Making Diffusion Work for You: Finding Culprits, Filling Missing Values, and Classification Sans Text

Shashidhar Sundareisan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

B. Aditya Prakash, Chair
Naren Ramakrishnan
Dhruv Batra

June 17, 2014
Blacksburg, Virginia

Keywords: Data Mining, Social Networks, Epidemiology, Culprits, Missing nodes, Diffusion, Protests, Classification

Making Diffusion Work for You: Finding Culprits, Filling Missing Values, and Classification Sans Text

Shashidhar Sundareisan

(ABSTRACT)

Can we find people infected with the flu virus even though they did not visit a doctor? Can the temporal features of a trending hashtag or a keyword indicate which topic it belongs to without any textual information? Given a history of interactions between blogs and news websites, can we predict blogs posts/news websites that are not in the sample but talk about the "the state of the economy" in 2008? These questions have two things in common: a network (social networks or human contact networks) and a virus (meme, keyword or the flu virus) diffusing over the network. We can think of interactions like memes, hashtags, influenza infections, computer viruses etc., as viruses spreading in a network. This treatment allows for the usage of epidemiologically inspired models to study or model these interactions. Understanding the complex propagation dynamics involved in information diffusion with the help of these models uncovers various non-trivial and interesting results.

In this thesis we propose (a) A fast and efficient algorithm NetFill, which can be used to find quantitatively and qualitatively correct infected nodes, not in the sample and finding the culprits and (b) A method, SansText that can be used to find out which topic a keyword/hashtag belongs to just by looking at the popularity graph of the keyword without textual analysis. The results derived in this thesis can be used in various areas like epidemiology, news and protest detection, viral marketing and it can also be used to reduce sampling errors in graphs.

Acknowledgements

First of all I would like to thank Aditya Prakash, for being a great advisor and a mentor. I have learnt a great deal under his guidance and am thankful for providing me with the right direction in my research. I would also like to thank my committee members Dr. Naren Ramakrishnan and Dr. Dhruv Batra for their valuable inputs in my thesis.

The work presented in Chapter 2 was conducted in collaboration with Dr. Jilles Vreeken, Max-Planck institute and my adviser Dr. B. Aditya Prakash. While the work presented in Chapter 3 was in collaboration with Abhay Rao, Mohammad Akmal Saquib Khan, Dr. Naren Ramakrishnan and Dr. B. Aditya Prakash. I would like to thank all my collaborators for their contributions and also for the permission to include the work in my thesis. I would also like to thank Ting Hua for discussions about the keywords in the protest dataset.

Finally I would like to thank my parents for their blessings and support, without which my journey through life would have been difficult.

Funding and Disclaimer: This work was partially supported by the NSF via grant IIS-1353346, by the NSA (under a Science of Security lablet), by the VT College of Engineering and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D12PC000337.
The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, IARPA, NSA, DoI/NBC, or the US Government.

Attributions

For Chapter 2:
Dr. Jilles Vreeken, Independent Research Group Leader, Saarland University, Max-Planck Institute of Informatics, contributed towards the MDL score we describe in 2. Dr. B. Aditya Prakash, my adviser was key to help me construct the approach and algorithm to solve the problem.

For Chapter 3:
Abhay Rao (MS, Department of Computer Engineering) and Mohammad Saquib Akmal Khan (MS, Department of Computer Science) worked with me on a group project for CS6604: Data Mining in Large Networks and Time Series in Fall 2013, taught by Dr. B. Aditya Prakash. The work culminated to a conference paper which is to appear in ASONAM-2014 [Sundareisan et al., 2014]. Abhay's main contributions include help in collecting the dataset and discussions. Saquib was involved in the discussions and curve fitting. We had access to the Twitter dataset from South America with the help of Dr. Naren Ramakrishnan (Department of Computer Science). He also contributed towards the collection of the protest dataset with his knowledge of the EMBERS project and problem formulation. Dr. B. Aditya Prakash, was my adviser and was instrumental in giving the right direction and discussing ideas about our approach.

Contents

List of Figures

Note: All figures and or images in this thesis were created by the author unless or otherwise specified.

List of Tables

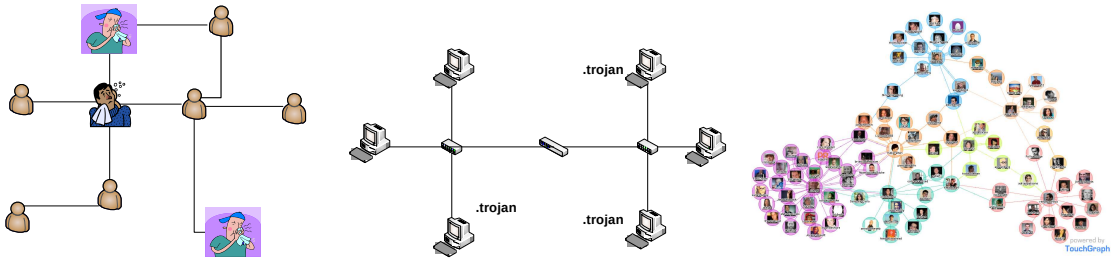
Chapter 1

Introduction

This thesis describes how we can take advantage of propagation dynamics in the form of information diffusion in different fields like epidemiology and social media. By using models inspired by epidemiology to model the behavior of a contagion spreading in a network we try to reverse engineer the infection spread in the network. This reverse engineering is done with the help of only historical data and a snapshot of the infected nodes in the graph. Another application that results from such modeling is that we can infer the topic of a keyword just by using temporal features extracted from its popularity time series. In short we try to answer questions like (a) Can the topology of the underlying network help us find people who help spread a rumor even though we do not have evidence that they spread the rumor? (b) Do keywords belonging to politics or Flu have a different and distinguishing patterns in the popularity time series that helps us to infer the topic?

So why are these questions important? With the growth of the internet, social media platforms like Twitter, Facebook, Google Plus etc have made a way into our daily lives. The communication over social media is competing with traditional communication methods. There are 255 million active users on Twitter, Facebook has over a Billion user accounts while Google+ has over 350 Million users[1]. Unlike traditional communication media, social media platforms collect an enormous collection of data and meta-data which can be used to understand how humans communicate and behave.

In this chapter, we describe the motivations of the thesis in Section 1.1, followed by a brief overview of the thesis in Section 1.2 and finally in Section 1.3, we discuss the contributions and applications of the work.



(a) Flu in human network (b) Trojan in computer network (c) Hashtag in Twitter

Figure 1.1: Various examples[2] *of contagions in networks/graphs.*

1.1 Motivation

As illustrated in Figure 1.1, diffusion in a network is a very versatile concept which can be applied to various fields like network security, epidemiology, Biology, and social media analytics. In epidemiology, a virus like influenza spreading in a human contact network can be thought of diffusion. In computer networks, propagation of a computer virus can be thought of diffusion. While in biology, the information could be the increase of the concentrations of the proteins in the gene-regulatory network. Diffusion can also occur in social media platforms [2] in the form of information diffusion.

Propagation of a contagion can be carried out in various ways in social media. A meme is a phrase, a group of words or any element of culture that is used in disseminating information from person to person. A hashtag is a '#' symbol followed by a word or a group of words without a space used mainly to highlight a special meaning. A retweet or a share means that a person is reposting someone else's content. In all these cases information is said to have diffused through a network.

With the data obtained from social networks we can construct various graphs or networks for different data mining purposes. For example, from the twitter datasets we can find the who follows whom network in which there is a directed edge from the follower to the followee. We could also have the retweet graph which is a weighted and directed graph, where the weights denote the probability that a person will retweet the parent's tweet. In Facebook we have a network of friends, where an edge exists between two user accounts if they are friends. Other than these popular datasets we can also have various networks constructed from Memetracker, Flixster, Google Plus, Yahoo meme!, and a host of other data sources.

So what information can be extracted if we know how the contagion spreads or propagates? There are various problems that come up when we study propagation dynamics in graphs. A few interesting problems are (a) Who started the spread of the flu virus? (b) Who are the most influential people in the graph? (c) Which nodes in the graph are easily susceptible

[1] http://www.statisticbrain.com/

[2] Hashtag in Twitter: Image obtained from www.touchgraph.com used under fair use, 2014

to infection? (d) Can we infer the structure of the graph just by knowing when nodes get the information? (e) Who are the best set of people I should advertise my product? In this thesis we explore only a small subset of these questions.

1.1.1 Thesis Statement

Leveraging the dynamics of propagation on networks helps in designing faster and better algorithms for locating patient zeros, finding missing-data, and domain classification in applications like social media and epidemiology.

To elaborate, dynamics of propagation is the way in which a contagion diffuses or propagates in a network. In the first part of the thesis we are interested in using this dynamics in a network by applying models inspired by epidemiology to find patient zeros and missing data in epidemiology and social media by reconstructing the spread of the contagion. Patient zeros in epidemiology are the nodes from which the virus initially started propagating (For example whoever started the H1N1 influenza epidemic). This would be analogous to the epicenter of an earthquake; the place where the event started propagating. Missing nodes is a set of nodes that were infected, but were not detected as being infected in the sampling process. Twitter user accounts that actually talk about a hashtag, but do not show up in a database of user tweets are an instance of 'missing nodes'. Finding such accounts would be very beneficial, as we get access to only a small percentage ($\leq 10\%$) of the actual tweets even when we purchase a firehose of Twitter data from companies like DataSift. In the second part of the thesis we are interested in domain classification with the help of dynamics of contagion propagation in a network. By understanding this dynamics, we can extract some temporal features from the time series that describes the number of infected people and the popularity of the keyword. Using these features we can classify keywords into topics without the help of the text (For example does a given hashtag belongs to politics or technology).

