose to use the features as *flags*, working from the assumption that each individual feature can function as a signal without taking into account other features. A node may be flagged for having several different types of suspicious behavior, and the domain expert may assign different severity to these flags. Where applicable we chose to use additive risk, increasing with a sigmoid function:

$$F_i = \frac{1}{1 + exp(-1 * f_i)} \tag{9.3}$$

where f_i is the total flagged risk, summed for all potential causes for suspicion. The node potential for node i, then, is $\phi_i(R) = F_i$ and $\phi_i(NR) = 1 - F_i$.[2]

When a node is highly flagged it also sends a stronger risk signal to its neighbors. However, if a flagged node's neighbors all have a low initial probability of being risky, the flagged node will be dampened. This is a reasonable action, since isolated flags are more likely to occur in error.

One key advantage of SNARE is that it will find risky associated nodes. Fraud schemes as they occur in accounting often involve many accounts, which often allow fraudsters to hide their actions. Since each account may have a very small risk score associated with it, traditional methods may not pinpoint the accounts as abnormal. However, SNARE will use the fact that the accounts interact with each other, and raise the associated risk of each account, allowing experts to more easily find the fraudulent behavior.

Since the flags are determined by the domain expert, this procedure can be successful on a wide variety of node labeling tasks, as we will show in the next section.

[2]It may be possible to learn the appropriate flag increments through machine learning techniques; this is left for future work.

Data	Problem description	Size (Nodes, Edges)	Classes	Flags
General Ledger1	Identifying misstated accounts from a general ledger.	$1,380$ accounts, $3,820$ edges (edge occurs if transaction)	$1,354$ Normal, 26 Misstated	Expert-identified flags of certain suspicious behaviors, $11,532$ flags total on the $1,380$ accounts.
General Ledger2	Identifying misstated accounts from a general ledger.	$1,678$ nodes, $18,720$ edges	$1,305$ Normal and 373 Misstated (noisy labeling, see Sec. 9.4.1).	Same as *GeneralLedger1*, $11,401$ flags total on the accounts.
Political Blogs	Labeling political affiliation of blogs.	$1,224$ blogs joined by hyperlinks	636 Conservative and 558 Liberal	220 flags total, 171 unique blogs with nonzero flags. Blogs flagged based on key substrings in blog domain name.
Campaigns	Correctly classifying political candidates on a bipartite network of candidates and political action committees.	(2004 cycle) $1,357$ nodes, $11,334$ edges. Edge occurs if there was a donation from committee to candidate.	Republican or Democrat	Flags were on stated class of committees, so candidate labels were acquired only through propagation.

Table 9.2: Descriptions of data and corresponding labeling problems.

9.4 Experimental Results

We developed SNARE to help detect risks in accounting data, so we will primarily evaluate it on its ability to find misstated accounts in a company's general ledger.[3] However, since the general ledger data is proprietary, and because we believe SNARE is more generally useful, we also evaluate its performance for graph labeling using public data from social media and political campaigns. A description of the data and the problems addressed may be found in Table 9.2.

9.4.1 Detecting misstated general ledger accounts

The general ledger of a company is an accounting record that summarizes its financial activity with double-entry bookkeeping. Within every general ledger is a set of accounts which can be thought of as variables representing the allocation of monetary resources. Business events, such as the purchase of machinery, would result in a transaction that reduces the value of the the cash

[3]Some of the terminology we use here is for the purpose of conducting research in the area of accounting and is by necessity highly simplified and abbreviated. It not descriptive of how PricewaterhouseCoopers analyzes general ledgers.

account but increases the value in the fixed asset account by an equivalent amount. The general ledger is used to prepare the financial statements by aggregating the balances of the accounts and thus auditors are extremely interested in finding misstatements in this data.

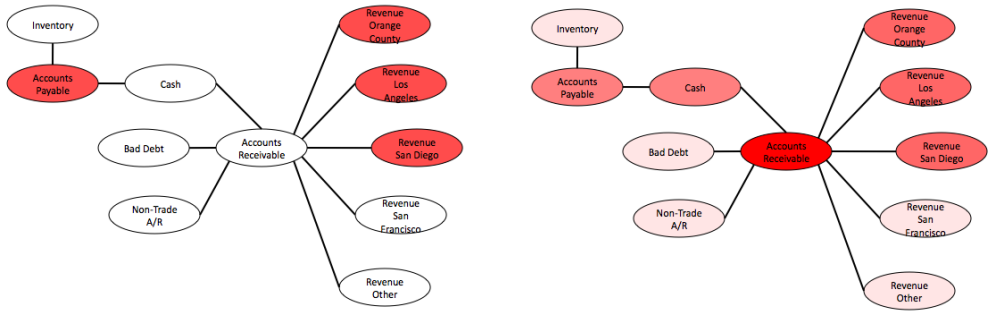
Manipulation of records can be found by experts on both the general ledger and financial statement level. There are many different fraud schemes [89, 210] for which experts have identified "red flags" that indicate suspicious behavior based on domain knowledge [60, 89, 166, 187]. For example, one fraud scheme is known as *channel stuffing*. In order to meet earnings expectation, fictitious sales are recorded to increase the revenue for the current quarter. These sales are typically not complete and are recorded solely to meet the earnings target. The company overloads their distribution channels to make it appear as if additional sales have been completed. This helps the company appear to meet its target. Such channel stuffing is usually followed by an increase in the number of returns at the beginning of the next quarter. In the general ledger, one could record the return of a sale by debiting revenue and crediting accounts receivable; thus to look for channel stuffing one might create a threshold test or red flag that highlights an account when there are an excessive number of these transactions.

In practice however, the creation of such a flag to detect channel stuffing or other schemes is fraught with difficulty and pitfalls. For instance with our example of channel stuffing one would need to determine what is an excessive amount of returns since some will always occur for normal business reasons. Setting the threshold too high could result in missing potential frauds, but setting the threshold too low could result in too many false positives. Furthermore, people who intentionally manipulate the general ledger are often well aware of the red flags used by auditors and actively attempt to avoid detection. Thus, for example, they may try to hide the activity by spreading the returns over many accounts so as to not set off any thresholds. Our hope with SNARE is that we could set the thresholds relatively low so as to be more sensitive to risky activity and use belief propagation to aggregate risk in the network to identify misstated accounts with a low false positive rate.

To analyze general ledger data with SNARE we first need to create a network with nodes, edges, and initial risks. For our application, we construct the network as follows:

- Each account in the general ledger becomes a node in the network.
- For every pair of accounts (X, Y) in the general ledger, they are connected with an edge if there are transactions where the sum of the amounts debiting X and crediting Y exceeds a minimum threshold.
- The initial risks on the nodes is determined by performing a preliminary scan over the data to detect red flags as determined by domain experts. The red flags are given equal weight and taken together they determine the initial risk as defined by Equation 3.

For example, Figure 9.1 shows a partial network with nodes for accounts receivable, accounts payable, bad debt, non-trade A/R, and several revenue accounts. In our example of channel stuffing, thresholds for our red flags could be set low enough to flag multiple revenue accounts and SNARE would then propagate the risk to accounts receivable where the collected belief would be strong enough to implicate it. In the next two sections, we present results of SNARE on general ledgers with known misstatements and show that on real data it is effective at aggregating risk across the network.



(a) Flagged network (b) After propagation

Figure 9.1: An example network with general ledger accounts represented by nodes and edges connecting pairs of accounts with significant amounts debited/credited with each other, under a fraud scheme of *channel stuffing*. The left image shows flagged accounts in red (revenue accounts flagged by abnormal debits), before propagation. The image on the right is the relative risk scores based on beliefs after propagation. Notice that now, since *Accounts Receivable* had many flagged neighbors, it now has the highest risk in the network, while *Accounts Payable* had a lower relative risk, due to the influence of unflagged *Inventory*.

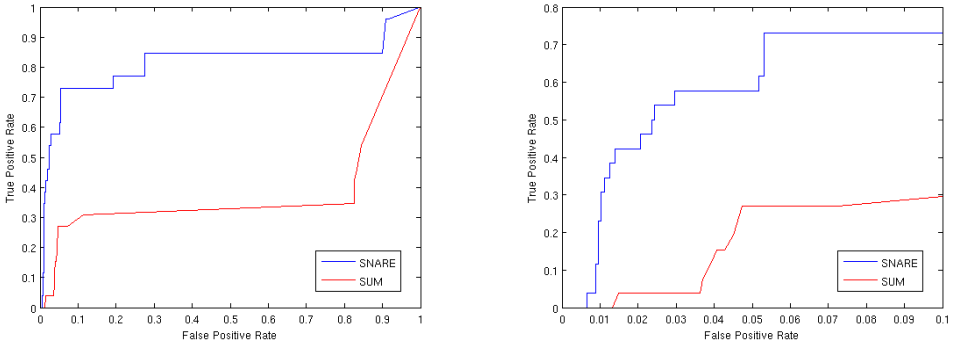


Figure 9.2: ROC curves for SNARE vs. SUM on *GeneralLedger1*. The first graph shows the entire range and the second shows performance for false positive rates of less than 0.1.

GeneralLedger1

In the first set of general ledger data there were a total of $1,380$ accounts, $3,820$ edges, and $11,532$ red flags (nearly every node had at least one flag). From prior domain knowledge, 26 accounts were identified as being misstated. We applied SNARE to this network and the message-passing process converged after 6 iterations. Our initial node potentials were $\phi_i(Risky) = 0.1$ and $\phi_i(NotRisky) = 0.9$ for a node i with no flags, and additional flags changed node potential according to Equation 9.3, so key information is in the nodes' number of flags relative to each other.

Figure 9.2 shows the ROC curve for the SNARE approach under the assumption that the 26 identified accounts was the complete set of true positives (and all other accounts are true negatives). In addition to SNARE , we plotted to ROC curve for a default approach based on simply ranking the accounts by the number of tests flagged. From the graph, we note that SNARE dom-

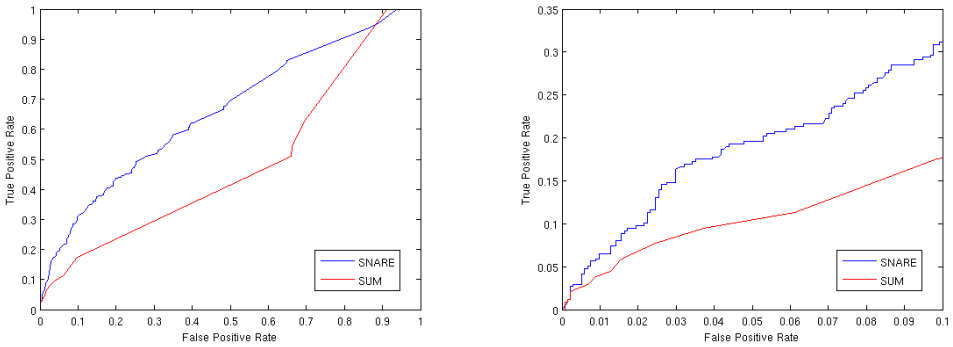


Figure 9.3: ROC curves for SNARE vs. SUM on *GeneralLedger2*. The first graph shows the entire range and the second shows performance for false positive rates of less than 0.1.

inated the default sum approach over all regions of the ROC curve. Furthermore, SNARE produced an extremely steep initial curve at low false positive rates. This is very promising as this is the region of the operating space most interesting from an application viewpoint.

GeneralLedger2

The second set of general ledger data contained $1,678$ nodes, $18,720$ edges, and $11,401$ red flags. Unfortunately, with this data set we had only coarse label information available that identified general groups of misstated accounts. For our experiments we treated all accounts in an identified group as being misstated, resulting in a total of 337 positive labels.

The results for *GeneralLedger2* are shown in Figure 9.3. The results are not as strong as for the previous general ledger, but this may be due to the noisy class labels. However, there is still significant improvement in the ROC curve compared with the default strategy of using the number of flags as a scoring mechanism.

Relevant non-proprietary risk-related data with a network structure is challenging to collect and institutions are reluctant to share data due to privacy concerns. Therefore, we will next show the use of SNARE for labeling nodes in using publicly available social network data.

9.4.2 Political blogs

The domain of social media presents the difficult task of automatically assessing political stance of a blog, news site, or other webpage. Doing so often requires analysis of sentiment in the text, which is both difficult and computationally expensive. Being able to do so by using the structure of the induced web graph can aid in this problem.

To this end, we tested SNARE on a network of political blogs, human-labeled as Conservative or Liberal. The data contained 758 Liberal blogs and 732 Conservative blogs, which were joined with edges based on hyperlinks made by the blog owners. (For details of building the network and labeling, see [1].) Of these, $1,224$ had degree greater than 0: 558 Liberal and 636 Conservative, which we chose to focus on for our experiments. The network was relatively dense, with $16,718$ total edges.

In this case, node information was noisy. We chose to flag nodes as more likely to be Conservative/Liberal based on substrings in the blog title. We chose the following flags, and indicate each substring's prevalence in blogs human-labeled as Conservative and Liberal.[4] Of the connected nodes, 171 had flags. Some blogs had multiple flags, so we used additive risk score.

String	Incidence	Flag
"con"	34 conservative, 9 liberal	+1
"right"	33 conservative, 2 liberal	+1
"rep"	19 conservative, 9 liberal	+1
"bush"	8 conservative, 6 liberal	+1
"lib"	11 conservative, 18 liberal	-1
"left"	3 conservative, 28 liberal	-1
"dem"	4 conservative, 28 liberal	-1
"kerry"	2 conservative, 6 liberal	-1

Since the number of Conservative and Liberal blogs was expected to be approximately equal, we used a default potential $(\phi(x_L), \phi(x_c) = \{0.5, 0.5\}$. With $\epsilon = 0.3$, 95% on nodes ($1,188$ of $1,247$) were classified correctly. An additional 233 nodes ended with a belief score $b_{con} = 0.5$, which we did not consider to be classified one way or the other (though most of them were Liberal). Most of these were isolated nodes; fewer than 20 had a degree greater than 0. For isolated nodes we simply classified them based on the flag, which was 0 in most nodes.

SNARE presented improvements over using the flag method alone or through clustering based on structure. Often times the flag was misleading, such as in the case of `laughatliberals.com` or `johnkerrymustlose.com`, but the edge effects usually allowed SNARE to correct the classification, without needing to do sentiment analysis on the words. On the other hand, there were occasions where a few blogs of one class formed a sort of "appendage" on the main cluster of the opposite class, which typical graph clustering methods would fail to identify but were successfully labeled using SNARE. One example of this is the two blogs `enemykombatant.blogspot.com` and `democratvoice.org`. The former blog was connected to the Conservative cluster, but the flag on the latter blog, its neighbor, propagated into it, correctly labeling both blogs as Liberal. This is shown in Figure 9.4.

In fact, most misclassifications occurred on cases of unflagged blogs of one class only bordering on blogs of the opposite class, and in cases along the middle between the two clusters. These cases would be difficult to classify using node information or edge information alone.

9.4.3 Political campaign contributions

While labeling political party membership for individuals running for office is not typically a challenge, we used it as a way to test our approach to labeling nodes by leveraging connection structure.

[4]Crawling the blogs themselves and using textual analysis would have potentially provided more accurate flags; however, we chose the more naive flag for experimental purposes, showing that even imperfect node information provides good results.

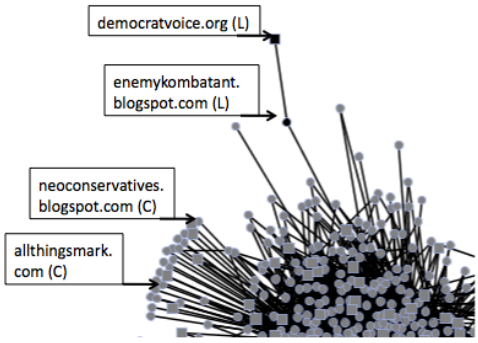


Figure 9.4: The political blog network, where human-labeled conservative blogs are shown in gray and liberal blogs shown in black. Flagged nodes (in either class) are shown as squares. This section highlights two outlier Liberal blogs connected to the cluster of Conservative blogs. Since `democratvoice` was flagged as Liberal, these two blogs were correctly classified with SNARE.

We took subsets of data from the United States Federal Election Commission [5] from the election cycles of 1980 through 2006, that listed donations from political action committees to political candidates for President, Senate, and House of Representatives. We then built a bipartite network of committees and candidates, creating edges between a committee and a candidate if a committee had, at some point, donated funds to the candidate. The largest cycle, 2004, contained $1,357$ nodes with positive degree (686 candidates and 671 committees) and $11,334$ edges. The classification task was to label a candidate as Democrat or Republican, based only on the committees it was connected to through donations.

Of the 671 committees, 583 were labeled with a party. We used these labels as flags (+1 or -1). From there, we ran SNARE on the bipartite graph to propagate labels to candidates. SNARE correctly labeled 659, mislabeled 12, and did not label 25, which gave an accuracy of 96 percent. With one exception (the earliest cycle, 1980, with an accuracy of 82%), all other cycles had above 90 percent accuracy.[6]

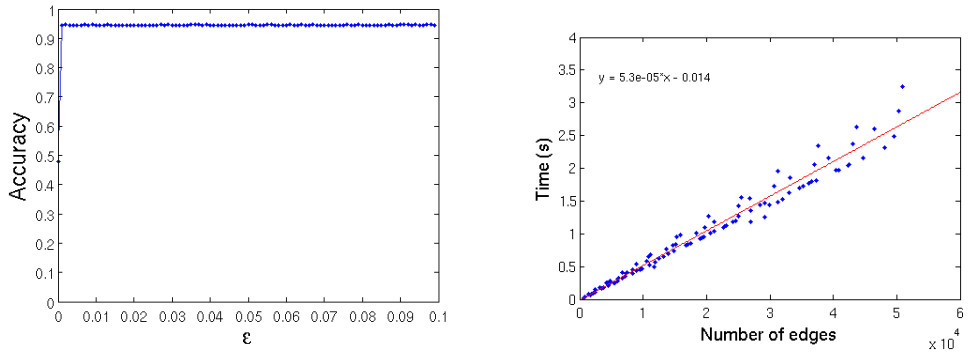
We find that varying parameters does not drastically affect accuracy, and the method is scalable to large graphs, as we will explain in the next section.

9.5 Analysis

We next demonstrate the robustness of SNARE to different parameter ranges, analyze its computational efficiency, and compare the accuracy to to spectral clustering on the task of graph labeling.

[5] `www.fec.gov/finance/disclosure/ftpdet.shtml`, downloadable in parsed format from `www.cs.cmu.edu/~mmcgloho/data.html`

[6] In fact, using very sparse flags (randomly selecting 10 committees from each class to flag) produced comparable results.



(a) (b)

Figure 9.5: (a) A demonstration of the robustness of SNARE, by varying the ϵ for *Political-Blogs* data, between 0 and 0.1, with the accuracy plotted on the y-axis. Note that even the smallest ϵ is effective. Accuracy results are similar for ϵ up to 0.5 (omitted to avoid redundancy). (b) Scalability results for *Campaigns* data: computation time vs. number of edges. SNARE scales linearly, with a 50,000 edge graph converging in under 3.5 seconds.

9.5.1 Sensitivity of parameters

SNARE is very robust and easy to use. Some domain knowledge is necessary for determining the node potential for both flagged and unflagged nodes. Default node potential is typically set at the expected percentage from each class (for example, $\{0.9, 0.1\}$ if one expects 90% of nodes in class 0 and 10% in class 1). Modifications of the sigmoid function tend to work well for additive risk for flagged nodes.

The edge potential parameter ϵ may be set in the range of $0 < \epsilon < .5$ without drastically affecting results. In *Campaigns*, we observed high sensitivity on the node potentials, and putting any bias on class tended to cause one class to dominate. This would seem natural, since the data were approximately split equally among the two classes, so any initial bias will dominate the final result. However, the ϵ parameter showed little sensitivity, and varying it between 0 and 0.5 affected results by less than 1 percent on both *Campaigns* and *PoliticalBlogs*. (Setting $\epsilon \geq .5$ would remove the homophily assumption, which would not be useful for tasks addressed here.) Figure 9.5(a) shows finer-grained results of varying parameters on blog data; even the smallest ϵ is effective, and accuracy does not change up to $\epsilon = 0.5$.

9.5.2 Computational performance

The most costly operation of SNARE occurs during the message-passing. Each iteration runs in $O(|E|)$ time, where $|E|$ is the number of edges in the network. Our experiments also reached convergence in relatively few iterations (less than 10 for all datasets). Other negligible computational costs are in assessing node potentials and calculating beliefs (both $O(N)$), and in all cases convergence occurred within 10 message-passing iterations.

Since the data varied in structure, we chose to run scaling experiments only on *Campaigns*. To sample, we took different window-sizes of election cycles, for every possible cycle, and timed

the completion of SNARE 100 times apiece. A plot of average time vs. number of edges in the graph is shown in Figure 9.5(b), including the best linear fit.

9.5.3 Comparison to existing work

To compare our performance to the state of the art, we also run spectral clustering on our data, which is an unsupervised method for node labeling. For *Campaigns* and *PoliticalBlogs* the data were already well-clustered, and visual analysis could cluster reasonably successfully. Spectral clustering, however, performed less well than SNARE even on these data sets.

On *PoliticalBlogs*, attempting to find two clusters failed. However, clustering results were better by allowing for a third cluster that did not fit with the other two. The two major clusters roughly corresponded to the conservative and liberal sectors. In full, of 1224 non-isolated blogs, 1133 were correctly classified. There were 83 misclassifications, and 8 in the third "undecided" cluster. This gave an accuracy of 92.5%, slightly less than SNARE.

On *Campaigns*, results were similar. There were two distinct clusters roughly corresponding to the parties. There were 617 correct classifications, 19 incorrect, and 60 unclassified, for 88.5% accuracy.

However, for data sets such as the general ledger data where the nodes do not form very clear clusters, spectral clustering does not perform well. In this type of data SNARE has a distinct advantage.

9.6 Summary of Contributions

We successfully applied link analysis to the domain of risk detection for accounting data and produced results that were a significant improvement over a the method that flags suspicious accounts. Formerly, an automated system simply flagged entities that appeared risky, with some sense of priority. Using link analytic methods, one can rerank the risk of an account not only based on irregularities in a single account, but also in other accounts with which it shares transactions. Also, a group of accounts that are closely related and have distributed risk may be identified while under individual flags they would fall below the threshold. In many other domains there may be a cluster of related entities (for example, collaborators in a social network), where the collection of evidence from each party may put the collective risk above the threshold.

We also show that SNARE is successful for the task of node labeling in networks in general. Since risky nodes may be relatively sparse in a graph it may be more useful in anomaly detection to use an initially low probability for risky nodes; however, by adjusting initial belief scores one can use SNARE on tasks where labels are more evenly divided between two classes. SNARE also has the capability of considering prior node-specific domain knowledge for flags— while we used accounting-specific flags in *GeneralLedger1* and *GeneralLedger2*, we chose text flags in *PoliticalBlogs* and committee information in *Campaigns*.

The SNARE system is simple to implement and extend to other domains, and may be particularly useful for other types of fraud detection that ordinary graph clustering methods may have difficulty with, such as link farms or botnets in the web graph, or fraud in mobile phone networks.

In summary, our contributions are the following:

- We have introduced SNARE, which uses belief propagation, taking into account both domain knowledge as well as network effects for labeling nodes in a graph, for risk detection and other applications. SNARE has the following characteristics:
- **Flexible:** We have applied SNARE to a variety of domains, including a sample of general ledger accounting data as well as public datasets (blog labeling, election contributions).
- **Accurate:** SNARE has a high labeling accuracy, compared to simply using flags for accounting irregularity detection (up to 6.5 lift, more than twice that of the default heuristic), and performs better than spectral clustering (with up to 97% accuracy).
- **Scalable:** The algorithm is very efficient, running in linear time with the number of edges in the graph; 50,000 edges completed in 3 seconds.
- **Robust:** SNARE is robust with a variety of parameters, so it requires almost no tweaking of parameters to work correctly. It is therefore flexible, simple to implement, and can be applied to many other domains, in addition to those we have already introduced.

Chapter 10

Star Quality: Analysis of online reviews

PROBLEM STATEMENT: *Given a set of online reviews aggregated from a variety of sources, how do we provide a reliable ranking of the rated objects?*

Given a set of reviews of products or merchants from a wide range of authors and several reviews websites, how can we measure the *true quality* of the product or merchant? How do we remove the bias of individual authors or sources? How do we compare reviews obtained from different websites, where ratings may be on different scales (1-5 stars, A/B/C, etc.)? How do we filter out unreliable reviews to use only the ones with "star quality"? Taking into account these considerations, we analyze data sets from a variety of different reviews sites (the first work, to our knowledge, to do this). These data sets include 8 million product reviews and 1.5 million merchant reviews. We explore statistic- and heuristic- based models for estimating the true quality of a product or merchant, and compare the performance of these estimators on the task of ranking pairs of objects. We also apply the same models to the task of using Netflix ratings data to rank pairs of movies, and discover that the performance of the different models is surprisingly similar on this data set.

10.1 Introduction

The perceived value of reviews on the Web is uncontested: consumer surveys show that people cite product reviews as a top influencer in purchase decisions. According to Nielsen, consumer recommendations are the most credible form of advertising among 78% of survey responders [196]; and a BIGresearch survey indicates that 43.7% of consumer electronics purchases are affected by word of mouth [31]. Additionally, retailers see $15 - 100\%$ greater conversion rates and decreases in product returns for items with reviews [23, 184]. On the other hand, a recent article in the *Wall Street Journal* publicized that the average rating for top review sites is an astoundingly positive 4.3 out of 5 stars [76]. Given the important influence of reviews, we might then ask, how accurate *are* user review ratings on the Web? More particularly, is it possible to extract an aggregate signal from a collection of reviews that accurately reflects the relative quality of the objects under review?

The de facto aggregate review score used by almost all Web properties is the average rating

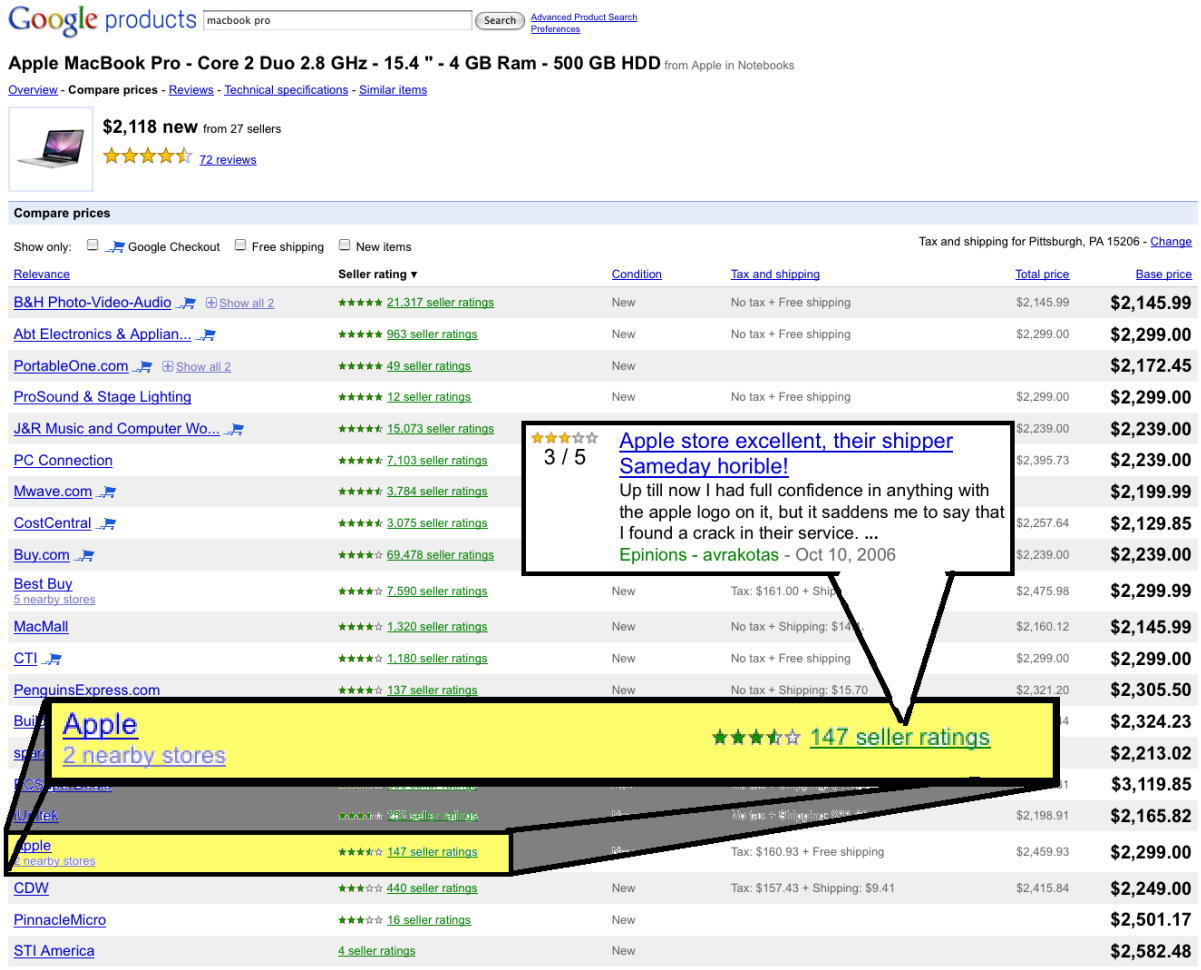


Figure 10.1: The list of merchants for a particular product in Google Product Search, ordered by average rating. Apple, a widely-used seller, appears toward the bottom of the list, weakened by the aggregates. (Note that the default sorting uses a different heuristic.)

per item. However, as shown by Hu et al. [105], this is not always the best way of measuring the true quality of a product or merchant. For instance, we see in Figure 10.1 that for a Macbook computer, Apple is one of the lower-ranking merchants. Since Apple very often ships directly to the user, the lower rank would seem surprising. Does this suggest that average review is a poor reflection of an item's true quality? A few Web properties use "secret sauce" to calculate proprietary composite rating scores or item rankings from a collection of reviews, like alaTest and raveable.com. Do such approaches yield better results?

There are many other issues that arise in aggregating reviews data across sources and authors. Different sources have different rating scales (1-5 stars, 0-10 stars, etc.) or rating distributions (almost all positive, mostly "complaints," etc.) Authors may vary not only in their opinions of products, but in their biases, which may cloud the signal. Furthermore, reviews may be plagiarized, otherwise faked, or irrelevant (reviewing a brand instead of a product, or a product

instead of a merchant) [57].

Our goal in this work is to address some of these issues and compare the performance of average rating against more sophisticated techniques for determining a composite quality score. Specifically, we detail several algorithms for generating composite scores. We then use the scores to rank reviewed objects and compare the performance of the different algorithms against a test set of ranked pairs.

We study three different data sets: product reviews and merchant reviews from Google Product Search, and Netflix movie ratings. The merchant and product review data sets are compiled from hundreds of third party review collections, including Epinions, Amazon and CNET for product reviews and Resellerratings and Shopzilla for merchant reviews. As a result of the issues associated with aggregating a wide range of sources, we initially hypothesized that average rating over aggregated reviews, even with re-scaling, would be a relatively poor predictor for ranking reviewed items. To our surprise, the average proved to be equally accurate as more sophisticated composite scores. While we found a non-surprising result for the particular task at hand, our work lends insight into the problem of finding and evaluating an aggregate ranking.

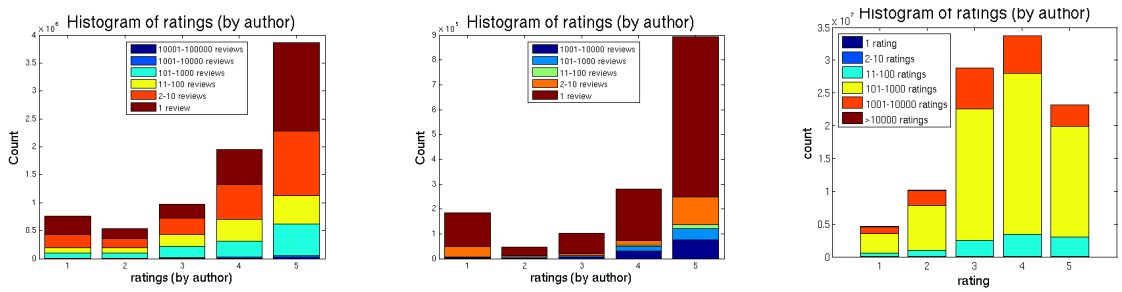
10.2 Related Work

There has been a wide range of work on consumer reviews, from that studying the motivations of consumer behavior in terms of both purchasing and reviewing, to mining product features and predicting scores.

Reviews of products can have a large impact on how well a product sells. Chevalier et al. showed that there exists a *causal* relationship from book reviews to purchasing behavior [53] . Archak et al. further examined the relationship between product features in reviews and sales [12]. However, there are a number of biases reviewers tend to display. Wu and Huberman found that viewing existing reviews caused subsequent reviews written to become increasingly polarized [211]. This may be explained, as Talwar et al. suggested, by users having an "expectation" for a product based on prior reviews; their rating is then impacted based on whether or not the product (in this case, a hotel room) met expectations [197]. On the other hand, Gilbert et al. showed that in many cases reviewers simply echo previous reviews without adding anything new, and report interviews with reviewers on the motivations for doing so [82].

Our problem lies in developing a good relative measure of a product's "true quality" based on user ratings. Hu et al. suggest the average review score becomes an unreliable indicator of a product's "true quality" [105]. However, a better composite measure for a product's true quality is yet to be determined. We therefore explore the possibility that some reviews are more "reliable" than others in terms of their ratings. Duplicate reviews [59] and opinion spam [111] are common, the latter study showing that it is difficult to identify untruthful reviews (plagiarized or deliberately misleading), but two other types can be detected using typical classification techniques: reviews that are irrelevant because they review the brand (not the product), and non-reviews. Authors found spam was more frequent among low-selling products, in reviews which deviate significantly from the average rating for a product, and from people who write multiple negative reviews on one brand.

Many reviews sites also allow users to label reviews as "helpful," which has a disproportion-



(a) Product reviews (b) Merchant reviews (c) Netflix ratings

Figure 10.2: Distribution of ratings in the different data sets, segmented by prolificity of authors.

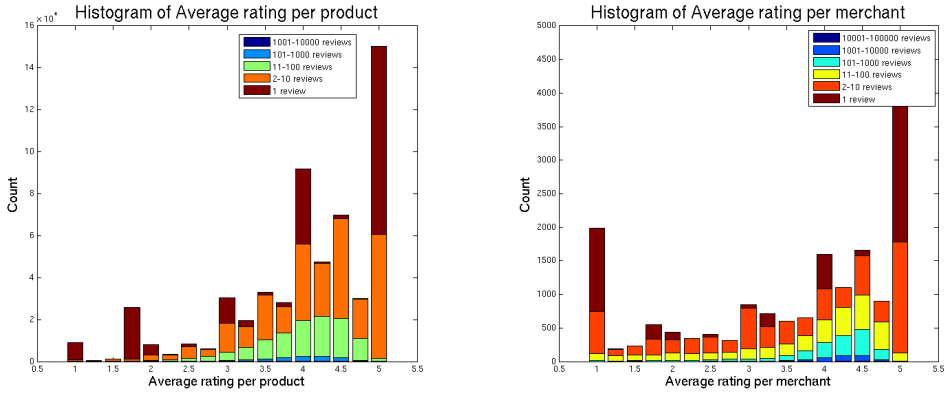
ate impact on purchase decisions [52]. While the problem of finding helpful reviews (finding a very small set of individual reviews that best help a user make a buying decision) is only tangentially related to our problem (deducing true quality of products based on opinions of a large base of users), it lends insight into which reviews to weight the most highly.