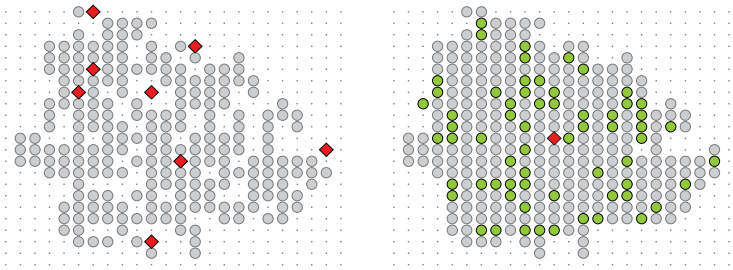
1.2 Overview

The thesis is structured as follows: First we discuss the problem of finding missing nodes and the culprits in Chapter 2. In Chapter 3, we discuss how to use the temporal features that we extract from the time series that describes the popularity of the hashtag for classification into various topics. We provide a brief overview of these two chapters in the following sections.

1.2.1 Finding Missing Nodes with the Help of Culprits

In this chapter we find the source of the epidemic in the graph and reconstruct the trail of the epidemic just by using the graph topology and the set of infected nodes. Doing so we can also uncover nodes in the graph that were not reported as infected. As mentioned in Section 1.1, the analogy of an epidemic can be extended to various types of information that can be diffused in the network. In such cases the notion of infected means that the node is aware of the information. So just by having the topology of who cites whom in the world of blogs and news websites and a sample of all the blogs that talk about a meme like "the state of the economy", our goal is to find the probable set of websites that started this epidemic (meme) and also the set of blog and web sites that were missed in the sampling process by reverse engineering the flow of information in the network.

Our proposed approach involves the use of the Minimum Description Length Principle which involves encoding the model and the data in bits. The MDL principle is similar to Occam's razor in the sense that we choose a model that is the simplest explanation of the events that happened, which in turn would have the smallest number of bits in its representation. We discuss the details of the MDL principle and the encoding that we use in Section 2.3.



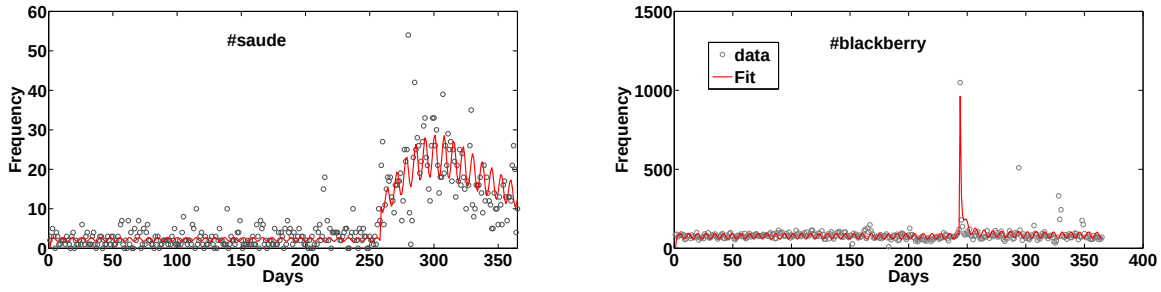
(a) NetSleuth (b) NetFill

Figure 1.2: Finding culprits with missing infections: (a) the state of the art, NetSleuth, finds 8 sources (red diamonds). (b) Our method, NetFill*, finds* both *the right number of sources (one) and recovers the missing nodes with high precision (green).*

In this chapter, we propose NetFill, a near linear, fast, and efficient algorithm that not only finds the culprits of the epidemic but it also finds the missing nodes. NetFill automatically finds the number of missing nodes to be found. We discuss the extensive set of experiments that we performed in various synthetic as well as real graphs in Section 2.6. We tested NetFill with various baselines on real as well as synthetic cascades in various real as well as synthetic graphs and found that NetFill does the best job in finding the correct missing nodes quantitatively as well as qualitatively, and it also finds a good set of patient-zeros. For the *MemeTracker* [Leskovec et al., 2009] dataset (a dataset of blog posts and news articles which tracks memes) in a case study we found websites like "www.chicagotribune.com" as well as "www.nbcbayarea.com" and some blog posts that talk about "the state of the economy", but these articles were absent in the *MemeTracker* dataset. With respect to

precision we perform between 1.5x to 7x better as compared to the baselines in all of the datasets(We perform better in real graphs). We show the performance of NetFill along with that of the state of the art in Figure 1.2.

1.2.2 Topic Classification using Temporal Features



(a) Flu (b) Technology

Figure 1.3: Modeling using SansText*: We can see that* SansText *is a good model for temporal spikes and can capture the differences between the two topics.*

The goal in this chapter is to prove that keywords can be mapped to different topics without the help of textual analysis. Keywords in general can belong to different topics. A good amount of work has been done in order to infer the topic from textual analysis. When a keyword is becomes popular in a social network over a small period of time the keyword is said to be trending. We can visualize how this trending pattern occurs by looking at the its popularity time series. The popularity time series is the number of mentions of the keyword over a time period. These plots are similar to the ones shown on Google trends (see http://www.google.com/trends/). The question that we are trying to answer is by extracting various temporal features from this graph can we find out what the topic is?

News websites and News channels have started to look for alternate sources to gather information about events. Twitter and other social media outlets are becoming a rich source which informs them about various events happening across the world in real time. Social media is also being used to increase awareness about various social issues and to gather masses for protests. It has also becoming a platform for citizens to voice their concerns and dissatisfaction as we saw in Arab Spring revolution in 2012, protests in Venezuela in the year 2012/2013 and civil activism in India also in 2012. Thus knowing which topic a trending keyword belongs to or to understand if a protest becomes violent or not, is really important.

The main problem with the textual analysis approach is that we will not be able to classify new keywords easily. Another issue is that we need to save the whole vocabulary for different languages in order for them to work. In contrast the approach described in this chapter, SansText, can be used on new keywords and it needs very little space. As we can see

from Figure 1.3, that SansText is a good model for contagions and that we can use it to differentiate between topics (Technology and Flu in this case). From the experiments in Section 3.6, we observe that SansText performs better than all the baselines, some of which are not trivial. On average SansText performed way more than twice as better than a naive predictor in all classification problems.

The work in this chapter is to appear in the Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM-2014 [Sundareisan et al., 2014].

1.3 Contributions

The contributions in the thesis can be summarized as follows:

1. In Chapter 2, Finding Missing Nodes with the Help of Culprits, the main contributions are
   (a) We are the first to formulate the problem of finding missing infected nodes and concealed culprits in terms of the Minimum Description Length principle (See Section 2.4.5).
   (b) We present a fast and effective near-linear time algorithm NetFill, based on a novel alternating minimization approach to *automatically* recover both: high quality missing nodes and the correct number and identity of the seeds. We give solutions for both cases where we do, and where we do not know the sampling rate and we perform extensive experiments using both simulated and *real* cascades on both synthetic and real networks (See Sections 2.6.2, 2.6.3 and 2.6.4).
2. In Chapter 3, Topic Classification using Temporal Features, the main contributions include
   (a) We formulate the domain classification problem using activity profiles, and propose a simple yet powerful and efficient low-cost approach based on learning an aggregate information diffusion model (see Section 3.4).
   (b) We demonstrate the effectiveness of our approach, SansText via first classifying simple popular keywords to domains and then with protest event data from South America. We compare against several baselines, explore the robustness of our approach to different parameter sets of our model and to limited data (see Section 3.5).

Applications: The missing nodes problem has applications in viral marketing, sampling in networks, epidemiology, and research on social networks. In social networks it is difficult

to collect complete data because of size limitations as well as limitations imposed by social media platforms. For example twitter API provides only a sample of all results. With methods discussed in Chapter 2, the dataset can be expanded and hence there would be more accurate data to perform experiments with. For epidemiology, not all patients report to the doctor when they are ill. In such cases we can use NetFill, to infer the number of infected patients more accurately.

The classification problem discussed in Section 3.4 has many consequences for the news reporters. Since there is active participation in the dissipation of news on social media, we can use these methods to find out which topic the trending trending keywords belongs to. Such tools would also help the government plan for different types of protests and events happening in the country.

Chapter 2

Finding Missing Nodes with the Help of Culprits

In this chapter, we use propagation dynamics to find missing nodes and culprits in uncertain graphs. That is we try to reverse engineer the spread of the virus in the network just by observing who got infected in the graph. As discussed in the Chapter 1, the diffusion could be caused by the propagation of a virus, computer virus, information, contagion etc. In this chapter we use the word virus to mean any contagion propagating in the network. We also use the words graph and network interchangeably.

The chapter is structured as follows: First we introduce the problem in Section 2.1, then we discuss the work surrounding the problem by a literature survey in Section 2.2. After which we discuss the preliminaries for the problem in Section 2.3, followed by a discussion on our problem formulation in Section 2.4 and then proposed approach and algorithms in Section 2.5. Finally we end with extensive experiments in Section 2.6 with discussions and conclusions in Section 2.7 and 2.8.