Even helpfulness votes, however, have user bias. Otterbacher et al. analyze a set of reviews and note a number of biases and preferential behavior in terms of helpfulness scores [173]. Mizil et al. analyzed a set of reviews from Amazon.com, along with their "helpfulness" scores, as rated by users. Some domains (amazon.us and amazon.de) were more "controversial" in reviews than others (amazon.uk, amazon.jp) [57]. Furthermore, helpfulness scores seemed to depend on the variance of the reviews for a product: for highly controversial products, reviews at the extremes were most helpful.

There have been several studies to automatically assess helpfulness or usefulness of individual reviews. RevRank uses feature selection techniques to construct a "virtual core review" to represent the review space to find a set of the most helpful reviews [200]. Other models that have been used to classify review helpfulness or to identify product features in reviews include [79, 104, 120, 144, 146, 214, 215].

Before proceeding, it is important to distinguish between our problem of finding an overall composite ranking, and the similar problem of finding a ranking for an individual user (personalization). While the problem of aggregating ratings to obtain a composite ranking may seem like it would be an easier problem than that of personalization, in some ways it presents its own challenges. Personalization (e.g. recommender systems, such as that in the Netflix Prize [29]) can rely more heavily on user data and comparisons between users, while an aggregate ranking does its best to find a "one size fits all" ranking. While user information may still be used to weight data, the target user's information is not used.

Given that personalization may be a more well-defined problem, one might ask, why obtain an overall ranking? There are several reasons: First, rich user data is not available for many users. Second, even if the data is available, individual preference may not be relevant under some circumstances. In a way, finding the aggregate ranking lends more insight into the *reviewed object* itself, while personalization also factors in the *individual user*.



(a) Product reviews

(b) Merchant Reviews

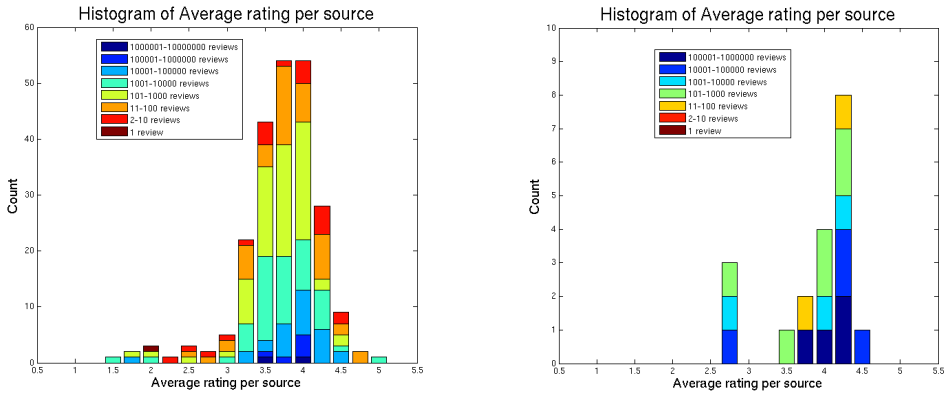
Figure 10.3: Histogram of the average review score for different objects (segmented based on the number of reviews an object receives), for (a) product reviews, and (b) merchant reviews. In both data sets, while highly-reviewed products/merchants have an average of around 4.5, those with very few reviews tend to have average scores of 1 or 5.

10.3 Data Description

We have three data sets we will use: product reviews, merchant reviews, and Netflix movie ratings. Each review has an *author* (an Epinions.com user, for example), an *object* (product, merchant, movie), and a *rating* (1-5 stars). Product and merchant reviews, aggregated from a crawl of several reviews sites, also include the *source*. Product reviews consist of reviews of books, consumer electronics, and other retail items; and merchant reviews consist of users rating experiences with different merchants (similar to rating an Ebay transaction). Both types of reviews are between 1 and 5 stars (normalized according to different scales). These are normalized according to different scales, which as we discussed earlier, has drawbacks. Since our data also contains information on the source, we can use that information to avoid some of the problems caused by normalization. Netflix is a narrower space of reviews: it consists of users rating movies from 1-5 stars. Though this data does not have multiple sources, many of the same methods used to rank products and merchants will still hold for Netflix movies.

There are many subtleties that arise in aggregating data from multiple sources. Each reviews site has its own bent: there may be sites focused largely on complaints (such as those meant to call out scams in merchants), which may translate into a high prior for a review being negative. On a finer scale, the sites may have different foci on each review: some merchant sites may ask for an overall merchant experience while others elicit per-transaction reviews. Some product sites may focus on certain product verticals, such as video games or movies. Furthermore, many review sites are retail sites, and can enforce some power over reviews, whether by following up with users or by removing inaccurate reviews at the request of a merchant. (In these cases, merchants who do not keep a close eye on their reviews may have relatively lower ratings.)

Aside from source biases, the data points themselves may be noisy. Different sites have different rating scales: some are in range 1-10, some 0-5, and some even have only positive/negative labels. Furthermore, some sites simply have a larger number of reviews, which may mean more



(a) Product reviews (b) Merchant Reviews

Figure 10.4: Histogram of the average review score from different sources, for (a) product reviews, and (b) merchant reviews. Notice in (b) that while most sites have an average of a little over 4, there are a few sites with a much lower average of around 2.75.

consistent data, and/or more opportunity for less reliable reviews.

To illustrate some of the inherent biases to confront in aggregating data, we next describe some empirical behavior of the data.

10.3.1 Product reviews

The product reviews data set, gathered from 230 sources, consists of over 8 million ratings, of 560,000 products, reviewed by 3.8 million authors. The distribution of all ratings given is shown in Figure 10.2(a). A overwhelming majority of ratings are positive. Bars are segmented based on the number of reviews an author has given. For instance, a majority of the 5s awarded are by authors who have only given one review.

Examining authors giving a single review, we see that approximately 37% of 5s are given by these authors, but only 25% of 4s are. Likewise, this set of single-review authors appears to give disproportionately more 1's. This makes intuitive sense: one explanation may be that fake reviews are likely to be anonymous to escape blocking (although testing this hypothesis is outside the scope).

However, there does not seem to be a significant correlation between the number of reviews received for a product and its rating: a product with a single review is equally likely to receive a five-star rating as a product with 100 reviews. This is surprising, considering that in Jindal and Liu [111] single-review products were often found to be spam (where spam also tends to have extreme ratings). As shown in Figure 10.3(a), most products have an average review of around 4.25.

Further calculations suggest there is less variance within an author rating different products than within a single product's reviews. This is unsurprising: we would expect a product's reviews to cover a variety of opinions, while an author is likely to have the same "bias" around products they rate. More prolific authors tend to have wider variance. Note that this applies across groups. 1-review authors all have an estimated variance of 0, but the general trend of more prolific authors

having a higher variance applies past that. This could be due to the possibility that the most prolific take time to review both products they liked and disliked, while less prolific ones only did one or the other.

What is more surprising is that within a single *source* there is also little variance. As indicated in Figure 10.4, however, there is a great deal of variance *between* sources, with some having an average as low as 1.5 and others having an average as high as 5! This vastly different rating behavior between sites suggests we should pay attention to the source of a review when determining its quality or bias.

10.3.2 Merchant reviews

The merchant reviews data set is smaller, with 1.5 million ratings for 17,000 merchants. There are 1.1 million authors, from 19 sources.

The distribution of ratings is shown in Figure 10.2(b). The dominating number of 5's is even more prevalent here. As in product ratings, most ratings come from less active authors. As we saw with product reviews data, if an author has written a single review, then the author is disproportionately more likely to have given a 5. Also, as with product reviews, the sources vary widely in terms of rating behavior (see Figure 10.4(b)). For example, ReviewCentre.com has an average rating of about 2.9 while Pricegrabber.com has an average rating of around 4.5.

Looking more closely we see some different effects than what we saw in product reviews. Reviews given to a merchant may have a high variance, more so than for products. While a disproportionate number of merchants with a single review receive either a 1 or a 5 (as in product data), the average review for any given merchant hovers at around 4.5. Authors of merchant reviews had, overall, a higher average *and* a higher variance than authors of product reviews (that is, an author is more likely to use the full scale of 1-5 stars when rating merchants than s/he is when rating products), as suggested by Figure 10.3(b).

10.3.3 Netflix ratings

The Netflix data does not contain multiple sources as the previous two data sets did, but had a larger number of ratings. It consists of around 100 million user ratings (on a scale of 1-5), for 17,770 movies, by 480,189 authors. The data provided for the Netflix Prize was primarily sampled from users with many ratings. As shown in Figure 10.2(c), the mode of ratings is at 4, not 5 as in our other data sets. (However, among the few authors with a single rating, 5 is the mode.) We observe more correlation between the number of reviews for a movie and its average rating: the movies that had more ratings tended to have more positive ratings. For example, movies with over 100,000 ratings had 63% positive (4 or 5), while movies with 101-1,000 ratings had only 42% positive. This effect was not as strongly observed in the other reviews data.

Now that we have provided a brief overview of the data, we will describe our objective in using the data and how we intend to evaluate solutions.

10.4 Problem Statement

Formally, given a set of ratings R where each rating $r(o_i, a_j)$ is a numeric value representing author a_j's opinion of object o_i, our overall goal is to correctly rank each object o_i's "true quality" q_i, relative the other objects. A challenge to ranking is that this "true quality" is unobservable (if such a thing is even definable in such a subjective space), so we will later propose a framework for evaluation.

Each of our models will provide an estimate $\hat{q}_i$ of the quality of each object o_i, and then rank the objects according to that estimated "score." For example, for our baseline method estimates the quality $\hat{q}_i$ as the average rating given by all reviewers, objects with the highest average rating will appear at the top of a ranked list. We will next detail some proposed models for estimating $\hat{q}_i$ in order to perform the task of ranking objects.

10.5 Proposed Models

Our proposed models fall into two main categories: statistical and reweighting. Our baseline model, average rating, falls into the first category. Reweighting models involve filtering reviews out or downgrading their influence in the composite score, considering some reviews to be more important or reliable than others.

We will use the following notation: the estimated quality by a model, for an object o_i, is $\hat{q}_i$. The set r_{i*} represents all ratings *to* a given object o_i, and the set r_{*j} represents all ratings *from* a given author a_j.

10.5.1 Statistical models

Average rating:

This is the model we use as a baseline measure. The estimated quality of an object is the average rating it has received from all authors in the training data. Formally, $\hat{q}_i = \bar{r_i} = \frac{1}{|r_{i*}|} \sum_{j \in r_{i*}} r_{ij}$

Median rating:

This is set up identically to average rating, except for the statistic used. Here the estimated quality of an object is the *median* rating the object has received from all authors in the training data.

Lower bound on normal confidence interval:

Some products have more consistent ratings than others. For example, we would like to give a higher score to a product that has received 100 5-star reviews than to a product that has received a single 5-star review, even though the average rating model would give these the same score. We may also trust a product with a solid string of 4s more than one with a noisier signal. We approximate $r_i \sim N(q_i, \sigma_i^2)$; that is, a rating for a product falls in a distribution around its true quality, with some variance. We then use a *lower confidence bound* for the quality score. More precisely, $\hat{q}_i = \bar{r_i} - z_{\alpha/2} \frac{\sigma_i}{\sqrt{|r_{i*}|}}$, where the constant $z_{\alpha/2} = 1.96$, for a 95% confidence.

Lower bound on binomial confidence interval:

Such a normal approximation may not be accurate. However, we could instead simplify the star ratings into positive/negative (for instance, every rating of 4 stars or above is positive) and then take the lower bound of the confidence interval of the *percentage of positive reviews*. Also known as the Wilson Score, it is calculated in the following manner: First, obtain $\hat{p}$, the proportion of "positive" ratings for a given object o_i. We also define $n = |r_{i*}|$, the number of reviews for an object. Next, the statistic is:

$$\hat{q}_i = \hat{p} + \frac{z^2_{\alpha/2}}{2n} - z_{\alpha/2}\frac{\sqrt{[\hat{p} * (1-\hat{p}) + z^2_{\alpha/2}/4n]/n}}{(1 + z^2_{\alpha/2}/n)}$$

This measure was suggested in [158] as a better way to aggregate ratings for products with few ratings.

Average percentile of order statistic:

One issue with aggregating across several review sites is the different scales used in ratings. For example, while one site uses 1-5 stars, another may use a binary "positive/negative" label, two scales that may not easily translate. How does one maintain a faithful comparison of products, particularly when objects are not reviewed by all sources?

We would like to make a statement such as, "most of the time, object o_1 was ranked above object o_2." We can devise a method for this. We rank products *according to each site*, and calculate a score for an object based on where it occurs on each list. Specifically, we take the *average percentile* for this object over all sites and use that as $\hat{q}_i$.

This score is calculated in a few steps:

1. Aggregate reviews by source, and for each object, calculate the average rating $\bar{r_{ij}}$ from all authors j that are reviewers *from that source* (for example, the average of all ratings received for a product on Epinions).
2. Sort all objects o_i for each source, based on that average.
3. From each sorted list, assign a percentile score for a source-object pair.
4. For each object, take $\hat{q}_i$ to be the average *percentile* it receives for all its sources.

For example, suppose that an object o_i was reviewed on two different sites s_1 and s_2. We would then sort *all products* on each site according to average rating. Suppose that after doing this, o_i was in the 80th percentile on site s_1, and in the 50th percentile on site s_2. The "score" for o_i would then be 0.65.

We can use the same process with authors instead of sources to counteract author bias, but this data is more sparse in the product and merchant reviews where many authors rated few objects. However, it may be useful for the Netflix ratings where authors are comparatively prolific.

10.5.2 Re-weighting models

If we have some idea of which reviews are more reliable, or having more "star quality," we can decide to give more importance to these when training a model. An instance of re-weighting

is *filtering*, which decides that certain reviews are likely to be misleading, and assigns these a weight 0. In essence, we are pruning the training set. We next detail models we used in this class.

Filter out anonymous reviews:

Using regular expressions on the reviewer name ("A Shopper," "Anony*," etc.), remove from the training data any reviews from an apparently anonymous reviewer. Then, calculate the average rating given to an object in the remaining data. Anonymous reviews comprised between 5-10 percent of the training data in products and merchants data. This model is irrelevant for Netflix ratings, where all users were anonymized.

Filter out non-prolific authors:

Sort the data by authors, and remove from the training set any review from an author with fewer than m total reviews (we used $m = 10$ in experiments).

Weighting authors according to maximum likelihood

A more sophisticated re-weighting model involves deriving some "bias" for different authors and re-weighting their ratings accordingly. We propose a model based on an assumption of the distribution of ratings around "true quality." We model $r_{ij} = r(a_j, o_i)$ as a stochastic function of the "true quality" q_i of an object o_i and the "noise" of an author. Some authors are more precise than others, so we say each author has a variance σ_j^2. That is, we make the following assumption:

$$r_{ij} \sim N(q_i, \sigma_j^2)$$

Based on this assumption, the maximum likelihood estimate for the parameters q and σ is:

$$argmax_{\sigma,q} \Pi_{r_{ij}} \frac{1}{\theta_j \sqrt{2\pi}} exp(\frac{(r_{ij} - q_i)^2}{2\theta_j^2})$$

To find each q_i and σ_j, we use the EM algorithm [61] to maximize the value above, iteratively updating each q and σ until convergence. The update equations are as follows: $\hat{q}_i$ is a weighted average of ratings by all authors, where each author is weighted according to the noise among their ratings.

$$\hat{q}_i = \frac{\sum_{r_{i*}} \frac{1}{\sigma_j^2} * r_{ij}}{\sum_{r_{i*}} \frac{1}{\sigma_j^2}}$$

The noise for an author is then simply the sample variance around the quality scores.

$$\hat{\sigma^2}_j = \frac{1}{|r_{*j}| - 1} \sum_{r_{*j}} (q_i - r_{ij})^2$$

One can add more parameters to the assumed distribution of ratings, assigning a bias to an author in addition to a variance term, or assigning variance to a product's quality. However, we found that in practice the more complex models did not outperform the two-parameter one, at least on the Netflix data set.

10.6 Evaluation

10.6.1 Methodology

Since there is no ground truth for the true quality of a product, merchant, or movie, deciding how to evaluate the proposed models is another interesting problem. We cannot exactly answer the question "Can we rank objects correctly, according to quality?"

However, we can answer a related question, which is "Given no knowledge of a user, can we usually replicate their ranking of objects?" To accomplish this, we can first rank objects based on our estimated quality scores $\hat{q}_i$. Then, we can sample from user ratings and see how reliably our ranking matches the preferences of users, in the long run. Essentially, since ground truth data is not available, we are seeing how well an aggregate ranking does on the specific task of personalization. Thus, for the purposes of evaluating different estimates of $\hat{q}_i$ to rank objects, we propose evaluating using a holdout method to obtain training and test sets of reviews. The steps for this are as follows:

1. For each author a_j with greater than n reviews ($n = 100$ in our experiments), pick k pairs of objects rated by the author. Each of these pairs $\{r(o_{k,1}, a_j), r(o_{k,2}, a_j)\}$ will become one data point in the test set.
2. For each pair, label it with whichever object has received a higher rating from the author.[1] *The goal of any model will be to reproduce this ranking for each pair.*
3. Reviews not used in the test set are placed in training set.

Any given model will use the training set to come up with an overall estimated quality, $\hat{q}_i$ for each object o_i and thereby an ordered ranking of objects. Then, for each pair of objects in the test set, the ranking between the pairs is compared to how those two objects are ranked in the model's list[2]. If the model correctly reproduces the relative ranking for the pair, it counts as a hit; otherwise it is a miss. Accuracy is then determined based on the number of pairs it ranks correctly. For example, a random model would arbitrarily choose one of the two objects as the better one for each pair, and receive an accuracy score of about 50%. (It is possible to have conflicting data points in the test set, if the same pair of objects is selected from two authors with differing opinions, however that occurrence is unlikely given the sparse sample in the test set.)

In practice, to build the test sets for each data set, we took one pair of reviews from each author with more than 100 ratings. Each test set was further pruned by the threshold mentioned earlier: if the difference in ratings that the author gave the two products was less than 2 stars, the test data point was not used. This resulted in a test set size of 1423 pairs in product reviews, and 205 pairs in merchant reviews, and 13,141 pairs in Netflix (small test sets due to the selectivity of test points).

Method/Accuracy	Products	Merchants	Netflix
Random[3]	50%	50%	50%
Average rating	70.4%	69.3%	69.0%
Median rating	48.2%	50.2%	40.7%
Lower bound: normal	69.0%	70.2%	68.9%
Lower bound: binomial	65.1%	68.3%	69.1%
Order statistic (by sources)	69.9%	66.3%	N/A
Order statistic (by authors)	62.1%	58.5%	69.1%
Filtering anonymous	67.1%	68.7%	N/A
Filtering non-prolific authors (minimum of 10 reviews)	68.6%	38.6%	69.0%
Reweighting authors by reliability			69.4%

Figure 10.5: Results from running various aggregate quality metrics. Notice that the

10.6.2 Results

We test the various models on the different data sets. Results are summarized in Figure 10.5. Each model was trained on data from all authors (save data points in the test set), and results are computed on the test sets from a select few authors. Accuracy is calculated as the number of *correctly ranked* pairs. Ties and unclassified pairs were considered to be misclassifications. Overall, we found that average rating performed significantly better than random, around 70% in all data sets. Surprisingly, in spite of the different domain, the different measures performed similarly on Netflix data as on the product and merchant reviews.

Some statistic-based models performed up to, but not exceeding average rating. The confidence-interval based models performed promisingly, although a common source for error were objects with a single rating: a variance of 0 made the lower bound of the confidence interval equal to that one rating. (There may be ways to combat this issue, which are worth exploring.) Order statistic-based measures tended to perform as well as average, suggesting that there were few pitfalls in this approach, but it is perhaps more complex than would be useful. Median performed poorly (worse than random) due to a vast number of ties; most ratings were one of the whole numbers.

Performance of re-weighting models suggested that removing data entirely is not always useful. Filtering out anonymous reviews did not have a significant effect either positively or negatively, but filtering out non-prolific authors removed a large amount of data in the merchant reviews, making the model unable to score many pairs in the test set as an object was not rated at all in the filtered training data[4]. We explored the more sophisticated measure of re-weighting

[1]We require that the difference between the pair to be greater than some threshold to use it in the test set. The motivation is that it is more important a model distinguish between a 5-star object and a 4-star object than between a 1-star and a 4-star object.

[2]Since the goal is not personalization, the identification of the author of the pair of reviews in the test set is not used

[4]Using average rating as "backup" in these cases seemed to still not produce an overall improvement over average rating alone (in cases where there was improvement, it was not statistically significant).

"more reliable" authors as described earlier on the Netflix data. Also surprisingly, the results were nearly identical, at 69.1% accuracy.

10.7 Summary of Contributions

We have explored in depth three reviews data sets, including a data set of aggregated product reviews, one of aggregated merchant reviews, and one made up only of movie ratings. Our objective has been to compare different metrics for ranking objects by "true quality," given an aggregated set of ratings for that object. We have tested several statistic-based models and various forms of data-cleaning on this task, and while none thus far have been able to outperform the average rating model (which performs well, but not as well as would be desired), our analysis provides several new observations and promising directions.

Our major contributions are as follows:

- This is the first work, to our knowledge, over aggregated reviews from different sources. We observe that there are often biases of different sources and authors: different authors and review communities will often have very different behavior. We compare reviews coming from these different review sites and investigate how this may help deduce the true quality of an object rated.
- We propose several diverse models for ranking the true quality of reviewed objects.
- We build a framework for evaluating the true quality of reviewed objects and test different approaches.
- We compare performance of different models on multiple datasets, and find surprisingly similar results in terms of performance of different measures.

As we have shown, finding a consistently accurate ranking of objects based on a diverse aggregate set of reviews is not straightforward, but is rather a complex problem with many potential pitfalls in review quality, user and community bias, and in the open-ended nature of reviewing. Learning to properly navigate these challenges will help form a more complete perspective of not only online reviews themselves, but also of the consumer experience and online user behavior.

Part IV

Conclusion and appendices

Chapter 11

Concluding remarks

To review, we address some of the major contributions in this work, and propose other interesting applications.

11.1 Summary of contributions

In this work we addressed patterns and models for network topology and network interactions, and performed case studies of network effect in action. For a list of publications that this work has appeared in, refer to Appendix C.

11.1.1 Topology: New patterns and realistic generators

In Part I, we have examined many different types of large (millions of nodes) weighted graphs, such as social networks of blogs, political campaign contributions, and patent citations. We make several discoveries across these networks, three of which we briefly review here: a) the superlinear behavior of edge weights, b) the constant/oscillating behavior of components, and c) the design of generative models.

- *Fortification:* One surprising finding is that of fortification: the *superlinear* relationship between the number of unique edges in a graph and the total edge weight (such as packet sizes in network transfers or dollar amount in campaign donations). We observe this superlinear relationship on a local scale as well as on a graph at large– for example, candidates with more donors receive superlinearly more money in total. This finding is illustrated in Figure 11.1.
- *Component sizes:* Our second topological observation is that once the giant connected component forms (as is known to occur in social networks), the sizes of the *secondary components* (the second- and third-largest components) oscillate. Surprisingly, there appears to be a certain threshold that a component will reach before joining to the largest component– while this threshold varies between networks it appears to remain near-constant over time within each network.
- *Realistic generative models:* Being able to model these behaviors is important for understanding the mechanisms causing them and allows us to make predictions. As our third

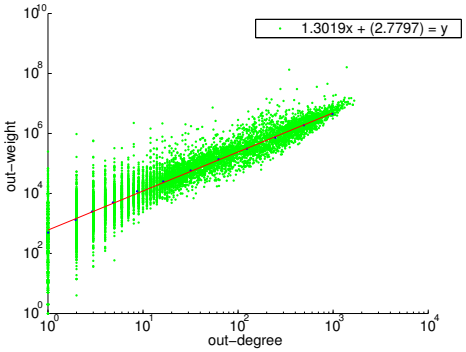


Figure 11.1: A plot of the Snapshot Power Law, detailed in Chapter 3. Here, in the donation network between political action committees and candidates, each point represents one candidate. As a candidate receives more checks, the total amount received increases superlinearly.

topological finding, we have developed two complementary models that match several patterns observed in real evolving networks. The first is an agent-based model which we call the *Butterfly* generator, the first model to reproduce the NLCC property in addition to other known patterns– most agent-based models will produce a single component. The second model is the *Recursive Tensor Model*, which uses tensor multiplication and self-similarity to create a weighted network in time that provably follows the observed power-laws as well as observed bursty behavior. Each of these two models is useful– the agent-based Butterfly model allows us to understand the local mechanisms forming networks and forecast "what if" scenarios, while the tensor-based RTM can be simulated in parallel and is useful for theoretical analysis.

11.1.2 Surprising patterns of interaction

In addition to understanding global properties, we have discovered mechanisms that lead to local diffusion. To this end, we have performed large case studies in blog and online group citations. We completed extensive analysis of a set of 2 million blog posts and identified common patterns of information propagation by analyzing *cascades*, or conversation trees. We were the first to examine the shapes of conversations in blogs, and we have three major discoveries in diffusion that we present: a) the power-law decay of in-links, b) power laws in cascade sizes, and c) that different communities have different cascade patterns. Details of these findings, as well as accompanying generative models, are discussed in Part II.

- *Post popularity decay:* First, we found that in-links to a particular post over time drop off with a power law of exponent -1.5. While one would certainly expect a blog post's timeliness to be important, this finding is surprising as one would expect popularity to decay linearly or exponentially. This is shown in Figure 11.2.
- *Power law cascade sizes:* Our second finding is on cascade sizes. We found that sizes of cascades overall follow a Zipf distribution (power law with exponent -2)– most conversations consisting of a single post with a few being larger. Additionally, power laws apply to

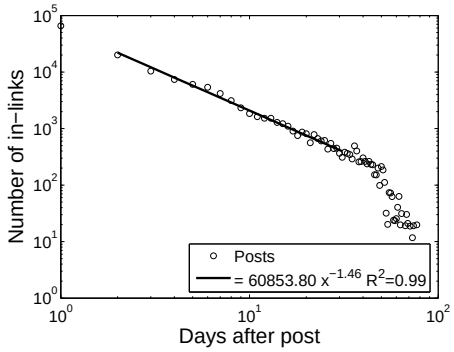


Figure 11.2: A plot of the post popularity decay power law, detailed in Chapter 6.

the sizes of *particular* cascade shapes. Sizes of "stars," or conversations involving many links to a single post, have a power-law dropoff exponent of -3.1, and sizes of "chains" (a series of posts, each linking to the previous) an exponent of -8. Again, timeliness is likely a key factor in forming these types of cascades.

- *Realistic Cascade Generators:* We proposed models to generate realistic cascades in both online groups and in blogs.

11.1.3 Impact

Applications for these findings abound, a few of them discussed in Part III.

- *Network effects for risk detection:* In Chapter 9, we have applied network effects to the domain of accounting. We used link analysis for detecting misstatements (either intentional fraud or unintentional errors) in general ledger data, a problem that causes billions of dollars in losses annually. Our goal was to assess risk of an account based on the series of transactions made, and how closely an entity is associated with other "risky" entities. The scalable method we developed is able to identify misstated accounts automatically, allowing auditors to complete their work more quickly and efficiently. We showed this produced a *lift of up to 6.5* over random, and a vast improvement over using auditing flags without considering network effects, with ROC curves shown in Figure 11.3. It is particularly effective for low false positive rates: for the same false positive rate of 5%, our method achieves a 70% true positive rate while the baseline achieves less than 30%.
- *Anomaly detection for network structure:* We also presented a direct application of topological patterns to detecting anomalies, in Chapter 8. This was able to find anomalous nodes, such as the CEO of Enron, political candidate with anomalous patterns of donations, and a blog post that served as an important connecting resource.

Social networks have increased dramatically in scale over the past few years, and a deep understanding of networks is becoming a necessity to the advancement of online services and security, as well as other problems in the field of data mining [149]. We have presented several surprising patterns in the structure and interactions in networks; proposed generators; and offered several success stories, such as finding anomalous nodes in political campaign contribution

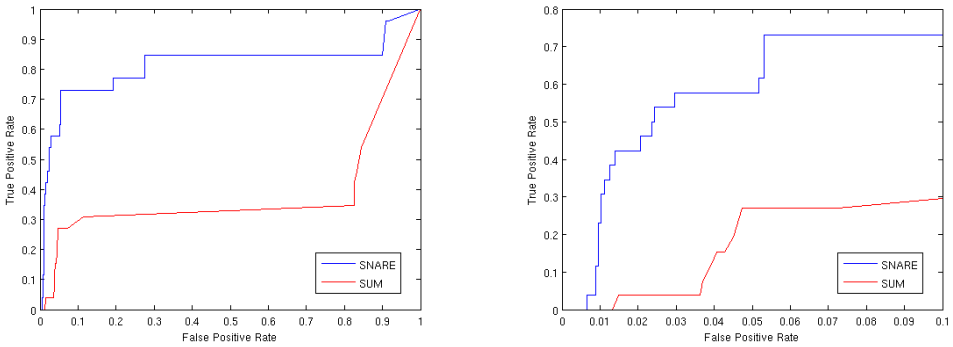


Figure 11.3: ROC curves for SNARE vs. baseline on general ledger accounting data, as detailed in Chapter 9. The first graph shows the entire range and the second shows performance for false positive rates of less than 0.1.

networks (a domain with significant impact) and detecting misstatements in accounting networks (another problem with multi-billion dollar consequences). We believe that these are only a few of the opportunities that our findings present.

Appendix A

Case study in online groups: Inter-group patterns and cross-posting

PROBLEM STATEMENT: *Can we develop a deeper understanding of cross-posting behavior in groups? When a cascade (in this case, a thread on a group's message board) appears in several different groups, how can we tell which groups find it most interesting?*

Social networks, both on- and off-line, are rich structures of communities and communities-within-communities. An individual may be a member of multiple social circles. While this property enhances the flow of communication across networks, it makes community identification difficult in most on-line social network data. Unlike many Web 2.0 communities, Usenet has a pre-defined structure for topics of discussion, which allows us to identify which individuals are most responsible for bridging communities and aiding in information diffusion not only within, but also *between* communities. In this work we examine the structure of communities and diffusion patterns of nearly 200 politically-oriented newsgroups, both the interactions inside newsgroups and, in particular, at the borders of them, where membership, interests, and topics of discussion overlap.

Studying the pre-defined Usenet groups allows one to bypass the obstacle of community detection. This advantage, however, presents a host of interesting challenges, as the borders do tend to blur. *Cross-posting*, where a single article is posted into several groups simultaneously, is frequent in Usenet. While studying cross-posts can aid us in finding gateways for information transfer, an improper cross-post leads to confusion of relevance. Since users can simultaneously (and nearly without cost) "spam" multiple groups, and often times respondents to an article will "reply-to-all," so that an entire conversation can appear to happen in a group when none of its regular readers are taking part. To combat this, we propose a framework for assessing *ownership* of an article, or post.

Our work is one of the first principled approaches towards analyzing diffusion patterns in Usenet. Our contributions are the following: We perform a study of a large set of Usenet newsgroups over an extended period of time, comparing the structure of the induced social networks. We find that induced networks of groups obey a form of the densification power law, with slope of 1.2. However, despite this structural similarity, we find that reciprocity and degree distribution varies in the different groups. Understanding these structures helps us properly assess similarities

in newsgroups based on membership and cross-posting activity. We then present a framework for assessing which of many cross-posted newsgroups is responsible for most of the activity in a thread, and which ones are responsible for influencing other groups. Using this framework, we show how cross-posting later in a conversation induces higher activity, which illustrates the flow of information between communities, and observe some precise diffusion patterns between Usenet groups.

A.1 Comparing structure in newsgroups

First we examine structural properties within each newsgroup. Here, we make an induced social network $G = (N, E)$ of authors based on replies to posts. In each group, if author a_n replies to a post by author a_m, there is a directed edge e_{mn} from a_n to a_m.

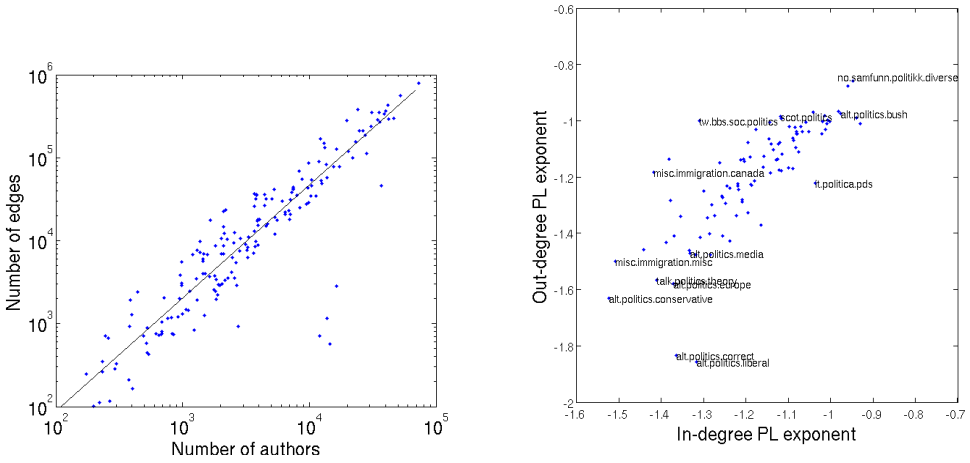
A.1.1 Size

The size that a group reaches is one key feature examined. Interestingly, groups seem to mimic the *densification power law* discovered by Leskovec et. al: as a graph size grows in nodes, the number of edges increases super-linearly [136]. However, while the densification law is traditionally applied to several snapshots of the same graph at different points in time, here we observe several different groups at the same point in time. The plot of edges vs. nodes is shown in Fig. A.1(a). The weighted graph, where the weight on an edge is the total number of replies, also follows densification with exponent of 1.3 (plot omitted for space). There are some notable, interesting anomalies. The points far below the fitting line (with abnormally low reply rates) are `tw` domains. The ones above the fitting line (high reply rates) tend to be in European domains.

A.1.2 Degree and reciprocity

We have shown that groups tend to maintain a certain edge to node property, but how are these edges distributed? The degree distribution indicates how skewed interactions are: a steeper slope on a power-law fit implies a higher proportion of activity by the "core" authors. Fig A.1(b) shows the in-degree vs. out-degree power law exponents for groups that did fit such a distribution, based on log-binning of histogram data. Among groups that had a fit value of $R^2 > .95$, the power-law exponent ranged from -0.95 (`no.samfunn.politikk.diverse`) to -1.5 (`alt.politics.conservative` for in-degree. Out-degree power law exponent ranged from -0.86 (`no.samfunn.politikk.diverse`) to -1.8 (`alt.politics.liberal`). While correlated, there was a wide range of exponents, and some did not even appear to be heavy-tailed, which was surprising.