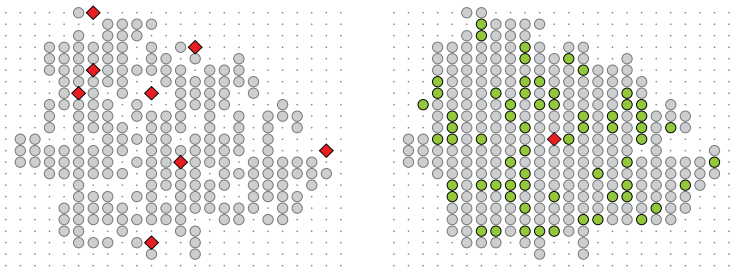
2.1 Introduction

Epidemics are commonplace in graphs. That is, many graph databases store, in one way or another, how information propagates in viral fashion through a graph. Real-world viral infections, such as the flu, are perhaps the most natural example. In these cases real viruses infect people, whom in turn infect other people. The graph in this case is a real social network, of whom interacted with whom. In the situation of an epidemic, institutions such as the Center for Disease Control (CDC) try to map these networks and aim to find out who is infected, in part to discover the sources of the epidemic (the so-called 'patient-zeros'). This is useful in understanding 'what-might-have-happened' (like helping identify critical environmental factors) and taking corrective measures (like sanitation) to prevent

re-occurrence.

Consider also memes in social media; popular phrases or links that are, e.g., picked up and posted on Facebook, or re-tweeted on Twitter, so 'infecting' followers to do the same. From both social science, as well as from advertising points of view it is interesting to see how such an epidemic behaved, which were the starting points, and whom helped spread the epidemic despite not being recorded as such, etc.

However, in reality, neither snapshots nor graphs are noise free; and 'recovering' missing data is an important problem in its own right. There are a number of reasons why we may have missing data in such epidemics/cascades. For example, in epidemiology, surveillance data on who is infected is limited and noisy [Salathé et al., 2012] (as demonstrated by the well-known 'surveillance-pyramid' [Nishiura et al., 2011] where the total detected infections at the top of the pyramid is a fraction of the actual infections at the bottom). In Facebook, most users keep their activity and profiles private. In Twitter, typically only a percentage sample of Tweets can be achieved by the public API. In particular as externals, we seldom have access to the complete cascade, and moreover, if only because of the extreme velocity of social media data, one typically has to resort to analyzing only a sample of the data. Both aspects imply we will have to make do with an incomplete snapshot.



(a) NetSleuth (b) NetFill

Figure 2.1: ***The need for a new approach,* NetFill*:*** *(a) the state of the art, NetSleuth, finds 8 sources (red diamonds). (b) Our method,* NetFill*, finds* both *the right number of sources (one) and recovers the missing nodes with high precision (green).*

In this thesis, we study the problems of finding both the source nodes of an epidemic in a graph given noisy data, and recovering the missing infections. Computationally, finding culprits in even a noise-free snapshot of a noise-free graph is a complex problem [Prakash et al., 2012, Shah and Zaman, 2011]. At the same time, in spite of its importance, missing data in context of cascades has not received much attention (see Sec. 2.2 for a discussion of related work). In this thesis we show that *both* these problems can indeed efficiently be solved *simultaneously*. Figure 2.1 demonstrates how our method, NetFill, recovers missing data, as well as identifies culprits with high precision.

More in particular, we consider finding culprits under the Susceptible-Infected (SI) model, and we allow the given snapshot to include false positives and false negatives; nodes erro-

neously reported as resp. infected and healthy. Our goal is to efficiently and reliably find both the starting points of the epidemic, as well as identifying these input errors; in particular, figuring out who was *truly infected* by the virus, but not reported as such in the snapshot.

Our contributions include:

(a) *Problem Formulation:* We formulate the problem of finding missing infected nodes and concealed culprits in terms of the Minimum Description Length principle.

(b) *Fast Algorithms:* We present fast and effective near-linear time algorithms based on a novel alternating minimization approach to *automatically* recover both missing nodes and seeds. We give solutions for both cases where we do, and where we do not know the sampling rate.

(c) *Extensive Experiments:* We perform extensive experiments using both simulated and *real* cascades on both simulated and real networks. The results show that our algorithms obtain high quality results, outperform the baselines, and consistently recover true missing nodes.

2.2 Related Works

Although diffusion processes have been widely studied, the problem of 'reverse engineering' an epidemic has received much less attention. [Shah and Zaman, 2010, Shah and Zaman, 2011] formalized the notion of *rumor-centrality* for identifying the single source node of an epidemic under the SI model, and showed an optimal algorithm for *d-regular trees*. [Lappas et al., 2010] study the problem of identifying k seed nodes, or *effectors* of a partially activated network, which is assumed to be in steady-state under the IC (Independent-Cascade) model. [Prakash et al., 2012] studied recovering multiple seed nodes under the SI model given a snapshot taken any time during the infection. Here we study the more complex problem of *simultaneously* recovering concealed culprits and missing nodes from a noisy and/or sampled snapshots.

Missing data in networks is an important yet relatively poorly understood problem. A related line of work studies the effect of sampling on measured structural properties [Costenbader and Valente, 2003, Kossinets, 2006, Borgatti et al., 2006] or network construction [Lakhina et al., 2003, Maiya and Berger-Wolf, 2011]. Correcting for the effects of missing data in cascades in general has not seen much attention—the exception is [Sadikov et al., 2011], whom tried to correct for the sampling, yet only in broad statistical terms (like recovering the average size and depth of cascades) assuming a modified new cascade model (k-trees). In contrast, we address the much more general problem of *automatically* directly correcting at a *per-node* level, under a fundamental epidemic model (the SI model).

Another related line of work is learning graphs from sets of observed cascades [G-Rodriguez et al., 2010, Gomez-Rodriguez et al., 2011], though they assume that no nodes are missing nodes in a cascade.

We categorize the rest of the related work mainly into areas dealing with epidemic/cascade-style processes and problems related to them like epidemic thresholds, immunization, and influence maximization. There is a lot of research interest in studying different types of information dissemination processes on large graphs in general, including (a) information cascades [Bikhchandani et al., 1992, Goldenberg et al., 2001], (b) blog propagation [Leskovec et al., 2007b, Gruhl et al., 2004, Kumar et al., 2003] and [Richardson and Domingos, 2002], and (c) viral marketing and product penetration [Leskovec et al., 2006].

Epidemic Thresholds The canonical text-book for epidemiological models like SI is [Anderson and May, 1991]. Much research in virus propagation studied the so-called epidemic threshold, that is, to determine the condition under which an epidemic will not break out [Kephart and White, 1993, Pastor-Santorras and Vespignani, 2001] and [Chakrabarti et al., 2008, Ganesh et al., 2005, Prakash et al., 2011].

Influence Maximization An important problem under the viral marketing setting is the influence maximization problem [Richardson and Domingos, 2002, Kempe et al., 2003, Goyal et al., 2011, Chen et al., 2010, Habiba and Berger-Wolf, 2011]. Another remotely related work is outbreak detection [Leskovec et al., 2007a] in the sense that we aim to select a subset of '*important*' nodes on graphs.

Immunization Another remotely related problem for such propagation processes is immunization - the problem of finding the best nodes for removal to stop an epidemic, with effective immunization strategies for static and dynamic graphs [Hayashi et al., 2003, Tong et al., 2010, Briesemeister et al., 2003, Prakash et al., 2010].

MDL We are not the first to use the Minimum Description Length principle [Grünwald, 2007] for a data mining purpose. [Faloutsos and Megalooikonomou, 2007] argue many data mining problems are related to Kolmogorov Complexity, which means they can be practically solved through compression—examples include clustering [Cilibrasi and Vitányi, 2005], pattern mining [Vreeken et al., 2011], and community detection [Chakrabarti et al., 2004].

In short, to the best of our knowledge, this thesis is the first to deal with finding both missing nodes and concealed culprits in sampled epidemics, in an automatic manner.

Table 2.1: Terms and symbols

$G(V, E)$	complete graph from historical data
β, p	virus infection probability, sampling rate
d_{n_i}	number of infected neighbors of node n
D	set of nodes observed as infected
I	set of nodes our model identifies as infected
C^-, C^+	nodes in I but not in D, and vice-versa, i.e., $D = (I \setminus C^-) \cup C^+$
S	set of seed nodes
R	infection ripple from S to I
$\mathcal{F}_d^i$	frontier set of iteration i of the ripple, the uninfected nodes under attack by d neighbors
$\mathcal{L}(.)$	length in bits

2.3 Preliminaries

In this section we give brief introductions to both the SI process and the MDL principle. We list the most important notations used throughout the chapter in Table 2.1.

2.3.1 The Susceptible-Infected Model

The most basic epidemic model is the so-called 'Susceptible-Infected' (SI) model [Anderson and May, 1991]. Each object/node in the underlying graph is in one of two states: Susceptible (S), or Infected (I). Once infected, a node stays infected forever. At every discrete time-step, each infected node attempts to infect each of its uninfected neighbors independently with probability β, which reflects the strength of the virus.

Note that $1/\beta$ hence defines a natural time-scale; intuitively it is the expected number of time-steps for a successful attack over an edge. As an example, if we assume that the underlying network is a clique of N nodes, under continuous time, the model can be written as: $\frac{dI(t)}{dt} = \beta(N - I(t))I(t)$, where $I(t)$ is the number of infected nodes at time t—the solution is the logistic function and it is invariant to $\beta \times t$.

2.3.2 Minimum Description Length Principle

The Minimum Description Length principle (MDL) [Grünwald, 2007], is a practical version of Kolmogorov Complexity. Both embrace the slogan *Induction by Compression*. For MDL, this can be roughly described as follows.