Reciprocity between groups represents whether most users reply to each other. The formula for reciprocity may be found in [36], but it is essentially a ratio of the number of pairs of nodes that have a mutual edge to the number of pairs of nodes that have a non-mutual edge (one that goes only one direction). A group with no reciprocated edges would have reciprocity 0, and a group where all edges are reciprocated would have a reciprocity of 1. The most reciprocated group (`hun.politika`) had a reciprocity of up to 0.58, and the least reciprocated



(a) (b)

Figure A.1: Comparing Usenet groups. (a) Number of author-to-author edges (interaction pairs) in groups vs. number of nodes (authors) in groups, based on replies. The power-law exponent is 1.2. (b) In-degree power law exponent vs. out-degree power law exponent, for groups with an R^2 fit of greater than 0.95. Some outliers are labeled. There is a general correlation of in-and out-degree, but there is a great deal of range in the steepness of slopes in the degree distribution.

group `tw.bbs.soc.politics`, had a reciprocity of 0.057. Interestingly, with the exception of `hsv.politics` (Huntsville, Alabama), all of the top 20 high-reciprocity groups were European, and most of these highly-reciprocity groups did not fit a power-law degree distribution at all. The low-reciprocity groups generally had low traffic (fewer than 100 authors in any given year, with the exception of `tw.bbs.soc. politics`). All of Taiwan-based groups in our data had very low reciprocity.

A.2 Similarity Measures Between Newsgroups

We have now compared the individual groups and showed some of their differences. But how can we draw similarities between the groups? Cross-posting may help provide us with information on how related different groups are, by making the assumption that if authors regularly post the same articles into multiple groups, then the groups share those related articles and are likely of similar motivation. Likewise, groups with shared authors may be related.

We first measured how often cross posts occurred. For this, we use the Jaccard coefficient: the ratio of intersecting articles to the union of articles in both groups.

$$Sim(g_1, g_2) = \frac{|Articles(g_1) \bigcap Articles(g_2)|}{Articles(g_1) \bigcup Articles(g_2)}$$

Fig. A.2 is a visualization of the resulting network [1], where an edge represents similarity

[1] All network visualizations in this work, including illustrations of threads later, use Eytan Adar's GUESS Graph Exploration tool, [2].

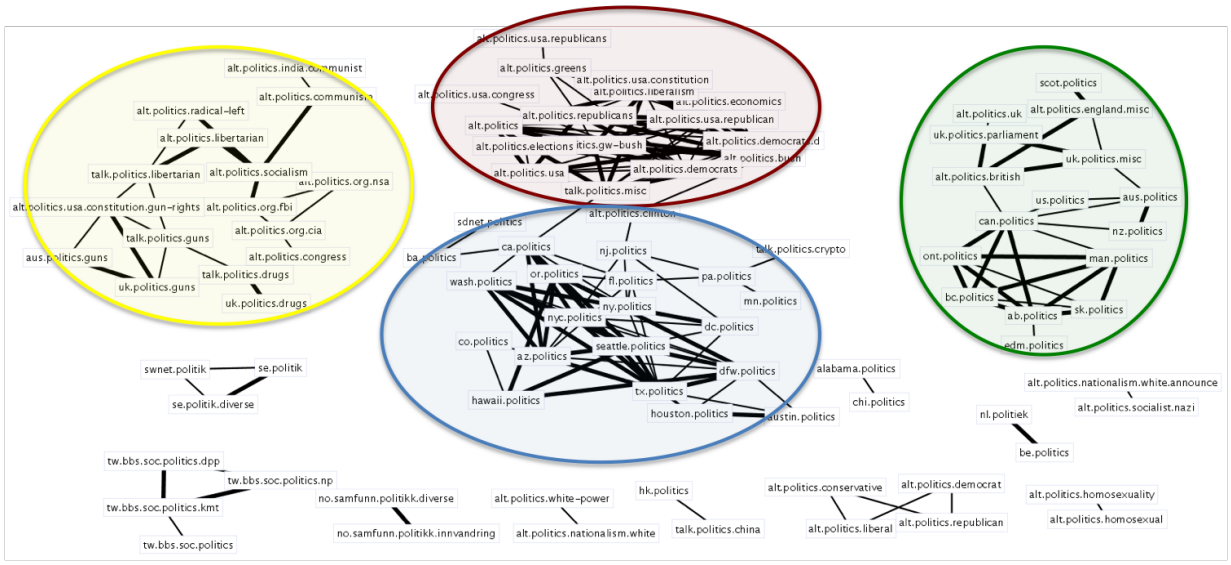


Figure A.2: Newsgroups clustered by cross-posting based on Jaccard coefficient. A thin edge indicates a similarity of over 0.1, and a thick edge of over 0.2. In the center there are distinct clusters for local U.S. politics groups and the main `alt.politics` groups. On the left are topical groups for issues and some political philosophies, and on the right are clusters for local Canadian groups and for other English-speaking countries. Otherwise, groups sharing language or physical borders tend to group together.

greater than 0.10, and a thick edge similarity greater than 0.20. There are some interesting groups forming: the large cluster on the right includes most of the Canada local groups joined with thick edges. Notably, the group `qc.politique` was missing. We found that it actually had a higher similarity with `fr.politique` than with any of the other Canadian groups, likely due to language. Also joined to the Canada cluster (green) are other general politics groups for English speaking countries, such as the U.K., Australia, and New Zealand. In the center there is a cluster largely devoted to the U.S., with most of the regional and statewide groups on the bottom (blue). There is a surprising rate of cross-posts in this area; however, some of the less-well-connected regional groups tend to be connected in an intuitive manner: for instance, `sdnet.politics` (San Diego, Cali.) and `ba.politics` (Bay-Area, Cali.) are connected, and `houston.politics`, `dfw.politics`, and `austin.politics`, three groups for major cities in the state of Texas, along with `tx.politics`, form a clique. Above the local-U.S. cluster (in red) is a cluster of most of the `alt.politics.*` hierarchy; cross-posting is very high among these groups. To the left is a fourth cluster (yellow), mainly centered around topical groups such as guns, drugs, or specific political philosophies, with fairly intuitive connectedness. Otherwise, groups joined by language or physical borders tend to cluster together. Groups focused on Sweden, Taiwan, Norway, Hong Kong/China, and Netherlands/Belgium are related. About half of the groups are not shown, as they had no edges above the threshold.

We also measured similarity based on Jaccard coefficient of the author participation in each newsgroup, where similarity is the ratio of the size of the intersection of authors in each group to the union of authors (in the same manner we assessed cross-posts). Here we thresholded

edges at an coefficient of 0.2, thick edges at 0.3, which resulted in about half of the groups being connected to at least one other group. The visualization is omitted for space; however, we found that the structure formed similar clusters to those in Fig. A.2.

Next we will study patterns of diffusion, exploring whether similarity leads to more information flow. We do this by providing a method of assigning cross-posts to groups.

A.3 Proposed thread ownership method

In the previous section we completed a multi-scale analysis of the Usenet sample, both contrasting differences between the groups and clustering them based on similarity measures. We next analyze threads themselves, particularly focusing on how threads move between groups. Often times even when a thread is initially posted to one or a few groups, it may be later cross-posted to others. The thread may be picked up by the new groups, but even if members in the old groups are no longer interested in the discussion (or never were), people in other groups may still cross-post to that group (the "reply-to-all" effect). Therefore, as we describe the interactions we try to consider when we can truly consider a discussion as occurring in a given group. To that end, we propose a measure of *ownership* for authors and for articles, and show how it aids in studying diffusion patterns.

A.3.1 Post ownership

Since nearly half of all posts are cross-posted, it is difficult to assign ownership from articles alone. However, based on the *authors'* posting patterns, we can often discern where their loyalties lie, so to speak. If an author usually posts into g_1 and only occasionally cross-posts into both g_1 and g_2, then it is a safe assumption that posts written by that author "belong" to g_1. To aid in formalization, we define the following expressions:

Author-group activity, $act(a, g)$ is defined as the percent of author a's posts that are posted into group g. These may be cross-posted, so $\sum_g act(a, g) \geq 1$.

While this may give a realistic distribution of where an author is cross-posting, we feel that in order to capture whether an author truly considers himself a member of a group, we need to determine where that author is writing unique posts, because many cross-posts are unintentional "reply-to-alls." Therefore, we define *Author-group devotion*, $dev(a, g)$, as the percent of author a's posts that are *only* posted into group g, and not cross-posted into any other groups. Therefore, $0 \leq \sum_g dev(a, g) \leq 1$. From there, we can define a group g_i's degree of ownership of a post, based on how devoted the post's author is to the groups it is posted into.

$$own(g_i, p) = \frac{dev(a(p), g_i)}{\sum_{g_j | p \in g_j} dev(a(p), g_j)}$$

A simple extension gives us the ownership of a *set* $\mathcal{P}$ of posts, taking the mean of the ownership of each post. One can apply this ownership score to the set of all posts that have occurred, whether uniquely or as a cross-post, into a group[2]. In this manner we have aggregated

[2]For some posts $dev(a(p), g)$ is 0 for all groups in question. This is a relatively rare occurrence, particularly on

ownership for posts and devotedness scores for authors. We find that some groups "own" a large amount of their posts, while others have much sparser relative ownership. For instance, `fr.soc.politique` has a ratio of 0.92 while `alt.politics.bush` has an aggregate ownership score of 0.56: so under this score, `alt.politics.bush` actually has less activity. Some groups had even lower ratios of ownership; for example, `tw.bbs.soc.politics.kmt`'s was around 0.003.

We illustrate the importance of ownership using an example. In Fig. A.3, we show a conversation cross-posted to several groups, and then label each node with the group that the author most "belongs" to (based on highest ownership). The original article, "Kiss the national parks goodbye," was cross-posted to several large newsgroups, including `talk.politics.misc` and `alt.politics`. The second node from the left on the second level was a reply to that post, which was cross-posted to `talk.politics.misc`, `seattle. politics`, `or.politics`, and a few other local politics groups. According to our ownership rules, the bulk of the thread was made by authors that mainly posted to `seattle.politics` (16,000 members, marked in green) and `or.politics` (10,000 members, blue). Authors posting primarily onto talk.politics.misc (a much larger group, with over 50,000 participants) are marked in red. Even though nearly all of the posts were cross-posted to `talk.politics.misc`, few of the "devoted authors" of that group participated. Considering the subject line, it is not surprising that such a subject would appeal more to members of groups in the Pacific Northwest, which has a higher concentration of national parks.

The largest thread was over 9000 posts, occurring in major `alt.politics` subgroups and `talk.politics.misc`, and focused on the 2004 election. It was cross-posted to 38 groups during its tenure, yet, 85% of ownership was concentrated in three groups.

A.3.2 The effects of cross-posting on threads in groups

Once we have established which groups dominate conversation for a given thread, we can develop a better understanding of how cross-posting affects how well-received a thread becomes inside a group. We can start to answer the questions: How does cross-posting affect a conversation? Does a conversation pick up when cross posted, or die off? How does a thread fare if it begins in a group, compared to when it begins elsewhere? To assess whether cross-posting helps or hurts activity in groups, we can divide conversations happening in a group g_i into the following four categories:

1. An article is initially posted to g_i and never cross-posted to other groups in our data set. (No X-post)
2. An article is initially cross-posted both to g_i and another group in the data set. (Initial X-post)
3. An article is initially posted to g_i and, later in the conversation, a reply is cross-posted to a different group. (Late X-post, original group)
4. An article is initially posted to another group, and later in the conversation debuts in g_i. (Late X-post, late group)

the thread level.

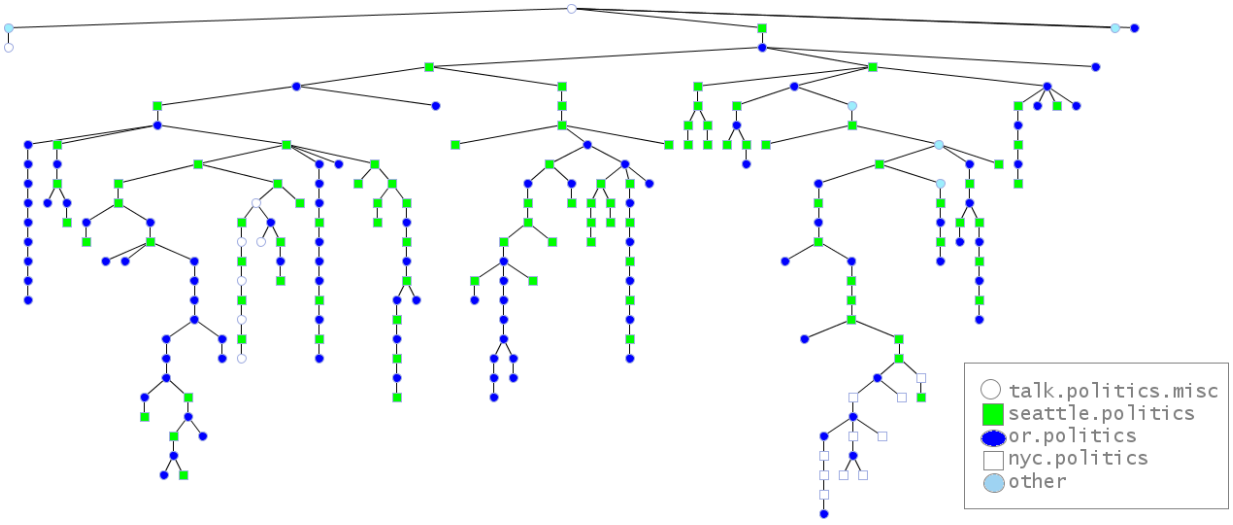


Figure A.3: An example of a thread that is posted into several groups but is "owned" by a very small number. It is described in detail in the text. While the original article was cross-posted to several large newsgroups, including `talk.politics.misc` and `alt.politics`, most of the posts are from authors who primarily make their non-cross-posts into `or.politics` and `seattle.politics`.

To compare these cases, we took the ownership of the set of posts in the thread. (In the fourth case this means taking the ownership of all posts below the point in the conversation where g_i appears). In Fig. A.4, we show the distribution of thread sizes, for the different "types." All types follow a heavy-tailed distribution. However, it is clear that most of the largest threads are of the "late-cross-posting" type. Furthermore, there is not much difference in overall thread size for threads with no cross-posts and those that are only initially cross-posted to multiple groups, so simply the act of cross-posting may often be associated with spam.

We recognize that there is some correlation between natural thread size and type (by definition, threads of type 3 and 4 must be at least of size 2, for instance). We can make a better assessment by instead examining what happens not simply to the thread overall, but what happens *within each group*. If we measure the cascade size based on ownership for a given group, we can more confidently state whether the act of cross-posting induces conversation. In doing this, we find that Type 4 threads do indeed have more activity. We are only measuring the size *below* the point where it reaches the group, making it a comparable measure to types 1 and 2. The resultant PDF is shown in Fig. A.4, normalized as there are relatively few Type 4 occurrences.

In other words, mass initial cross-posting does not lead to high activity within any given group. However, if somewhere in a thread an author decides it is relevant to group g_i and cross-posts, then g_i tends to gain more activity than it would for a post that was not cross-posted at all. Perhaps this is indicative of authors "discovering" threads that are relevant to a given group, and "recommending" these threads to the group by cross-posting their replies. Indeed, we find that for cases where a post is later cross-posted to a new group, about half the time the person who introduces the post is "devoted" to both the old group and the new group.

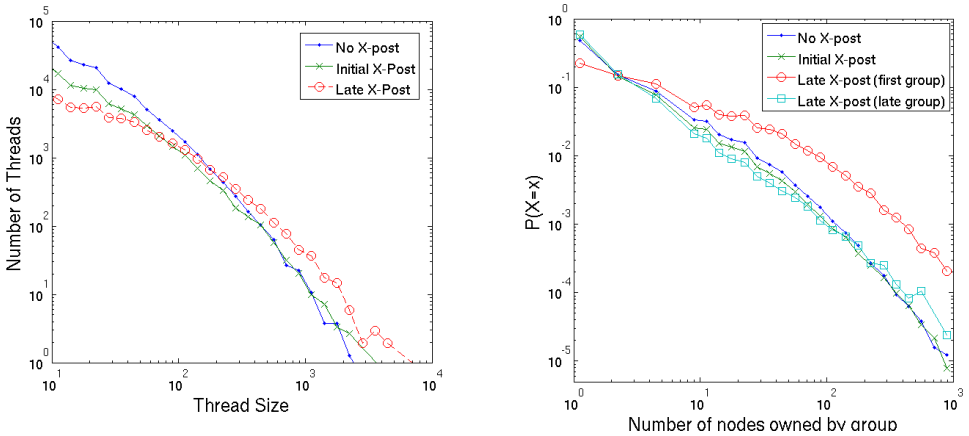


Figure A.4: **Top:** Histogram of thread sizes, where each thread is either never cross-posted, cross-posted only at the root, or cross-posted later. Most of the largest threads tend to have late-occurring cross-posts. **Bottom:** PDF distribution for per-group thread ownership. Here, threads are double-counted for each group they appear in. however, posts are divided amongst the groups such that each *post* is only counted once. For the first two types, a higher proportion of the probability mass is concentrated in less activity, while late cross-posting leads to higher activity in the new groups.

One example of this phenomenon occurs in a thread with subject line "The truth about British Racism & Imperialism." It begins by being cross-posted to `alt.politics.british` and `uk.politics.misc`. At one point in the conversation, one author replies saying "If you can be Scottish and British, why not Asian and Scottish?" A second author, who we have labeled as most "devoted" to `scot.politics`, then posts "Why not be Asian and Scottish? Most Asian people in Scotland consider themselves to be both." In the process of replying the author also sends the reply to `scot.politics`. At that point, there is an explosion of conversation; in fact, we find that 79 percent of the conversation occurs below this point, and largely among authors in `scot.politics`. We show a diagram of the conversation in Fig. A.3.2, emphasizing the point at which the late cross-posting occurs. Taking into account this mechanism of "discovery," we next assess diffusion in terms of thread ownership.

A.4 Applications of post ownership

Next, we propose applications of our method, including a way of measuring diffusion and a way of measuring group similarity.

A.4.1 Information flow based on post ownership

Without an idea of where posts are truly occurring, measuring how information flows across groups becomes difficult to assess. If a parent post p_p is cross-posted to g_1, g_2, g_3, and an author then replies to it by adding a child post p_c into g_4, how does one assess where the new author

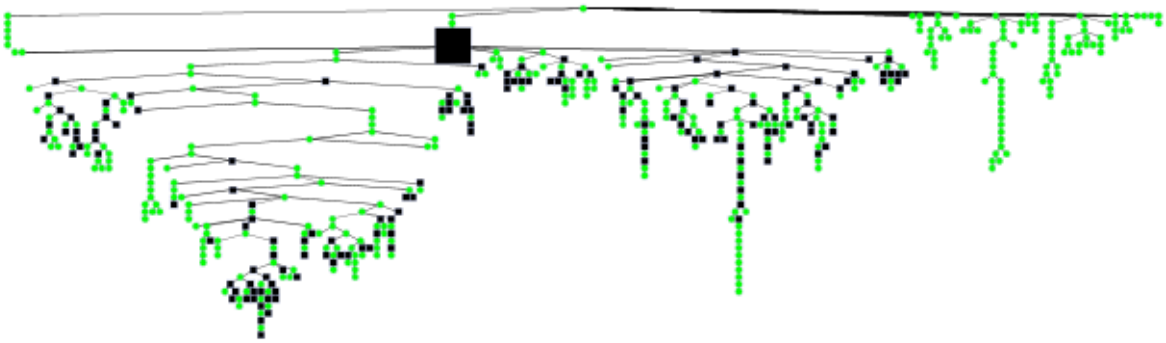


Figure A.5: An example of a thread that is first posted to `alt.politics.british` and `uk.politics.misc`, but later is cross-posted into `scot.politics`. At the point which the third group is added (denoted by a large black square node), the conversation takes off, and 79 percent of all nodes occur below that point. `scot.politics`-owned posts are marked in black.

read the original post; that is, which group influenced her to form edge e_{pc}?

The goal is to find an *influence measure* for any two groups, based on a given edge, which we can extend to the entire set of threads. We would like a score $Infl_{e_{pc}}(g_p, g_c)$ for each possible pair of groups. Without ownership information, one might assign the influence as a distribution from all of p_p's groups and all of p_c's groups. For each pair,

$$SimpleInfl_{e_{pc}}(g_p, g_c) = \frac{1}{|(g_k|p_p \in g_k)|} * \frac{1}{|(g_l|p_c \in g_l)|}$$

Under this case, since there are three groups in the parent post, and one in the child post, $SimpleInfl_{e_{pc}}(g_1, g_4) = \frac{1}{3}$. To get an influence score between two groups over an entire group of threads, one would simply sum the influence scores for each pair of parent-child posts. However, this measure has shortcomings: it ignores the fact that some cross-posting may be meaningless to authors who post only to a certain group. Therefore, we introduce ownership. We may decide to assign influence based on how devoted the parent post's author, $a(p_p)$, and the child post's author, $a(p_c)$, are to each group. The score for any pair of groups $(g_p, g_c|p_p \in g_p, p_c \in g_c)$ is then:

$$OwnInfl_{e_{pc}}(g_p, g_c) = dev(a(, g_i) * dev(a, g_j)$$

Still, we would like to take it a step further, to answer the question, *How often do authors in g_c respond to a post they first saw in g_p?*. One would then measure not g_p's influence based on the parent distribution, but rather the child author's distribution:

$$ChildOwnInfl_{e_{pc}}(g_i, g_j) = dev(a, g_i) * dev(a, g_j)$$

These three potential measures allow us to attribute influence over the entire set of threads. Summing over each e_{pc} where an edge is a reply, and normalized based on the "influencees" we can

get a total score of influence from each group to another. Under $SimpleInfl$, we find that a slim majority of the mass (57%) is along the diagonal of the adjacency matrix. By using $OwnInfl$, attributing the flow from an ownership distribution of the parent post, into an ownership distribution of the child's post, 67% of the mass is along the diagonal. Taking it a step further, by attributing influence based only on the newer author, under $ChildOwnInfl$, 85%. This would seem the most intuitive measure of influence, as one would expect most influence to occur within a group.

Based on the third measure we can claim that perhaps 15% of the time, information is traveling from one newsgroup to another. Which groups are responsible? Based on $ChildOwnInfl$, we found that the most influential were often the ones with the largest mass, such as `alt.politics.bush` and `alt.politics`, but were more often simply the larger groups in a cluster, such as `can.politics` in the Canadian groups, `seattle.politics` in the local US groups, or `talk.politics.guns` for topical groups. The following edges had the highest influence scores:

Influencer	**Influencee**
`swnet.politik`	`se.politik.diverse`
`de.soc.politik.misc`	`bln.politik.rassismus`
`can.politics`	`man.politics`
`can.politics`	`ab.politics`
`can.politics`	`bc.politics`
`can.politics`	`ont.politics`
`uk.politics.misc`	`uk.politics.constitution`
`uk.politics.misc`	`uk.politics.parliament`
`talk.politics.drugs`	`uk.politics.drugs`

A.4.2 Group similarity based on shared "devoted" authors and shared posts

This new framework of ownership brings previous measures of group similarity into a new light. We can re-assess group similarity based on "devoted" authors. By redefining group membership from "any member who posts into a group" into "any member who, at some point, single-posts into a group," and then taking the Jaccard coefficient, we paint a different picture of which groups truly share members. Naturally, the similarity scores are lower. One can also build a network using similarity of shared ownership of posts: a post is shared between two groups if $dev(a(p), g_i) > 0$ for both groups. While the general structure is similar, there are a few interesting differences. For example, the devoted-author network has a much more distinct divide in the local U.S. groups; with a couple of exceptions, the groups appear to be neatly divided between cities/states on either side of the Mississippi River (see Fig. A.6).

A.5 Contributions

Our contributions in this case study are the following:

- We compare structures of different politically-oriented Usenet groups. We find superlinear behavior between the number of authors and number of edges (similar to "densification" discussed in Section 2.1, only using snapshots of several different groups).

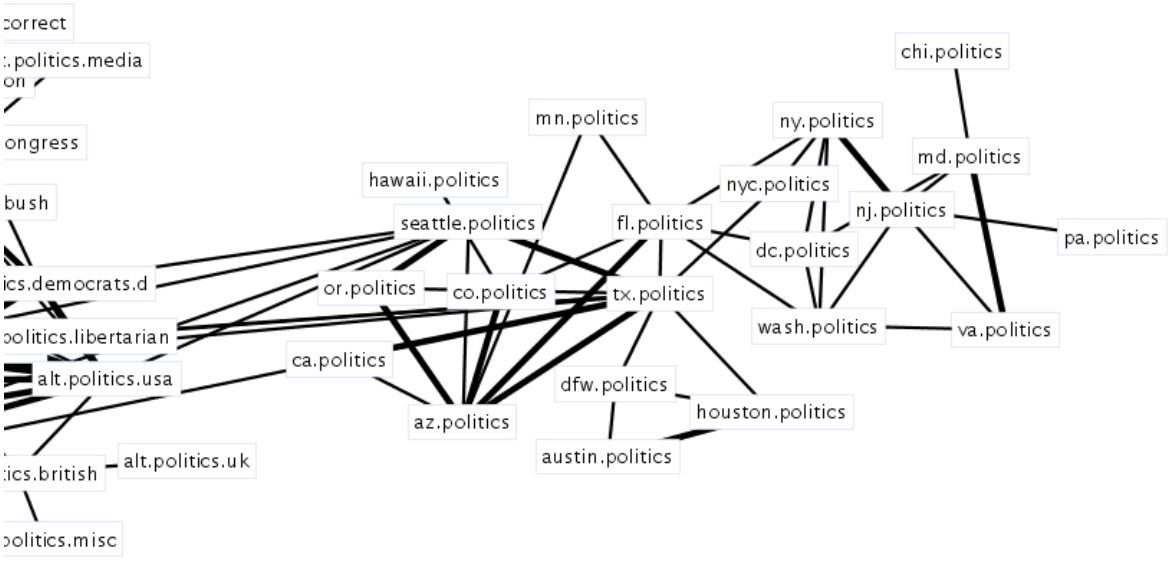


Figure A.6: Similarity based on devoted authors, focusing on the local US groups. A thin edge represents a Jaccard coefficient of ≥ 0.08, and a thick edge ≥ 0.1.

- We show that degree distribution and reciprocity vary widely across groups, even though each group may discuss many of the same topics and have overlapping membership.
- We are among the first to deeply study cross posting behavior in online groups. We show that cross-posting may lend insight into which groups are most similar (without requiring heavy text analysis!).
- We propose a *post ownership* method to help demystify which groups find a cross-post most interesting.
- We use our post ownership model to infer diffusion between groups, and to improve upon our similarity measures.

While cross-posting aids in analyzing similarity between groups, when it comes to assessing relevance within groups, cross-posting becomes a barrier to understanding. Therefore, we have proposed an ownership measure, which assigns posts in a thread to groups based on how "devoted" the post authors are to the various groups. Our ownership measure is an excellent tool for many applications in data analysis. By assigning ownership of posts to groups, we observed how threads evolved as cross-posts occurred. By looking at different "types" of cross-posting activity, we demonstrated that while cross-posting, when initially in a thread, does not lead to more activity, a cross-post that occurs later in the thread is correlated with higher activity. Furthermore, we were able to create an influence measure between groups, based on the ownership of parent and child threads. These experiments in cross-posting activity that examine the devoted authors and activity in groups are particularly relevant, as identifying individuals who are devoted to multiple groups serves to better understand how information is transferred across social group boundaries.

Appendix B

Case study in blogs: Labeling blogs using cascade features

PROBLEM STATEMENT: *Can we use cascade features to cluster blogs by genre, or to characterize a certain blog?*

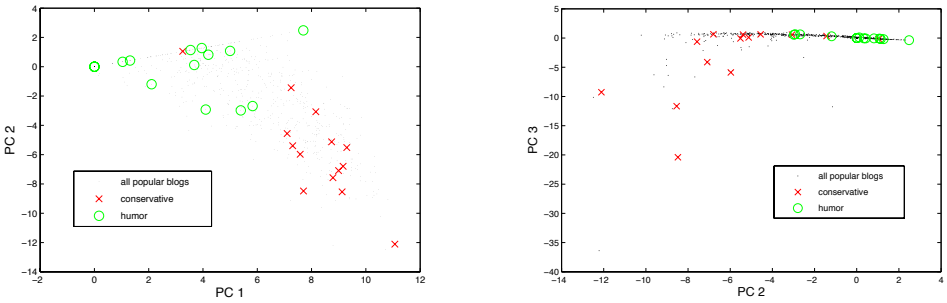
We use a method known as *principal component analysis* to cluster blogs. PCA is defined as follows. Given many vectors in D-dimensional space, how can visualize them, when the dimensionality D is high? This is exactly where Principal Component Analysis (PCA) helps. PCA will find the optimal 2-dimensional plane to project the data points, maintaining the pairwise distances as best as possible. PCA is even more powerful than that: it can give us a sorted list of directions ("principal components") on which we can project. See [113] or [126] for more details.

B.1 Clustering blogs by CASCADETYPE

Our first experiments involved performing PCA on a large, sparse matrix where rows represented blogs and columns represented different types, or shapes, of cascades (shapes, such as those shown in Figure 6.4 on page 71). Each entry was a count, and in order to reduce the variance, we took the log of each count. Our dataset consisted of $44,791$ blogs with $8,965$ cascade types.

It was of interest to impose social networks upon the blogs, based on what topics the blogs tended to focus on. We hand-classified a sample of the blogs in the ICWSM data by topic, and found that we could often separate communities based on this analysis. For the purposes of visualization we chose to focus on two of the larger communities, politically conservative blogs and "humorous" blogs (such as blogs for different web-comics and humorists). Figure B.1(a) shows these blogs plotted on the first two principal components, and Figure B.1(b) shows them plotted on the second and third principal components. Ovals are drawn around the main clusters. We notice a distinct separation between the conservative community and the humor community; this means that the two communities engage in very different conversation patterns.

Based on our CASCADETYPE analysis, we make the following observations:



(a) First vs. second PC

(b) Second vs. third PC

Figure B.1: Principal components for blogs by CASCADETYPE labeled by topic. PC's were generated by analyzing a matrix of blogs by counts of cascade types. Note that there is a clear separation between conservative blogs *(*represented by red crosses*)*, and humorous blogs *(*represented with by circles*)*, both on axes of the first and second PC *(*a*)*, and on axes of the second and third PC *(*b*)*. Ovals indicate the main clusters

Observation B.1.1 *Communities often cluster around the same types of cascades, with distinct conversation patterns.*

It seems that conservative blogs and the "humorous" blogs form separate clusters. We believe this is the case because conservative blogs tend to form deep, chainlike graphs whereas the humorous blogs form stars. Some similar observations may be made for other communities; we used these two because they were the most distinct. This result shows that blog communities tend to follow different linking patterns. We believe that by looking at a blog's cascade types that one can better make inferences about what community a blog might belong to.

Observation B.1.2 *The number of trivial cascades that a blog participates in (that is, its number of solitary posts with no in- or out-links) may be a key indicator of its community.*

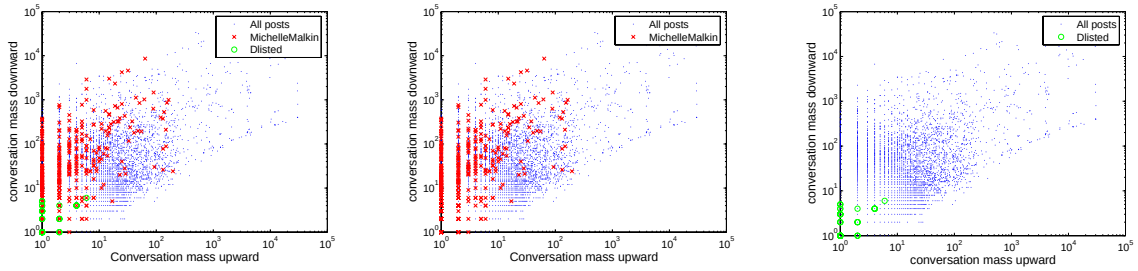
Removing the trivial cascades caused the clusters to become less clear, which indicates that these trivial cascades still play a significant role in the inferences one can make about that blog.

B.2 Clustering based on post features

We next sought to find how posts themselves behave. In order to do this, we performed PCA on a 6-column matrix. Each row represented a post, while the columns were as follows:

- Number of inlinks
- Number of outlinks
- Conversation mass upwards
- Conversation mass downwards
- Depth upwards
- Depth downwards

There were $6,666,188$ posts in the dataset. When we ran PCA, we found that the major two components that determined the blog's place in this space were conversation mass upwards and downwards. Therefore, we also plotted the posts on the two axes of conversation mass upwards



(a) All posts

(b) MichelleMalkin (superimposed on all)

(c) Dlisted (on all)

Figure B.2: Conversation mass for posts, an aspect of POSTFEATURES6. The top figure shows the Dlisted and MichelleMalkin clusters superimposed over points for all posts. The next two show the clusters separately, superimposed on all blog points for reference. Ovals are drawn around the main clusters. Note that while there is overlap between posts features of two blogs, they have different centers. This tells us that different blogs maintain different means and variances in conversation masses

and conversation mass downwards (See Figure B.2. To illustrate, we have plotted all posts, with special markers for two distinct popular blogs, Dlisted [1] and MichelleMalkin [2]. We have circled the main clusters in the plots. Notice that while Dlisted and MichelleMalkin points overlap, their clusters are centered differently. The mean and variance of these clusters can serve as another viewpoint into the profile of a blog.

Our POSTFEATURES6 analysis provided us the following observation:

Observation B.2.1 *Posts within a blog tend to take on common network characteristics, which may serve as another means of classification.*

Individual posting patterns may serve as another way of clustering blogs, because different blogs maintain different posting patterns.

B.3 Contributions

These findings have potential applications in many areas. One could argue that the conversation mass metric, defined as the total number of posts in all conversation trees below the point in which the blogger contributed, summed over all conversation trees in which the blogger appears, is a better proxy for measuring influence. This metric captures the mass of the total conversation generated by a blogger, while number of in-links captures only direct responses to the blogger's posts.

For example, we found that BoingBoing, which a very popular blog about amusing things, is engaged in many cascades. Actually, 85% of all BoingBoing posts were cascade initiators. The cascades generally did not spread very far but were wide (e.g., G_{10} and G_{14} in Fig. 6.4). On the

[1] `dlisted.blogspot.com`, a celebrity gossip blog.

[2] `www.MichelleMalkin.com`, a politically conservative blog.

other hand 53% of posts from a political blog MichelleMalkin were cascade initiators. But the cascade here were deeper and generally larger (e.g., G_{117} in Fig. 6.4) than those of BoingBoing.

In summary, our contributions here are:

- We propose using PCA to cluster blogs based on the cascade shapes (CASCADETYPE) appearing. We show that this will successfully separate "humorous" blogs from "politically conservative" blogs.
- We design POSTFEATURES6 to characterize blogs based on their influence in cascades, and show that some blogs have much more varied cascade behavior than others (e.g. MichelleMalkin vs. DListed).

Appendix C

List of publications

Part I: Topology and formation of networks

- M. McGlohon, L. Akoglu, and C. Faloutsos. Weighted Graphs and Disconnected Components: Patterns and a Generator. *SIG-KDD* Las Vegas, Nev., August 2008.
- L. Akoglu. M. McGlohon, and C. Faloutsos. RTM: Laws and a Recursive Generator for Weighted Time-Evolving Graphs. *ICDM IEEE Int'l Conference on Data Mining* Pisa, Italy, Dec. 2008
- U Kang, M. McGlohon, L. Akoglu, and C. Faloutsos. Patterns on the Connected Components of Terabyte-Scale Graphs. Under review.