Given a set of models $\mathcal{M}$, the best model $M \in \mathcal{M}$ is the one that minimizes $\mathcal{L}(M) + \mathcal{L}(\mathcal{D} \mid M)$,

in which $\mathcal{L}(M)$ is the length in bits of the description of M, and $\mathcal{L}(\mathcal{D} \mid M)$ is the length of the data encoded with M.

To use MDL, we have to define our model class $\mathcal{M}$, how a $M \in \mathcal{M}$ describes the data, and how to encode both model and data in bits. To allow for fair comparison between different $M \in \mathcal{M}$, MDL requires the encoding to be lossless. Importantly, in order to identify the best model we are only concerned with *ideal* code lengths, not actual code words.

2.4 Problem Formulation

In this section we discuss the problem setting, and then formalize our objective in terms of the MDL principle.

2.4.1 The Problem: General Terms

We consider an undirected graph $G(V, E)$, with V the set of nodes, and E the set of edges between nodes. In addition we are given a snapshot $D \subset V$ of nodes known to be infected at time t. We allow the snapshot to be incomplete with regard to the true set of infected nodes I^*—that is, there may exist nodes that are infected at time t but which are not in D (e.g., due to false negative tests, sampling errors, etc)—and in addition, we allow D to include nodes infected by an external process, i.e., nodes were infected independent of the graph structure. We assume that the contagion spread according to the SI model with parameter β. Informally, our goal is to correct the infected snapshot, and find the starting-points of the epidemic.

Our overall approach is to use MDL to find the most succinct description of D—of 'what happened'. Loosely speaking, we aim to find the shortest description of D in terms of a set of infection starting points and subsequent infection cascade (still assuming the SI model). The key idea is that describing D will be easier when we 'allow' the cascade to infect true missing nodes and identify these later, than when we force it to not infect these nodes and 'go around' this part of the graph.

To use MDL, we have to define a model class $\mathcal{M}$, how a $M \in \mathcal{M}$ describes the data, as well as how to encode both model and data in bits. Prakash et al. [Prakash et al., 2012] recently showed we can use MDL to identify infection sources given a *noise-free* snapshot. By this (strong) assumption they essentially did not have to consider the complexity of the data under the model: any valid model described the snapshot exactly. In our setting, we do allow noise, and as MDL requires lossless descriptions, we have to carefully formalize an encoding that besides the cost of the model, $\mathcal{L}(M)$, also incorporates the cost of the data given a model, $\mathcal{L}(D \mid M)$.

2.4.2 Our MDL Model Class

We refer to the starting points of an infection as its *seed* nodes. In the SI model, nodes neighboring an infected node are under constant attack. That is, per iteration each infected node has a probability β to successfully attack an uninfected neighbor. We refer to the set of nodes under attack at iteration i as the frontier set F^i. We write F^i_d for the subset of F^i of nodes under attack by d infected neighbors. Note that all nodes in F^i_d have the same probability of getting infected. That is, the probability of a node n getting infected depends only on β and d_{n_i}, the number of infected neighbors in iteration i.

We refer to a cascade of node infections as an infection *ripple*. Basically, a ripple R is a list that, starting from the seed nodes S, per iteration identifies the sets of nodes that were successfully attacked. We do not put any restrictions on the nodes that R may or needs to infect. That is, R may infect any node in V, also those missing from D, and R does not have to infect all of D, allowing for externally infected nodes.

Combined, a tuple (S, R) is a model $M \in \mathcal{M}$ for a given graph G and snapshot D. Together, they identify $I \subset V$ as the infected footprint, the set of nodes infected after having run the ripple from the seed nodes. In the ideal case, I will be equal to the true set of SI infected nodes I^*.

2.4.3 The Cost of the Data

Given a tuple (S, R), describing the data means reconstructing D from I. This means *correcting* I wrt. D, identifying the nodes in I that are not in D, and vice-versa. For the nodes missing from D we write $C^- = I \setminus D$, for those identified as externally infected we have $C^+ = D \setminus I$. It is easy to see that $(I \setminus C^-) \cup C^+ = D$ describes D without loss.

In terms of MDL we have $\mathcal{L}(D \mid S, R) = \mathcal{L}(C^-) + \mathcal{L}(C^+)$. Intuitively, nodes observed as infected but which prove hard (impossible) to reach from the seeds are likely candidates for C^+, whereas those nodes not observed as infected but which strongly simplify reaching the infected footprint are likely candidates for C^-. We will use this intuition in our algorithm.

In practice, there exist cases where we have an expectation on the number of missing nodes, e.g., when sampling D ourselves, as well as cases where such expectation is difficult or impossible to formalize. We consider both settings.

For when we do not have a clear expectation, we do not set any restriction but rather use the intuition that the larger C^- resp. C^+ is, the more this costs. Formally, we have

$$\begin{aligned} \mathcal{L}(C^-) &= \mathcal{L}_{\mathbb{N}}(|C^-| + 1) + \log \binom{|I|}{|C^-|}, \\ \mathcal{L}(C^+) &= \mathcal{L}_{\mathbb{N}}(|C^+| + 1) + \log \binom{|V \setminus I|}{|C^+|} \end{aligned}$$

for identifying C^- and C^+ respectively, where we first transmit the number of nodes, then their ids. For the former we use the MDL optimal code for integers ≥ 1, $\mathcal{L}_{\mathbb{N}}(z) = \log^*(z) + \log c_0$, where $\log^*(z) = \log z + \log\log z + ...$ including only the positive terms and set $c_0 = 2.8654$ to make sure all probabilities sum to 1 [Rissanen, 1983]. We transmit the node ids using a data-to-model code, assuming a canonical order over all ways to select m nodes out of n, encoding all ids together by an index over this order.

Alternatively, and more commonly, we will be provided a sampling rate probability p, which denotes the probability of keeping an infected node—in other words, we expect $p\%$ of the truly infected nodes I^* to be in D. In this paper we assume a uniform sampling rate—a common strategy, e.g., used in the Twitter API. We can also interpret this as a probability $\gamma = (1-p)$ on each node in I^* to not be in D, i.e., to be truly missing. In practice, we do not have access to I^*, yet if we assume I is a good approximation we can use γ as a probability on I to be in C^-. We then have

$$\begin{aligned} \mathcal{L}(C^- \mid \gamma) &= -\log \Pr(|C^-| \mid \gamma) + \log \binom{|I|}{|C^-|} \\ \text{with,} \quad \Pr(|C^-| \mid \gamma) &= \binom{|I|}{|C^-|} \gamma^{|C^-|} (1-\gamma)^{|I|-|C^-|} \quad , \end{aligned}$$

where we encode the size of C^- using an optimal prefix code, and then identify the node ids analogue to above.

2.4.4 The Cost of a Model

Next, we discuss how to encode (S, R). For this, we re-use the encoding for noise-free snapshots [Prakash et al., 2012]. For seeds, we have

$$\mathcal{L}(S) = \mathcal{L}_{\mathbb{N}}(|S| + 1) + \log \binom{|V|}{|S|} \quad ,$$

where we simply encode the number of seeds using $\mathcal{L}_{\mathbb{N}}$ and identify their ids as above.

Encoding the infection ripple is less straightforward.[1] For the encoded length of an SI-model infection ripple R starting from seed nodes S we have

$$\mathcal{L}(R \mid S) = \mathcal{L}_{\mathbb{N}}(T) + \sum_{i}^{T} \mathcal{L}(\mathcal{F}^i) \quad ,$$

[1] Alternative to encoding by a single ripple, in theory one can consider the negative log-likelihood of reaching I from S. This can be approximated by MCMC sampling, but is prohibitively expensive in practice. Computing a good ripple, however, is both cheap and gives good results [Prakash et al., 2012].

where T is the length of the ripple in number of iterations. Per iteration we require $\mathcal{L}(\mathcal{F}^i)$ bits to identify the nodes in $\mathcal{F}^i$ that are successfully attacked. Formally,

$$\mathcal{L}(\mathcal{F}^i) = -\sum_{\mathcal{F}_i^d \in \mathcal{F}^i} \Bigg(\log \Pr(m_d \mid f_d, d) + m_d \log \frac{m_d}{f_d} + (f_d - m_d) \log 1 - \frac{m_d}{f_d} \Bigg)$$

where $f_d = |\mathcal{F}_d^i|$, and m_d is the number of nodes of attack degree d that get infected. As the SI model considers every attack an independent event with probability of success β, we can calculate the probability of seeing m_d nodes out of f_d infected by a Binomial with parameter p_d, with

$$\Pr(m_d \mid f_d, d) = \binom{f_d}{k} p_d^{m_d} (1 - p_d)^{f_d - m_d} \quad .$$

where p_d expresses the independent probability of a node in $\mathcal{F}_d$ being infected, $p_d = 1 - (1 - \beta)^d$.

That is, the value for β determines p_d. Knowing p_d we can encode m_d using an optimal prefix code, the length of which can be calculated by Shannon entropy [Cover and Thomas, 2006]. Then, knowing m_d, we use optimal prefix codes to encode whether each node in $\mathcal{F}_d$ was successfully infected or not.

2.4.5 The Problem: Formally

With the above, we can now formally state the problem.

Minimal Noisy Infection Snapshot Problem *Given a graph $G(V, E)$, the SI model with infection parameter β, a subset of nodes $D \subseteq V$ observed as infected, and optionally a sampling probability p, find the set of missing nodes $C^- \subseteq V \setminus D$, and set of observation errors $C^+ \subseteq D$, a set of seed nodes $S \subseteq V$, and a propagation ripple R that starting from S infects all of $I = (D \cup C^-) \setminus C^+$, such that*

$$\mathcal{L}(D, S, R) = \mathcal{L}(S) + \mathcal{L}(R \mid S) + \mathcal{L}(D \mid S, R)$$

is minimal.

This problem setting explicitly identifies the noise in D. That is, the set of nodes $C^- = I \setminus D$ are the false negatives of D, the nodes that according to the model have been falsely observed as uninfected. The set of nodes $C^+ = D \setminus I$ are the false positives of D, the nodes falsely reported—according to the model—as infected through the observed network.

To find the optimal solution we have to consider an immense search space: any node in V not in D may be a missing infection, and similarly any node in D may be an erroneous

observation. To find the optimum, we would hence have to consider all ripples R for all subsets I of V, starting from any S, in turn all subsets of I. However, there is no trivial structure (e.g., monotonicity) that we can exploit for fast search. In fact, finding only a *single* MLE seed node in a general graph, without missing nodes or false-positives errors, is #P-Complete [Shah and Zaman, 2011] (equivalent to enumerating all the linear extensions of a poset). We hence resort to heuristics.

2.5 Solution and Algorithms

We now describe our proposed approach. Suppose an oracle gives us the true seeds from which the infection started. Loosely speaking, our goal then is to find a footprint I, from which it is easy to reconstruct the observed snapshot D. To make reaching I simpler, we can 'flip' nodes observed uninfected and consider them infected—and vice versa. For C^-, these should be nodes that make reaching I simpler; nodes for which the optimal ripple would otherwise time-and-again spend bits on unsuccessful attacks, as well as nodes that make it easier to reach D. In contrast, C^+ should consist of nodes observed as infected but which are extremely costly to reach from the observed infections (e.g., far-apart disconnected components, that we could only reach by adding many nodes to C^-), or nodes that have so many uninfected neighbors that it is expensive to encode all the unsuccessful infection attempts during the ripple.

2.5.1 Overall Strategy

The above observation suggests a simple strategy: keep track of how many bits the ripple needs to encode the final state of each node, and flip the nodes with the highest cost. That is, keep track of *both* the total cost to keep it uninfected and the effort to reach the node.

Intuitively, the first part seems to be aggregated local costs: nodes for which this is high are likely good candidates for C^-. Indeed, our algorithm quickly identifies all such candidates without having to calculate these individual costs.

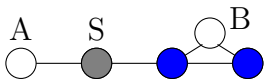
Minimizing the second part of the cost requires calculating the MLE path to *every node* in D, which is infeasible for large graphs. Instead, we note that in most applications, including epidemiology and social media, the false positive rate is very low (i.e., the probability that an *observed* 'infection' is wrong). This allows the leap of faith that all connected components in D of at least 2 nodes are not purely due to chance. That is, they either should have their own seed node, or, be connected to another component with a seed node—e.g. by nodes in C^-. Under the same assumption, we axiomatically identify C^+ as the (rare) disconnected singleton nodes of $D \cup C^-$.

To summarize, with C^+ defined as above, our task is to find a set of missing nodes C^-, a

seed set S and a ripple R such that under the SI model if we start from the seed set S the infection ripple R spreads to all nodes in D and C^-, and it is the cheapest setting according to our MDL score.

2.5.2 NetFill—Main Idea

Assume we are given a budget of at maximum k missing nodes $|C^-| \leq k$: how can we find the best nodes? Following from the discussion above, good candidates would be nodes which have a high cost of attempted infections. Intuitively, the larger the number of infected neighbors—i.e., the infected degree d_{n_i}—of a healthy node n, the larger the number of infection attempts, and hence the higher the cost we will have to pay to keep the node *uninfected*. Hence it seems sensible to choose the k nodes with highest d_{n_i} from the set $V \setminus D$. There are, however, two clear disadvantages to this approach: (a) we *do not* know k, and (b) we ignore both the seeds and the ripple of the infection. For example, consider the following:



Here if node S was the seed then intuitively node A should have been infected, whereas by using infected degrees, we will get B as the top-most candidate. Preliminary experiments showed that in practice this strategy indeed consistently outputs the wrong number and identity of seeds—even when given the true k as input.

To solve these issues, we follow a different approach. It is easy to see that the choice of C^- will affect the identity of the seed set S. Additionally, the above example demonstrates that the choice of S also affects the choice of the missing node set—this is because the seeds determine what is the best possible ripple. Consequently, cheaper ripples will require fewer missing nodes to be 'filled-in'. Following this observation, we take an EM-style alternating-minimization approach:

(a) find the best seeds for a given set of missing nodes,

(b) find the missing nodes, given a set of seed nodes.

Given an initialization, we alternate these steps until convergence. Task (a) is similar to finding seeds under perfect data.contrast, Task (b) requires us to efficiently find missing node sets given seeds.

2.5.3 NetFill—Details

We shall next discuss how we solve each of the two tasks above, and then how to combine them as NetFill.

2.5.3.1 Task (a): Finding Seeds given Missing Nodes

In this task, we are given a set of missing nodes, and need to find the best seeds under this perfect, noise-free, information. In principle we can use any seed-finding algorithm for this task. Under perfect information, however, our MDL score reduces to that of NetSleuth [Prakash et al., 2012], which is a solution towards finding a good set of seed nodes S given an accurate D: exactly Task (a)'s assumption. Hence, we instantiate FINDSEEDS$(D, G(V, E), C^-)$ using $NetSleuth(G, D \cup C^-)$. Note that the seeds might themselves be 'concealed' w.r.t. D, as they may be selected from the nodes in C^-.

2.5.3.2 Task (b): Finding Missing Nodes given Seeds

In this step, we assume that the seed set S along with the missing nodes from the previous iteration, C^-_{prev}, are given, and our task is to find the best set of missing nodes C^-. With S given and assumed accurate, the naive approach is to list all possible C^- and let MDL decide which is the best solution—sadly, this approach is computationally infeasible. Instead we propose to find C^- incrementally and greedily; at every step we find the next best node n^* to add to C^-.

How should we select this 'next best node' n? From our MDL score we know that adding a node will change both the cost of the missing nodes, $\mathcal{L}(C^-)$, as well as the cost $\mathcal{L}(R)$ of the ripple from S for reaching the infected set. By the connection between encodings and distributions [Grünwald, 2007], we can interpret $\mathcal{L}(R)$ as the negative log-likelihood of the ripple. Our strategy is hence to choose that node n which maximally increases the likelihood $\mathfrak{L}(\cdot)$ that S is indeed the seed set for the resulting infections $(D \cup n)$. While TOP-k-DNI only takes the cost d_{n_i} of *not* infecting a node into account, here the likelihood measures the *total effect* of flipping a node; that is, we implicitly consider the cost of infecting n, its neighbors, neighbors-of-neighbors and so on till we reach D. However, computing the exact total likelihood is computationally very expensive, and hence we use the total expected error $\mathfrak{R}$ instead, i.e., the empirical risk between the actual state and expected state of the snapshot if the seed set was S. Formally,

$$\mathfrak{R}(D \mid S) = \sum_{i \in V} (\mathbb{1}_{i \in D} - \mathbb{E}[\text{state of } i | S]) \quad , \tag{2.1}$$

where $\mathbb{1}_{i \in D} = 1$ if $i \in D$ (0 otherwise). We can then use the MDL score of $\mathcal{L}(C^-) + \mathcal{L}(R)$ to see if adding n reduced the total bits (as S is constant in this Task, so is $\mathcal{L}(S)$).

Single seed: Assume for now we have one seed $S = \{s\}$; later we discuss how to extend this to multiple seeds. Start with $C^- = \emptyset$. From above, the best single node to add to C^- minimizes total expected error $\mathfrak{R}(D \cup n \mid s)$. Equivalently,

$$n^* = \arg\max_n \left[\mathfrak{R}(D \mid s) - \mathfrak{R}(D \cup n \mid s)\right] \quad . \tag{2.2}$$

As s was computed in Task (a) via NetSleuth, we use Lemma 3 from [Prakash et al., 2012] for $\mathbb{E}[\cdot]$ into Equation 2.1 and obtain

$$\mathfrak{R}(D \mid s) \approx \sum_{i \in D} (1 - u_1(i)u_1(s)) \quad ,$$

where u_1 is the smallest eigenvector of $L_{D \cup C^-_{\text{prev}}}$—the *submatrix* of the graph laplacian $L = \text{Deg} - G$ corresponding to the 'infected set' on which s was computed, i.e. $D \cup C^-_{\text{prev}}$. In other words, we take the subset of rows and columns from L corresponding to the nodes in $D \cup C^-_{\text{prev}}$. Note that the sum is only over the *observed* infections D while the expected states are calculated using $u_1(\cdot)$ based on s (which was based on C^-_{prev}). So the best single node to add to C^- can be written as:

$$n^* \quad = \quad \arg\max_n \left[\sum_{i \in D \cup n} \tilde{u_1}(i)\tilde{u_1}(s) - \sum_{i \in D} u_1(i)u_1(s) \right] \tag{2.3}$$

where $\tilde{u_1}$ is the smallest eigenvector of the new laplacian submatrix we obtain after adding the node n. Thus we need to compute the *change* of the smallest eigenvector from u_1 to $\tilde{u_1}$ when the laplacian submatrix $L_{(\cdot)}$ changes after a node n is added to the infected set. Again, directly computing this change for each node is expensive as this involves $O(N)$ eigenvalue computations.