Part II: Conversation patterns in networks

- M. McGlohon, J. Leskovec, C. Faloutsos, N. Glance, and M. Hurst. Finding patterns in blog shapes and blog evolution. *International Conference on Weblogs and Social Media*. Boulder, Colo., March 2007. January 2007.
- J. Leskovec, J, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Patterns of Cascading Behavior in Large Blog Graphs. *Society of Industrial and Applied Mathematics- Data Mining*. Minneapolis, Minn., April 2007.
- R. Kumar, M. Mahdian, M. McGlohon. Dynamics of Conversations *SIG-KDD*. Washington DC, July 2010.
- M. Goetz, J. Leskovec, M. McGlohon, and C. Faloutsos. Modeling Blog Dynamics. *International Conference on Weblogs and Social Media (ICWSM09)*. San Jose, Cali. May 2009.

Part III: Network effects in action

- L. Akoglu, M, McGlohon, C. Faloutsos. OddBall: Spotting Anomalies in Weighted Graphs. *The 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Hyderabad, India, June 2010. (Chapter 8)
- M. McGlohon, S. Bay, M. Anderle, D. Steier, and C. Faloutsos. SNARE: A Link Analytic System for Graph Labeling and Risk Detection *SIG-KDD* Paris, France. June 2009. (Chapter 9)
- M. McGlohon, N. Glance, and Z. Reiter. Star Quality: Aggregating Reviews to Rank Products and Merchants. *International Conference on Weblogs and Social Media (ICWSM10)*, Washington DC, May 2010. (Chapter 10)

Appendices

- M. McGlohon and M. Hurst. Community Structure and Information Flow in Usenet: Improving analysis with a thread ownership model. *International Conference on Weblogs and Social Media (ICWSM09)*. San Jose, Calif. May 2009.
- M. McGlohon and M. Hurst. Considering the Sources: Comparing linking patterns in Usenet and blogs. *International Conference on Weblogs and Social Media (ICWSM09)*. San Jose, Calif. May 2009.

Bibliography

[1] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 U.S. election: divided they blog. In *LinkKDD '05: Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43, 2005.

[2] Eytan Adar. GUESS: a language and interface for graph exploration. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 791–800, New York, NY, USA, 2006. ACM. ISBN 1595933727. doi: 10.1145/1124772.1124889. URL `http://dx.doi.org/10.1145/1124772.1124889`.

[3] Eytan Adar and Lada A. Adamic. Tracking information epidemics in blogspace. In *WI '05: Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 207–214, Washington, DC, USA, 2005. IEEE Computer Society. doi: http://dx.doi.org/10.1109/WI.2005.151. URL `http://dx.doi.org/http://dx.doi.org/10.1109/WI.2005.151`.

[4] Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. In *SIGMOD*, pages 37–46, 2001.

[5] L. Akoglu, M. McGlohon, and C. Faloutsos. RTM: Laws and a recursive generator for weighted time-evolving graphs. *Carnegie Mellon University Technical Report*, Oct, 2008.

[6] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Rtm: Laws and a recursive generator for weighted time-evolving graphs. In *International Conference on Data Mining*, December 2008.

[7] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Anomaly detection in large graphs. In *CMU-CS-09-173 Technical Report*, 2009.

[8] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *In Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, June 2010.

[9] Reka Albert and Albert L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74, 2002. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0106096`.

[10] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[11] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States of America*,

97(21):11149–11152, October 2000. ISSN 0027-8424. doi: 10.1073/pnas.200327197. URL http://dx.doi.org/10.1073/pnas.200327197.

[12] Nikolay Archak, Anindya Ghose, and Panagiotis G. Ipeirotis. Show me the money!: deriving the pricing power of product features by mining consumer reviews. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 56–65, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1281192.1281202. URL http://dx.doi.org/http://doi.acm.org/10.1145/1281192.1281202.

[13] Andreas Arning, Rakesh Agrawal, and Prabhakar Raghavan. A linear method for deviation detection in large databases. In *KDD*, pages 164–169, 1996.

[14] Lars Backstrom, Ravi Kumar, Cameron Marlow, Jasmine Novak, and Andrew Tomkins. Preferential behavior in online groups. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 117–128, New York, NY, USA, 2008. ACM. doi: http://doi.acm.org/10.1145/1341531.1341549. URL http://dx.doi.org/http://doi.acm.org/10.1145/1341531.1341549.

[15] N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases*. Hafner, New York, second edition, 1975.

[16] A. L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435, 2005.

[17] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999. URL http://view.ncbi.nlm.nih.gov/pubmed/10521342.

[18] Albert-Laszlo Barabasi. *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume Books, April 2003.

[19] Albert-Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005. URL http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0505371.

[20] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley and Sons, Chichester, New York, 1994.

[21] Frank M. Bass. A new product growth for model consumer durables. *Management Science*, 15(5):215–227, 1969.

[22] Stephen Bay, Krishna Kumaraswamy, Markus G. Anderle, Rohit Kumar, and David M. Steier. Large scale detection of irregularities in accounting data. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 75–86, Washington, DC, USA, 2006. IEEE Computer Society. doi: http://dx.doi.org/10.1109/ICDM.2006.93. URL http://dx.doi.org/http://dx.doi.org/10.1109/ICDM.2006.93.

[23] BazaarVoice. Bazaarvoice: Ratings and reviews. urlhttp://www.bazaarvoice.com/products/interaction-suite/ratings-and-reviews, 2010.

[24] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *KDD*, pages 16–24,

2008.

[25] R. Behrman and K. Carley. Modeling the structure and effectiveness of intelligence organizations: Dynamic information flow simulation. In *Proceedings of the 8th International Command and Control Research and Technology Symposium.*, 2003. URL http://www.casos.cs.cmu.edu/publications/papers/behrman_2003_modelingstructure.pdf.

[26] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1281192.1281206. URL http://dx.doi.org/http://doi.acm.org/10.1145/1281192.1281206.

[27] T. Bell and J. Carcello. A decision aid of assessing the likelihood of fraudulent financial reporting. *Auditing: A journal of practice and theory*, 19:169–184, 2000.

[28] M. Beneish. The detection of earnings manipulation. *Financial Analysts Journal*, 55(5): 24–36, 1999.

[29] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[30] Zhiqiang Bi, Christos Faloutsos, and Flip Korn. The "DGX" distribution for mining massive, skewed data. *KDD*, August 2001.

[31] BIGresearch. Word of mouth influences most electronics, apparel purchases, according to rama survey. http://www.bigresearch.com/news/bignrf120709.htm, December 2009.

[32] Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A theory of fads, fashion, custom, and cultural change in informational cascades. *Journal of Political Economy*, 100(5):992–1026, October 1992.

[33] Marián Boguná and Romualdo P. Satorras. Epidemic spreading in correlated complex networks. *Phys. Rev. E*, 66(4):047104, 2002. doi: 10.1103/PhysRevE.66.047104. URL http://dx.doi.org/10.1103/PhysRevE.66.047104.

[34] B. Bollobas. *Random Graphs*. Cambridge, 2001.

[35] B. Bollobas and O. Riordan. *Mathematical Results on Scale-Free Random Graphs*, pages 1–37. Wiley–WCH, 2002.

[36] Bela Bollobas. *Modern Graph Theory*. Springer, corrected edition, July 1998. ISBN 0387984887. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0387984887.

[37] R. Bolton and D. Hand. Statistical fraud detection: A review, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.7.9050.

[38] Richard J. Bolton and David J. Hand. Unsupervised profiling methods for fraud detection, 2001. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.5743.

[39] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis T. Paras. Link analysis ranking: algorithms, theory, and experiments. *ACM Trans. Inter. Tech.*, 5(1):231–297, 2005. doi: http://doi.acm.org/10.1145/1052934.1052942. URL `http://dx.doi.org/http://doi.acm.org/10.1145/1052934.1052942`.

[40] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.

[41] J. R. Busemeyer and J. T. Townsend. Decision field theory: a dynamic-cognitive approach to decision making
in an uncertain environment. *Psychological Review*, 100(3):432–459, July 1993. URL `http://view.ncbi.nlm.nih.gov/pubmed/8356185`.

[42] George Casella and Roger Berger. *Statistical Inference*. Duxbury Resource Center, June 2001.

[43] Deepayan Chakrabarti. Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD*, pages 112–124, 2004.

[44] Deepayan Chakrabarti. *Tools for Large Graph Mining*. PhD thesis, Carnegie Mellon University, June 2005.

[45] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), 2006. ISSN 0360-0300. doi: 10.1145/1132952.1132954. URL `http://dx.doi.org/10.1145/1132952.1132954`.

[46] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *SDM*, 2004.

[47] Soumen Chakrabarti, Byron E. Dom, S. Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins, David Gibson, and Jon K. Einberg. Mining the web's link structure. *Computer*, 32(8):60–67, 1999. doi: http://dx.doi.org/10.1109/2.781636. URL `http://dx.doi.org/http://dx.doi.org/10.1109/2.781636`.

[48] Vineet Chaoji, Mohammad A. Hasan, Saeed Salem, and Mohammed J. Zaki. Sparcl: Efficient and effective shape-based clustering. In *ICDM*, 2008.

[49] Duen H. Chau, Shashank Pandit, and Christos Faloutsos. Detecting fraudulent personalities in networks of online auctioneers. *PKDD*, 2006.

[50] Amitabh Chaudhary, Alexander S. Szalay, and Andrew W. Moore. Very fast outlier detection in large multidimensional data sets. In *DMKD*, 2002.

[51] H. C. Chen, M. Magdon-Ismail, M. Goldberg, and W. A. Wallace. Inferring agent dynamics from social communication network. In *Proc. 9th WebKDD*, 2007.

[52] Pei-Yu Chen, Samita Dhanasobhon, and Michael D. Smith. All Reviews are Not Created Equal: The Disaggregate Impact of Reviews and Reviewers at Amazon.Com. *SSRN eLibrary*, 2008.

[53] Judith Chevalier and Dina Mayzlin. The effect of word of mouth on sales: Online book reviews. *Journal of Marketing Research*, 43:345–354, 2006.

[54] Yun Chi, Shenghuo Zhu, Xiaodan Song, Junichi Tatemura, and Belle L. Tseng. Struc-

tural and temporal analysis of the blogosphere through community factorization. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 163–172, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1281192.1281213. URL http://dx.doi.org/http://doi.acm.org/10.1145/1281192.1281213.

[55] Aaron Clauset, Cosma R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661+, Feb 2009. ISSN 00361445. doi: 10.1137/070710111. URL http://dx.doi.org/10.1137/070710111.

[56] Trevor Cohn. *Scaling Conditional Random Fields for Natural Language Processing*. PhD thesis, University of Melbourne, 2007.

[57] Cristian Danescu-Niculescu-Mizil, Gueorgi Kossinets, Jon Kleinberg, and Lillian Lee. How opinions are received by online communities: A case study on Amazon.com helpfulness votes. In *Proceedings of WWW*, pages 141–150, 2009.

[58] Tamraparni Dasu and Theodore Johnson. *Exploratory Data Mining and Data Cleaning*. Wiley-Interscience, 1 edition, May 2003. ISBN 0471268518. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0471268518.

[59] Shay David and Trevor Pinch. Six degrees of reputation: The use and abuse of online review and recommendation systems. *First Monday*, 11(3), 2006.

[60] Patricia M. Dechow, Weili Ge, Chad R. Larson, and Richard G. Sloan. Predicting material account manipulations. *AAA 2008 Financial Accounting and Reporting Section (FARS)*, 2008.

[61] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[62] Z. Dezsö, E. Almaas, A. Lukács, B. Rácz, I. Szakadát, and A. L. Barabási. Dynamics of information access on the web. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 73(6):066132, 2006. doi: 10.1103/PhysRevE.73.066132. URL http://link.aps.org/abstract/PRE/v73/e066132.

[63] Mingzhou Ding and Weiming Yang. Distribution of the first return time in fractional brownian motion and
its application to the study of on-off intermittency. *Phys. Rev. E*, 52(1):207–213, Jul 1995. doi: 10.1103/PhysRevE.52.207. URL http://dx.doi.org/10.1103/PhysRevE.52.207.

[64] Pedro Domingos and Matt Richardson. Mining the network value of customers. *KDD*, pages 57–66, 2001.

[65] D. Dooley and G. Lamont. Pwc 2005 securities litigation study. Technical report, PricewaterhouseCoopers LLP, 2006.

[66] R. Durrett. *Random Graph Dynamics*. Cambridge, 2006.

[67] William Eberle and Lawrence B. Holder. Discovering structural anomalies in graph-based

data. In *ICDM Workshops*, pages 393–398, 2007.

[68] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[69] Alex Fabrikant, Elias Koutsoupias, and Christos H. Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the internet. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pages 110–122, London, UK, 2002. Springer-Verlag.

[70] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. *SIGCOMM*, pages 251–262, Aug 1999.

[71] Andrew Fast, Lisa Friedland, Marc Maier, Brian Taylor, David Jensen, Henry G. Goldberg, and John Komoroske. Relational data pre-processing techniques for improved securities fraud detection. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 941–949, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1281192.1281293. URL `http://dx.doi.org/http://doi.acm.org/10.1145/1281192.1281293`.

[72] Tom Fawcett and Foster J. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.2902`.

[73] Andrew T. Fiore. Observed behavior and perceived value of authors in usenet newsgroups: Bridging the gap. In *in Usenet Newsgroups: Bridging the Gap. Proceedings of CHI 2001*, pages 323–330. ACM Press, 2002.

[74] Danyel Fisher, Marc A. Smith, and Howard T. Welser. You are who you talk to: Detecting roles in usenet newsgroups. In *HICSS*, 2006.

[75] Gary Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35(3), March 2002.

[76] Geoffrey A. Fowler and Joseph D. Avila. On the internet, everyone's a critic but they're not very critical. *Wall Street Journal*, October 2009.

[77] Lisa Friedland and David Jensen. Finding tribes: identifying close-knit individuals from employment patterns. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 290–299, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-609-7. doi: 10.1145/1281192.1281226. URL `http://dx.doi.org/10.1145/1281192.1281226`.

[78] Michael Gamon, Sumit Basu, Dmitriy Belenko, Danyel Fisher, Matthew Hurst, and Arnd C. Konig. Blews: Using blogs to provide context for news articles. In *International Conference on Weblogs and Social Media*, 2008.

[79] Anindya Ghose and Panagiotis G. Ipeirotis. Designing novel review ranking systems: predicting the usefulness and impact of reviews. In *ICEC '07: Proceedings of the ninth international conference on Electronic commerce*, pages 303–310, New York, NY, USA, 2007. ACM. doi: http://doi.acm.org/10.1145/1282100.1282158. URL `http://dx.doi.org/http://doi.acm.org/10.1145/1282100.1282158`.

[80] Amol Ghoting, Matthew E. Otey, and Srinivasan Parthasarathy. LOADED: Link-based outlier and anomaly detection in evolving data sets. In *ICDM*, 2004.

[81] Amol Ghoting, Srinivasan Parthasarathy, and Matthew E. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16 (3):349–364, 2008.

[82] Eric Gilbert and Karrie Karahalios. Understanding deja reviewers. In *CSCW '10: The 2010 ACM Conference on Computer Supported Cooperative Work*, New York, NY, USA, 2010. ACM.

[83] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *PNAS*, 99:7821, 2002.

[84] Natalie S. Glance, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. Deriving marketing intelligence from online discussion. In *KDD*, 2005.

[85] David Godes and Dina Mayzlin. Using online conversations to study word-of-mouth communication. *Marketing Science*, 23(4):545–560, 2004. doi: http://dx.doi.org/10.1287/mksc.1040.0071. URL `http://dx.doi.org/http://dx.doi.org/10.1287/mksc.1040.0071`.

[86] David Godes and Dina Mayzlin. Firm-Created Word-of-Mouth Communication: Evidence from a Field Test. *MARKETING SCIENCE*, 2009. doi: 10.1287/mksc.1080.0444. URL `http://mktsci.journal.informs.org/cgi/content/abstract/mksc.1080.0444v1`.

[87] Michaela Goetz, Jure Leskovec, Mary McGlohon, and Christos Faloutsos. Modeling blog dynamics. In *International Conference on Weblogs and Social Media*, May 2009.

[88] K. I. Goh, Y. H. Eom, H. Jeong, B. Kahng, and D. Kim. Structure and evolution of online social relationships: Heterogeneity in warm discussions, Jan 2006. URL `http://arxiv.org/abs/physics/0601223`.

[89] W. Golden, S. Skalak, and M. Clayton. *A Guide to Forensic Accounting Investigation*. John Wiley & Sons, Hoboken, N.J., 2006.

[90] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 2001.

[91] B. Golub and M. O. Jackson. The power of selection bias in explaining the structure of observed Internet diffusions. *Proc. National Academy of Sciences*, To appear.

[92] Vicenc Gómez, Andreas Kaltenbrunner, and Vicente López. Statistical analysis of the social network and discussion threads in slashdot. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 645–654, New York, NY, USA, 2008. ACM. ISBN 9781605580852. doi: 10.1145/1367497.1367585. URL `http://dx.doi.org/10.1145/1367497.1367585`.

[93] M. Granovetter. Threshold models of collective behavior. *Am. Journal of Sociology*, 83 (6):1420–1443, 1978.

[94] H. Grove and T. Cook. A statistical analysis of financial ratio red flags. *Oil, Gas and*

Energy Quarterly, 53(2):3212–3346, 2004.

[95] D. Gruhl, David Liben-Nowell, R. Guha, and A. Tomkins. Information diffusion through blogspace. *SIGKDD Explor. Newsl.*, 6(2):43–52, December 2004. doi: 10.1145/1046456.1046462. URL http://dx.doi.org/10.1145/1046456.1046462.

[96] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 403–412, New York, NY, USA, 2004. ACM. doi: http://doi.acm.org/10.1145/988672.988727. URL http://portal.acm.org/citation.cfm?id=988672.988727.

[97] T. E. Harris. Contact interactions on a lattice. *Annals of Proabability*, 2:969–988, 1974.

[98] T. E. Harris. *The Theory of Branching Processes*. Dover, 2002.