How to do this faster? We propose to use matrix perturbation theory to compute this change approximately. Lemma 1 together with the fact that many real graphs have large eigen-gap gives us a way of quickly approximating and finding the node n^*. We use the following recent result (Theorem 1 in Wu et al [Wu et al., 2011]) for proving the lemma. Let A be a $n \times n$ matrix, and we apply a symmetric 'perturbation' E (a symmetric matrix) to it to get $\tilde{A} = A + E$. Let λ_i and x_i be the i-th largest eigenvalue and eigenvector of A. Similarly, let $\tilde{\lambda}_i$ and $\tilde{x}_i$ denote the eigenvalue and eigenvector for $\tilde{A}$. Let $U = (x_1, x_2, \ldots, x_n)$ and $\theta_{ij} = x_i^T E x_j$. Then:

Theorem 1 (Spectral Perturbation [Wu et al., 2011]) *Assume the following conditions hold: 1)* $\delta = |\lambda_i - \lambda_{i+2}| - ||x_i^T E x_i||_2 - ||U^T E U||_2 > 0$*; 2)* $\gamma = ||U^T E U||_2 \leq \delta/2$*; and 3)* $|\lambda_i| \gg |\lambda_j|$ *for any* $i = 1 \ldots k$ *and* $j = k+1, \ldots, n$*. Then,*

$$\tilde{x}_i \approx x_i + \sum_{j=1, j \neq i}^{k} \frac{\theta_{ji}}{\lambda_i - \lambda_j} x_j + \frac{1}{\lambda_i} E x_i \tag{2.4}$$

Theorem 1 is an extension to the classical result by Stewart and Sun [Stewart and Sun, 1990] to when the first k eigenvalues are significantly greater than the rest—as is typical in real graphs.

Lemma 1 *Given a seed node* s*, and* $\lambda_1 - \lambda_2 > 3$ *for* $L_{D \cup C^-_{\text{prev}}}$*, with* $nb(n)$ *the neighbors of a node* n*, under spectral perturbation we have* $n^* \approx \arg\max_n \sum_{i \in nb(n)} u_1(i)$.

Proof Let $Y = D \cup C^{-}_{\text{prev}}$. It was proved in [Prakash et al., 2012] (Lemma 2) that the eigenvector corresponding to the largest eigenvalue of $I - \frac{1}{\sigma}L_Y$ is also the smallest eigenvector u_1 of L_Y. Here, $\sigma = d_{max} + 1$ and d_{max} is the maximum degree in the graph. Consider the matrix $L_X = I - \frac{1}{\sigma}L_Y$ (we augment it with a zero column and zero row—note that this will not change any of the eigenvectors and eigenvalues). Now we want to see how its eigenvectors change when we add a node n to it. WLOG, here adding a node n to the infected set results in a single row and column change to the last row and column in matrix L_X (this column/row was originally all zero). Let the new matrix be $\tilde{L}_X$. Hence $\tilde{L}_X = L_X + E$, where E can be written as: $\begin{pmatrix} \vec{0}_{n-1,n-1)} & z/\sigma \\ z^T/\sigma & d_n/\sigma \end{pmatrix}$ where z is a vector for node n's connectivity (if n is connected to i, then $z(i) = 1$, else $z(i) = 0$; so that $|z| = d_{n_i}$, the *infected* degree of node n). d_n denotes the total degree of node n in the underlying network. Let us write the eigenvectors x_i of L_X as $\begin{pmatrix} v_i \\ 0 \end{pmatrix}$ since L_X is augmented with a zero row and zero column. As L_X and E are symmetric, $|E| < 1$ and $\delta > 3 * |E|$ [Wu et al., 2011], we can apply Theorem 1.

For our problem the second term in the Equation 2.4 is:

$$\begin{aligned}
\sum_{j=2}^{k} \frac{\theta_{j1}}{\lambda_1 - \lambda_j} x_j &= \sum_{j=2}^{k} \frac{x_j^T E x_1}{\lambda_1 - \lambda_j} x_j \\
&= \sum_{j=2}^{k} \frac{[v_j^T 0] E \begin{pmatrix} v_1 \\ 0 \end{pmatrix} \begin{pmatrix} v_j \\ 0 \end{pmatrix}}{\lambda_1 - \lambda_j} \\
&= \sum_{j=2}^{k} \frac{[v_j^T 0] [\vec{0}_{1,n-1} z^T v_1] \begin{pmatrix} v_j \\ 0 \end{pmatrix}}{\lambda_1 - \lambda_j} x_j \\
&= \sum_{j=2}^{k} \frac{[v_j^T 0] \ \vec{0}_{n,1}}{\lambda_1 - \lambda_j} x_j = 0
\end{aligned}$$

The third part of Equation 2.4 for our problem becomes:

$$\begin{aligned}
\frac{1}{\lambda_1} E x_1 &= \frac{1}{\lambda_1} \begin{pmatrix} \vec{0}_{n-1,n-1} & z/\sigma \\ z^T/\sigma & d_n/\sigma \end{pmatrix} \begin{pmatrix} v_1 \\ 0 \end{pmatrix} \\
&= \frac{1}{\lambda_1} \begin{pmatrix} \vec{0}_{n-1,1} \\ z^T/\sigma v_1 \end{pmatrix}
\end{aligned}$$

Thus

$$\tilde{x_1} - x_1 = \frac{1}{\lambda_i} \begin{pmatrix} \vec{0}_{n-1,1} \\ z^T/\sigma v_1 \end{pmatrix}$$

For our problem we have $x_1 = \begin{pmatrix} v_1 \\ 0 \end{pmatrix}$, $u_1 = v_1$ and $\tilde{x_1} = \tilde{u_1}$. So the above equation transforms to

$$\tilde{u_1} - \begin{pmatrix} u_1 \\ 0 \end{pmatrix} = \frac{1}{\lambda_i} \begin{pmatrix} \vec{0}_{n-1,1} \\ z^T/\sigma u_1 \end{pmatrix} \tag{2.5}$$

Note that the change in the eigenvector is only to the last component. The node n^* that minimizes the risk(as defined in section 5.4.2) can also be written as

$$\begin{aligned} n^* &= \arg\max_n \left[\mathfrak{R}(D \mid s) - \mathfrak{R}(D \cup n \mid s)\right] \\ &= \arg\max_n \left[\sum_{i \in D \cup n} \tilde{u_1}(i)\tilde{u_1}(s) - \sum_{i \in D} u_1(i)u_1(s)\right] \\ &= \arg\max_n \sum_{i \in D \cup n} \Big(\tilde{u_1}(s)(\tilde{u_1}(i) - u_1(i)) \\ &\qquad\qquad - u_1(i)(u_1(s) - \tilde{u_1}(s))\Big) \end{aligned} \tag{2.6}$$

Using the result obtained from Equation 2 we have

$$\begin{aligned} n^* &\approx \arg\max_n \ \tilde{u_1}(s) z^T u_1 \\ &= \arg\max_n \ z^T u_1 \\ &= \arg\max_n \sum_{i \in n_b(n)} u_1(i) \quad . \ \blacksquare \end{aligned}$$

Loosely speaking, u_1 measures the closeness of nodes in the infected graph to seed s. In particular, s has the largest value of $u_1(\cdot)$ and hence minimizes $\mathfrak{R}(\cdot)$. Intuitively, from Lemma 1, $\mathcal{Z}_n = \sum_{i \in nb(n)} u_1(i)$ hence measures how *connected* a node n is to *centrally located infected nodes w.r.t. s* in D. This is desirable, as it immediately captures our intuition that the missing nodes should depend on the seed *as well as* the structure. It is important to note that while choosing the new C^- we have to ignore the effect of the old C^-_{prev}; we need to find nodes C^- based directly on the seed s; *not* the missing-node set based on which s was itself computed. So before computing $\mathcal{Z}_n$, we set $\forall\ i \in C^-_{\text{prev}}\ u_1(i) = 0$, which ensures that z-scores are computed based only on the observed infected set and seed s. Though, nodes in C^-_{prev} can re-appear in C^- as they can still have non-zero $\mathcal{Z}$s.

Multiple seeds: The extension of the above to multiple seeds is not straightforward. Consider the case in which we have two seeds s_1 and s_2. Naively applied, our $\mathcal{Z}$-score will choose only those nodes that are 'close' to seed s_1—e.g., in the grid network of Fig. 2.4(e) we would choose only nodes close to the seed in the left-blob. How to instead choose nodes that are close to either left and right blobs? To boost diversity, we adopt 'exoneration': we set the first seed s_1 as *un-infected* (and hence 'exonerate' the nodes close to it) and recompute the smallest eigenvector of the laplacian submatrix defined by $D \cup C^-_{\text{prev}} \setminus s_1$ (call this vector

u_2). Then the $\mathcal{Z}$ based on u_2 will measure the appropriateness of adding a node based on its centrality w.r.t. to seed s_2. In general, for a l seed problem we will have u_1, u_2,...u_l. For these l eigenvectors we will have l $\mathcal{Z}_n$'s for every node in $V \setminus D$. For node n, we define the consolidated $\mathcal{Z}_n^S = \max\{\mathcal{Z}_n^1, \mathcal{Z}_n^2, ...\mathcal{Z}_n^l\}$ (as before, we also set $\forall\ i \in C^-_{\text{prev}},\ \forall\ l\ u_l(i) = 0$). Thus the best node to be added in the case of multiple seeds is a node which is very central and close to at least one of the seeds i.e. which has the maximum value of $\mathcal{Z}_n^S$.

How many nodes to add? Finally, we just add the top scoring nodes according to the $\mathcal{Z}_n^S$ scores to C^- until MDL tells us to stop. Note that we don't need to re-compute $\mathcal{Z}_n^S$ after every addition, as S is assumed correct in this Task. We give the pseudocode as Algorithm 1.

Algorithm 1 FINDMISSING: Finds the set of missing nodes given a set of seed nodes

Input: Data D, graph $G(V, E)$, seed set S and the old missing set C^-_{prev}
Output: Missing nodes C^-

Let $S = \{\emptyset, s_1, s_2...s_{l-1}\}$
$\mathcal{Z}_n^S =$ FINDNODESCORES$(G,\ D,\ C^-_{\text{prev}},\ S)$
$C^- = \emptyset$ and $i = 0$
while $\mathcal{L}(S, D, R, C^-)$ decreases **do**
  $C^- = C^- \cup \underset{n \in V \setminus D \cup C^-}{\arg\max}\ \mathcal{Z}_n^S$
  $i = i + 1$
end while
return C^-

Algorithm 2 Function FINDNODESCORES $(G, D, C^-_{\text{prev}}, S)$

for $i = 1$ **to** l **do**
  $G_i = D \cup C^-_{\text{prev}} \setminus \cup_{j=0}^{i-1}\{s_j\}$ = the infected subgraph
  u_i = smallest eigenvector of G_i
  $u_i(l) = 0\ \ \forall\ l \in C^-_{\text{prev}}$
  for $n \in V \setminus D$ **do**
    $\mathcal{Z}_n^i = z^T u_i$
  end for
end for
for node $n \in V \setminus D$ **do**
  $\mathcal{Z}_n^S = \max \mathcal{Z}_n^1, \mathcal{Z}_n^2, ...\mathcal{Z}_n^k$
end for
return $\mathcal{Z}_n^S$

Algorithm 3 NetFill

Input: Data D, graph G, and infectivity β
Output: Missing nodes C^-, ripple R and seed set S
1: $C^- =$ frontier-set of D in G
2: $\{S, R\} =$ findSeeds(D, C^-, G)
3: **while** $\mathcal{L}(S, D, R, C^-)$ decreases **do**
4: $\quad \{S, R\} =$ findSeeds(D, C^-, G)
5: $\quad C^-_{\text{prev}} = C^-$
6: $\quad C^- =$ findMissing$(G, S, D, C^-_{\text{prev}})$
7: **end while**
8: **return** S, R, C^-

2.5.3.3 The Complete Algorithm

Given the two procedures above for Task (a) and Task (b), we can now combine them into the NetFill algorithm. We give the pseudo-code as Algorithm 3. First we need to initialize C^- for the procedure. In principle any heuristic can be used, here we choose to use the frontier-set; we set the initial C^- as the set of all those uninfected nodes which are connected to at least one infected node (the frontier set). After initialization, we calculate the seeds S for this C^- and D. We then use the alternating approach by iteratively optimizing C^- and S, until the stopping condition. As discussed at the start of this section, C^+ is defined as the disconnected *singleton* nodes in $D \cup C^-$. We stop when the MDL cost ceases to decrease. One detail is that we need to calculate the ripple R when calculating the MDL score. Following [Prakash et al., 2012], we greedily maximize the likelihood of the ripple by iteratively infecting the most likely number of nodes, which is easily computed based on the mode of Bernoulli trials.

Due to its complex nature, $\mathcal{L}(S, D, R, C^-)$ is not a pure convex function—however, in practice it does show a convex like structure. So if we would add one node at a time in Algorithm 1 we might get stuck at a local minima. To be more robust, we do a batched-addition instead of just one node. Experiments show a value of $\sim$10 works well.

Complexity: In Algorithm 3, as NetSleuth is linear, lines 2 and 4 each take $\mathrm{O}(l \times (D + E))$ time, while it takes $\mathrm{O}(D + E)$ time to compute the $\mathcal{L}(\cdot)$ (line 3), where l is the number of seeds. Using the $O(E)$ Lanczos method for eigenvalue computations on sparse graphs and using max-heap, line 6 takes $\mathrm{O}(l \times (E+V-D) + k \times \log(V - D))$, where k is the no. of missing nodes. Combined, the complexity of NetFill is hence $\mathrm{O}(j \times (l \times (E+V) + k \times \log(V - D)))$, where j is the number of iterations the NetFill takes to converge. In our experiments we found $j \approx 3$, while l and k are $\ll D$. So, in practice, NetFill is sub-quadratic (and near-linear in many cases), which makes it suitable for large graphs.

Table 2.2: General statistics of the graphs

Dataset	$\|V\|$	$\|E\|$	$avg(degree)$
Grid	3600	7065	3.93
AS-Oregon	10670	22002	4.12
MemeTracker (HL)	851	4781	11.23
Flixster	67478	427744	12.6780

2.6 Experiments

In this section we empirically evaluate our methods. We make our code available for research purposes.[2]

2.6.1 Experimental Setup

We give more details of our experimental setup and some additional results here. As mentioned before, we implemented our algorithm in Matlab and ran our experiments on a 4 Xeon E7-4850 CPU with 512GB memory.

2.6.1.1 Data

***Grid*:** First off, we use a synthetic graph because this allows for evaluation in a controlled environment, as well easy visualization. To this end we construct a synthetic 2-dimension 60x60 grid network, in which all interior nodes have degree 4.

***AS-Oregon*:** The Oregon AS router graph represents the internet connections of routers in the University of Oregon network, and is divided into sub-graphs called Autonomous Systems (AS). We use the AS Oregon 1 graph, which consists of traffic data in that AS from March 31 2001 till May 26 2001. *AS-Oregon* has been shown to have a power-law distribution for the degree of the nodes [Faloutsos et al., 1999], which makes it a natural exemplar for many real world applications, such as biological as well as social networks.[3]

***MemeTracker*:** The *MemeTracker* dataset is not a graph itself, but contains mentions of over 200 million popular phrases, memes, in 96 million blog-posts and news media articles from August 2008 to April 2009. It has been used in various information propagation studies [Leskovec et al., 2009, G-Rodriguez et al., 2010]. There are different ways of learning the historical graph from this data and unlike for *Grid* and *AS-Oregon*, for this graph we

[2] `http://people.cs.vt.edu/shashi/hidden-hazards/`

[3] `http://topology.eecs.umich.edu/data.html`.

do not simulate the SI model: the data itself provides information on the 'viral' spread.[4]

***Flixster*:**[5] The *Flixster* dataset contains a graph of user id's connected using the friendship relationship. The graph is undirected and unweighted. The dataset also contains the information of the movies that a user has reviewed on the social media platform. With this information we can construct the data D, where a user id is infected if he reviews a movie.

2.6.1.2 Baselines

Although we are not aware of a direct competitor method, we compared NetFill against some intuitive baselines: Frontier, Simulation.

Frontier: Given an infected set D and the graph $G(V, E)$, the Frontier Set is defined as the set of nodes that have at least 1 infected neighbor. The Frontier baseline just returns the Frontier Set of the given infected graph. This would be a good candidate for a baseline as these nodes are connected to infected nodes and are next in line to be infected if we had let the SI process to continue.

Simulation $(\theta, Seeds)$: Another intuitive heuristic is to just simulate the SI process till we infect the observed snapshot, and consider the extra infected nodes as C^-. It takes two parameters $Seeds$ the seeds from which the simulation should begin and θ the threshold of nodes the Simulation baseline should infect. As the real seeds are not given to us for simulation, we need to estimate them. A natural choice is to estimate seeds from the given snapshot—but note that (a) the actual seeds may have been sampled off and (b) as the observed infections may contain disconnected components, this can result in a large number of unnecessary nodes during the simulation. To avoid these scenarios we used the $Seeds$ we got from NetSleuth on the infected subgraph and the frontier set. Secondly, we simulated the SI model from these seeds till we infect $\theta = 95\%$ of the nodes in the infected set D. As the process is stochastic, we chose 95% and not 100% (so that the baseline does not keep waiting to infect one unlucky node in D).

2.6.1.3 Evaluation—Subtle Issues

For Missing Nodes C^-: Evaluating the quality recovered missing nodes is more subtle than it seems at first glance. Recall—how many true missing nodes are recovered—for instance, does not suffice as it prefers large result sets. Often, the opposite is favorable as we do not want nodes frivolously reported as missing: the true negative rate is important. We hence use two metrics: (a) precision, how many recovered nodes are truly missing, and (b) the Matthews Correlation Coefficient (MCC) [Matthews, 1975] which unlike traditional

[4] `http://snap.stanford.edu/netinf/`

[5] `http://www.cs.sfu.ca/~sja25/personal/datasets/`

measures like precision, recall, or Jaccard considers the number of true positives (tp), true negatives (tn), as well as the false positives (fp) and false negatives (fn). A coefficient of $+1$ represents a perfect prediction, 0 no better than random and -1 indicates total disagreement between prediction and observation. Formally $MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}}$, with a score defined as 0 if any term in the denominator is 0.

For Seed Nodes S: Evaluating the quality of a set of seed nodes S and a ripple R is also more involved than seems at first glance. Just as in the case of noise-free snapshots [Prakash et al., 2012], the subtle issue is that the SI process is stochastic, and hence the recovered seeds may have a *better* score than the actual ones, for the same reason that the sample mean of a Gaussian minimizes least square error, while the theoretical mean does not. Additionally it is intractable to compute the exact likelihood of a snapshot for any non-trivial experiment. Hence, we measure the quality of the recovered seeds and ripple in terms of MDL; following [Prakash et al., 2012] we compare the total encoded length of the discovered solution to the ground truth, $Q = \frac{\mathcal{L}_{\text{alg}}(\cdot)}{\mathcal{L}_{\text{TRUE}}(\cdot)}$. The closer to 1, the better.