[99] D. Hawkins. Identification of outliers. *Chapman and Hall*, 1980.

[100] Harold S. Heaps. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, 1978.

[101] Herbert W. Hethcote. The mathematics of infectious diseases. *SIAM Rev.*, 42(4):599–653, 2000. doi: http://dx.doi.org/10.1137/S0036144500371907. URL http://dx.doi.org/http://dx.doi.org/10.1137/S0036144500371907.

[102] Shawndra Hill and Foster Provost. The myth of the double-blind review?: author identification using only citations. *SIGKDD Explor. Newsl.*, 5(2):179–184, December 2003. ISSN 1931-0145. doi: 10.1145/980972.981001. URL http://dx.doi.org/10.1145/980972.981001.

[103] Shawndra Hill, Foster Provost, and Chris Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 22(2):256–275, 2006.

[104] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM.

[105] Nan Hu, Paul A. Pavlou, and Jennifer Zhang. Can online reviews reveal a product's true quality?: empirical findings and analytical modeling of online word-of-mouth communication. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 324–330, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1134707.1134743. URL http://dx.doi.org/http://doi.acm.org/10.1145/1134707.1134743.

[106] Tianming Hu and Sam Y. Sung. Detecting pattern-based outliers. *Pattern Recognition Letters*, 24(16), 2003.

[107] Bernardo A. Huberman and Lada A. Adamic. Growth dynamics of the world-wide web. *Nature*, 399, 1999.

[108] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *KDD '07: Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.7290.

[109] David D. Jensen, Andrew S. Fast, Brian J. Taylor, and Marc E. Maier. Automatic identification of quasi-experimental designs for discovering causal knowledge. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 372–380, New York, NY, USA, 2008. ACM.

[110] Ruoming Jin, Chao Wang, Dmitrii Polshakov, Srinivasan Parthasarathy, and Gagan Agrawal. Discovering frequent topological structures from graph datasets. In *KDD*, 2005.

[111] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 219–230, New York, NY, USA, 2008. ACM. doi: 10.1145/1341531.1341560. URL http://dx.doi.org/10.1145/1341531.1341560.

[112] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 1998.

[113] I. T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[114] U. Kang, Mary McGlohon, Leman Akoglu, and Christos Faloutsos. Patterns on the connected components of terabyte-scale graphs. *Under review*, July, 2010.

[115] Amit Karandikar, Akshay Java, Anupam Joshi, , Tim F. Yelena Yesha, and Yaacov Yesha. Second Space: A Generative Model For The Blogosphere. In *ICWSM*. AAAI, 2008.

[116] Elihu Katz and Paul Lazarsfeld. *Personal Influence: The Part Played by People in the Flow of Mass Communications*. Transaction Publishers, October 1955. ISBN 1412805074. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/1412805074.

[117] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, 2003.

[118] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, pages 343–359, 1991. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=130801.

[119] Jeffrey O. Kephart and Steve R. White. Measuring and modeling computer virus prevalence. *Security and Privacy, IEEE Symposium on*, 0:2, 1993. doi: http://doi.ieeecomputersociety.org/10.1109/RISP.1993.287647. URL http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/RISP.1993.287647.

[120] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 423–430, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

[121] Jon Kleinberg. Bursty and hierarchical structure in streams. In *KDD*, 2002.

[122] Jon Kleinberg. Complex networks and decentralized search algorithms. In *Proc. International Congress of Mathematicians*, 2006.

[123] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. URL http://citeseer.ist.psu.edu/

kleinberg99authoritative.html.

[124] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The Web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–17, 1999.

[125] Edwin M. Knorr and Raymond T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, pages 392–403, 1998.

[126] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD*, pages 289–300, May 1997.

[127] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proc. 41st FOCS*, pages 57–65, 2000.

[128] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 568–576, New York, NY, USA, 2003. ACM Press. doi: http://doi.acm.org/10.1145/775152.775233. URL http://dx.doi.org/http://doi.acm.org/10.1145/775152.775233.

[129] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowedge Discover and Data Mining*, pages 611–617, New York, 2006.

[130] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Core algorithms in the CLEVER system. *ACM Trans. Inter. Tech.*, 6(2):131–152, 2006. doi: http://doi.acm.org/10.1145/1149121.1149123. URL http://dx.doi.org/http://doi.acm.org/10.1145/1149121.1149123.

[131] Ravi Kumar, Mohammad Mahdian, and Mary McGlohon. Dynamics of conversations. In *KDD*, July 2010.

[132] Ar Lazarevic, Aysel Ozgur, Levent Ertoz, Jaideep Srivastava, and Vipin Kumar. A comparative study of anomaly detection schemes in network intrusion detection. In *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.91.2580.

[133] J. Leskovec, L. Backstrom, and J. M. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *Proc. 15th KDD*, pages 497–506, 2009.

[134] Jure Leskovec. *Dynamics of Large Networks*. PhD thesis, Carnegie Mellon University, September 2008.

[135] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, and Christos Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. *PKDD*, pages 133–145, 2005.

[136] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, New York, NY, USA, 2005. ACM Press.

[137] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral

marketing. In *EC '06: Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 228–237, New York, NY, USA, 2006. ACM Press. doi: http://doi.acm.org/10.1145/1134707.1134732. URL http://dx.doi.org/http://doi.acm.org/10.1145/1134707.1134732.

[138] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie Glance, and Matthew Hurst. Cascading behavior in large blog graphs: Patterns and a model, October 2006.

[139] Jure Leskovec, Ajit Singh, and Jon Kleinberg. Patterns of influence in a recommendation network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2006.

[140] Jure Leskovec, Mary McGlohon, Christos Faloutsos, Natalie Glance, and Matthew Hurst. Cascading behavior in large blog graphs. *SIAM International Conference on Data Mining (SDM)*, 2007.

[141] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *KDD 2008*, Las Vegas, Nevada, USA, 2008.

[142] David Liben-Nowell and Jon Kleinberg. Tracing the flow of information on a global scale using Internet chain-letter data. *Proceedings of the National Academy of Sciences*, 105 (12):4633–4638, March 2008.

[143] T. M. Liggett. *Interacting Particle Systems*. Springer-Verlag, first edition, 1985.

[144] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA, 2005. ACM.

[145] Chao Liu, Xifeng Yan, Hwanjo Yu, Jiawei Han, and Philip S. Yu. Mining behavior graphs for "backtrace" of noncrashing bugs. In *SDM*, 2005.

[146] Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. Low-quality product review detection in opinion summarization. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, 2007.

[147] Sofus A. Macskassy and Foster Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *Proceedings of the NAACSOS Conference*, June 2005.

[148] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, August 1982. URL http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=AS\\IN/0716711869.

[149] Mary McGlohon. Data mining disasters: A report. In *Proceedings of the Second Annual Intercalary Workshop about Symposium on Robot Dance Party of Conference in Celebration of Harry Q. Bovik's 2^6th Birthday (ACH SIGBOVIK 2008)*, April 2008.

[150] Mary McGlohon and Matthew Hurst. Considering the sources: Comparing linking patterns in usenet and blogs. In *International Conference on Weblogs and Social Media*, May 2009.

[151] Mary McGlohon and Matthew Hurst. Community structure and information flow in

usenet: Improving analysis with a thread ownership model. In *International Conference on Weblogs and Social Media*, May 2009.

[152] Mary McGlohon, Jure Leskovec, Christos Faloutsos, Matthew Hurst, and Natalie Glance. Finding patterns in blog shapes and blog evolution. In *International Conference on Weblogs and Social Media*, Boulder, Colo., March 2007.

[153] Mary McGlohon, Leman Akoglu, and Christos Faloutsos. Weighted graphs and disconnected components: Patterns and a generator. In *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIG-KDD)*, August 2008.

[154] Mary McGlohon, Stephen Bay, Markus Anderle, David Steier, and Christos Faloutsos. Snare: A link analytic system for graph labeling and fraud detection. In *SIGKDD*, June 2009.

[155] Mary McGlohon, Zach Reiter, and Natalie Glance. Star quality: Aggregating reviews to rank products and merchants. In *International Conference on Weblogs and Social Media*, May 2010.

[156] M. Mihail and C. Papadimitriou. The eigenvalue power law, 2002.

[157] S. Milgram. The small-world problem. *Psychology Today*, 2:60–67, 1967.

[158] Evan Miller. How not to sort by average rating. urlhttp://www.evanmiller.org/how-not-to-sort-by-average-rating.html, February 2006.

[159] Gilad Mishne and Natalie Glance. Leave a reply: An analysis of weblog comments. In *In Third annual workshop on the Weblogging ecosystem*, 2006.

[160] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2), 2003.

[161] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6(2/3):161–180, 1995.

[162] Alan L. Montgomery and Christos Faloutsos. Identifying web browsing trends and patterns. *IEEE Computer*, 34(7):94–95, July 2001.

[163] H. D. K. Moonesinghe and Pang-Ning Tan. Outrank: a graph-based outlier detection framework using random walk. *International Journal on Artificial Intelligence Tools*, 17 (1), 2008.

[164] Geoffrey A. Moore. *Crossing the Chasm*. Harper Paperbacks, revised edition, September 2002. ISBN 0060517123. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0060517123`.

[165] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, 1995.

[166] C. W. Mulford and E. E. Comiskey. *The Financial Numbers Game: Detecting Creative Accounting Practices*. John Wiley & Sons, Hoboken, N.J., 2002.

[167] Jennifer Neville, " O. c Simc sek, David Jensen, John Komoroske, Kelly Palmer, and Henry Goldberg. Using relational knowledge discovery to prevent securities fraud. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 449–458, New York, NY, USA, 2005. ACM Press.

doi: http://dx.doi.org/10.1145/1081870.1081922. URL http://dx.doi.org/10.1145/1081870.1081922.

[168] M. E. J. Newman. Threshold effects for two pathogens spreading on a network. *Physical Review Letters*, 95, 2005. URL doi:10.1103/PhysRevLett.95.108701.

[169] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46, 2005. URL doi:10.1080/00107510500052444.

[170] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, 2004. URL http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0308217.

[171] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, pages 144–155, 1994.

[172] Caleb C. Noble and Diane J. Cook. Graph-based anomaly detection. In *KDD*, pages 631–636, 2003.

[173] Jahna Otterbacher. 'helpfulness' in online communities: a measure of message quality. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 955–964, New York, NY, USA, 2009. ACM. doi: http://doi.acm.org/10.1145/1518701.1518848. URL http://dx.doi.org/http://doi.acm.org/10.1145/1518701.1518848.

[174] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. URL citeseer.ist.psu.edu/page98pagerank.html.

[175] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. Anf: A fast and scalable tool for data mining in massive graphs. In *SIGKDD*, Edmonton, AB, Canada, 2002.

[176] Shashank Pandit, Duen H. Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 201–210, New York, NY, USA, 2007. doi: http://doi.acm.org/10.1145/1242572.1242600. URL http://dx.doi.org/http://doi.acm.org/10.1145/1242572.1242600.

[177] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *ICDE*, 2003.

[178] Seung T. Park, David M. Pennock, and Lee C. Giles. Comparing static and dynamic measurements and models of the internet's as topology. In *INFOCOM*, 2004.

[179] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks, 2001. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.7011.

[180] R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Physical Review E*, 65:036104+, 2002. doi: 10.1103/PhysRevE.65.036104. URL http://link.aps.org/abstract/PRE/v65/e036104.

[181] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics in finite size scale-free networks. *Physical Review E*, 65(3):035108+, March 2002. doi: 10.

1103/PhysRevE.65.035108. URL http://dx.doi.org/10.1103/PhysRevE.65.035108.

[182] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani. Dynamical and correlation properties of the internet. *Physical Review Letters*, 87(25):258701+, November 2001. doi: 10.1103/PhysRevLett.87.258701. URL http://dx.doi.org/10.1103/PhysRevLett.87.258701.

[183] David M. Pennock, Gary W. Flake, Steve Lawrence, Eric J. Glover, and C. Lee Giles. Winners don't take all: Characterizing the competition for links on the web. *Proceedings of the National Academy of Sciences of the United States of America*, 99(8):5207–5211, April 2002. doi: 10.1073/pnas.032085699. URL http://dx.doi.org/10.1073/pnas.032085699.

[184] PowerReviews. Powerreviews: Engage, connect, and sell. urlhttp://www.powerreviews.com/reviews.php, 2010.

[185] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing, 2002. URL citeseer.ist.psu.edu/richardson02mining.html.

[186] Everett M. Rogers. *Diffusion of Innovations, 5th Edition*. Free Press, August 2003. URL http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0743222091.

[187] H. Schilit. *Financial Shenanigans: How to Detect Accounting Gimmicks and Fraud in Financial Reports*. McGraw-Hill, 2002.

[188] Manfred Schroeder. *Fractals, Chaos, Power Laws: Minutes From an Infinite Paradise*. W. H. Freeman, 1991.

[189] Michael F. Schwartz and David C. M. Wood. Discovering shared interests among people using graph analysis of global electronic mail traffic. *Communications of the ACM*, 36: 78–89, 1992.

[190] Karlton Sequeira and Mohammed J. Zaki. Admit: anomaly-based data mining for intrusions. In *KDD*, 2002.

[191] Claude E. Shannon and Warren Weaver. *A Mathematical Theory of Communication*. University of Illinois Press, Champaign, IL, USA, 1963.

[192] Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power laws and the as-level internet topology. *IEEE/ACM Trans. Netw.*, 11(4):514–524, 2003. URL http://dblp.uni-trier.de/db/journals/ton/ton11.html\#SiganosFFF03.

[193] Georg Simmel. Fashion. *International Quarterly*, 10:130–150, 1904.

[194] S. Skalak and C. Nestler. Global economic crime survey 2005. Technical report, PricewaterhouseCooper LLP, 2005.

[195] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. *ICDM*, 2005.

[196] Nielsen G. Survey. Word of mouth the most powerful selling tool.

urlhttp://nz.nielsen.com/news/Advertising_Oct07.shtml, October 2007.

[197] Arjun Talwar, Radu Jurca, and Boi Faltings. Understanding user behavior in online feedback reporting. In *EC '07: Proceedings of the 8th ACM conference on Electronic commerce*, pages 134–142, New York, NY, USA, 2007. ACM.

[198] S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A simple conceptual model for the internet topology. In *IEEE Global Telecommunications Conference, 2001. GLOBECOM'01.*, 2001.

[199] Charalampos E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *ICDM*, 2008.

[200] Oren Tsur and Ari Rappoport. Revrank: A fully unsupervised algorithm for selecting the most helpful book reviews. In *International Conference on Weblogs and Social Media (ICWSM09)*, 2009. URL http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/180.

[201] T. C. Turner, M. A. Smith, D. Fisher, and H. T. Welser. Picturing usenet: Mapping computer-mediated collective action. *Journal of Computer-Mediated Communication*, 10 (4), 2005.

[202] Mark Uncles, Andrew Ehrenberg, and Kathy Hammond. Patterns of buyer behavior: Regularities, models, and extensions. *Marketing Science*, 14(3):G71–G78, 1995. URL http://www.jstor.org/stable/184149.

[203] A. Vazquez, Gama J. Oliveira, Z. Dezso, K. I. Goh, I. Kondor, and A. L. Barabasi. Modeling bursts and heavy tails in human dynamics. *Physical Review E*, 73, 2006. URL http://www.citebase.org/abstract?id=oai:arXiv.org:physics/0510117.

[204] Mengzhi Wang, Tara Madhyastha, Ngai H. Chang, Spiros Papadimitriou, and Christos Faloutsos. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. *ICDE*, February 2002.

[205] Yang Wang and Chenxi Wang. Modeling the effects of timing parameters on virus propagation. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 61–66, New York, NY, USA, 2003. ACM. doi: http://doi.acm.org/10.1145/948187.948198. URL http://dx.doi.org/http://doi.acm.org/10.1145/948187.948198.

[206] Yang Wang, Deepayan Chakrabarti, Chenxi Wang, and Christos Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. In *SRDS*, pages 25–34, 2003.

[207] Yi-Chia Wang, M. Joshi, W. Cohen, and C. P. Rose. Recovering implicit thread structure in newsgroup style conversations. In *International Conference on Weblogs and Social Media*, March 2008.

[208] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998. doi: 10.1038/30918. URL http://dx.doi.org/10.1038/30918.

[209] Duncan J. Watts. A simple model of global cascades on random networks. In *Proceedings*

of the National Academy of Sciences of the United States of America, volume 99, pages 5766–5771, 2002.

[210] J. Wells. *Corporate Fraud Handbook: Prevention and Detection.* John Wiley & Sons, Hoboken, N.J., 2004.

[211] Fang Wu and Bernardo A. Huberman. How public opinion forms. Technical report, Social Computing Lab, HP Labs, Palo Alto, CA 94304, USA, September 2008. URL `http://www.hpl.hp.com/research/scl/papers/howopinions/wine.pdf`.

[212] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *ICDM*, 2002.

[213] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. *Understanding belief propagation and its generalizations*, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. ISBN 1558608117. URL `http://portal.acm.org/citation.cfm?id=779352`.

[214] Richong Zhang and Thomas Tran. An entropy-based model for discovering the usefulness of online product reviews. In *WI-IAT '08: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 759–762, Washington, DC, USA, 2008. IEEE Computer Society. doi: http://dx.doi.org/10.1109/WIIAT.2008.149. URL `http://dx.doi.org/http://dx.doi.org/10.1109/WIIAT.2008.149`.

[215] Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 51–57, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/1183614.1183626. URL `http://dx.doi.org/http://doi.acm.org/10.1145/1183614.1183626`.

[216] Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005. URL `http://portal.acm.org/citation.cfm?id=1104523`.

[217] Cai N. Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005. doi: http://dx.doi.org/10.1007/s10796-005-4807-3. URL `http://portal.acm.org/citation.cfm?id=1108451.1108466\&coll=GUIDE\&dl=GUIDE\#`.

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Carnegie Mellon