2.6.2 Performance on Synthetic Data

As it allows straightforward visualization, we first consider multiple scenarios using the synthetic *Grid* data.

***Grid-Con* and *Grid-Disc*:** We simulate two scenarios on *Grid* using the SI process, $\beta = 0.1$: (a) n seeds to produce n distinct yet connected blobs (*Grid-Con*); and (b) n seeds to produce n disconnected blobs (*Grid-Disc*). After running the process to infect a total of in between 450 and 650 nodes, we sample to get the input using $p = 0.9$. We here report results for $n = 2$, noting these are representative for larger seed sets.

With regards to C^-, Figure 2.4 shows that all methods show good performance in returning true missing nodes (green). Whereas NetFill matches human intuition, Frontier and Simulation, however, return overly many false positives (cyan). We plot the precision and *MCC* in Figure 2.2 (left). This figure shows that NetFill performs best for recovering missing nodes. The baselines Simulation and Frontier choose overly many nodes. Hence, their predictions are no better than random, and we see *MCC* scores closer to zero.

Next, we investigate the near-convexity of $\mathcal{L}$, which decides how many missing nodes NetFill chooses. Fig. 2.3 shows that in practice the score is close to convex for k, with minima close to the ground truth at resp. $k = 90$ for *Grid-Con* and $k = 20$ for each of the components of *Grid-Disc*.

With regards to seed nodes, in Figure 2.4 we see that both NetFill and Frontier discover good candidates (black)—note that Simulation requires the true seed set as input parameter. To evaluate the quality of the ripple R and the seed set S, we consider the Q scores in Fig. 2.2. NetFill closely approximates the ideal—its solution requires as few bits as are

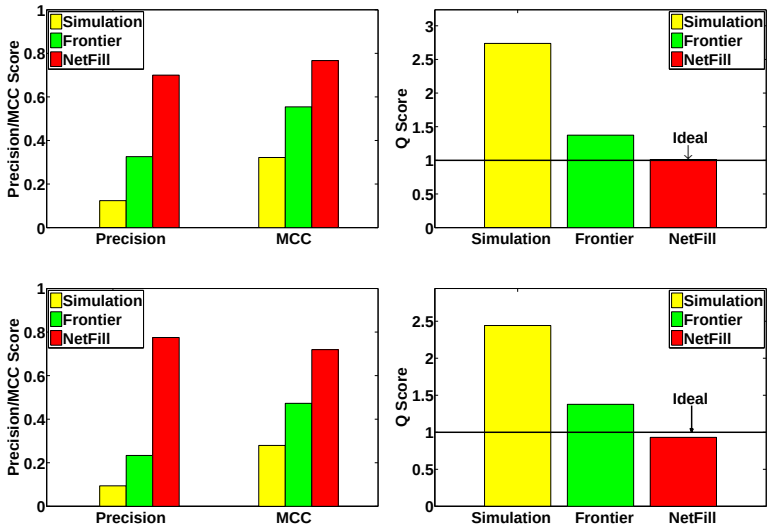


Figure 2.2: ***Performance on Simulated Data:*** NetFill *performs well. Precision and MCC (left, higher is better), and Q (right,* 1 *is ideal) for Grid-Con (top) and Grid-Disc (bottom)*

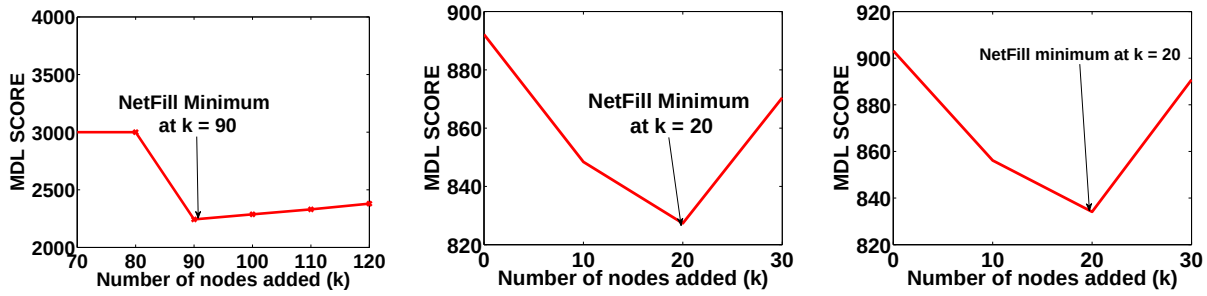
needed for the ground truth—while many more bits are needed to describe the solutions of Simulation and Frontier. For *Grid-Disc* NetFill even finds a solution that is more simple to describe (has higher likelihood) than the ground truth.

2.6.3 Performance on Real Graph and Simulated Cascades

Next, we evaluate performance of NetFill on a graph with power-law degree distribution, but keep control over the infection model. More in particular, we run SI multiple times on the *AS-Oregon* graph using a single random seed of medium to high degree. We vary the number of infected nodes from 700–1500, choose $\beta = 0.1$ and sampling using $p = 0.9$.

For the missing nodes, looking at the results in Fig. 2.5 (a) we see that NetFill performs well in terms of both precision and *MCC* scores, while the baselines return so many false positives that their precision and *MCC* scores are low.

For the culprits, Figure 2.5 (b) shows that NetFill scores well too. In fact, averaged over all simulations NetFill has a Q of 1.03—close to the ideal, implying that its seed sets S and ripple R are of high quality. Simulation and Frontier perform rather poorly with average Q scores of 7.45 and 4.57.



(a) (b) (c)

Figure 2.3: ***Using MDL for finding number of missing nodes:*** *Our score is near-convex, and identifies the correct size of $|C^-|$: MDL scores for Grid-Con (a) and the left (b) and right (c) component of Grid-Disc.*

2.6.4 Performance on Real Graph and Real Cascades

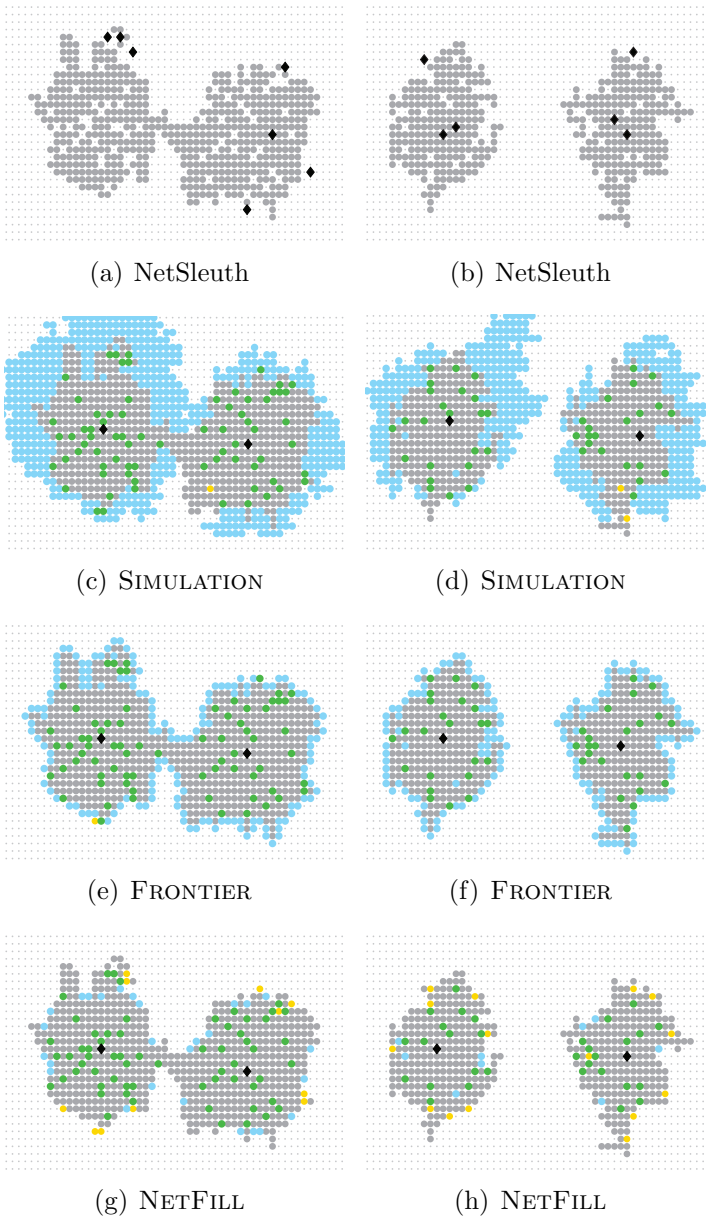
Finally, we evaluate NetFill on real graphs and real cascades. We consider the *MemeTracker* and *Flixster* datasets.

***MemeTracker*:** Here the data (or cascades) defines the infected set D for us, that is we *do not* simulate the SI model to construct D. There exist multiple ways to extract the cascades as well as learning the graph. We consider two sets of cascades, the Memetracker (MT) methodology of phrase matching, and cascades based on explicit hyperlinks (HL) [Leskovec et al., 2009]. For the graph, we consider both the HL and MT networks as learnt by NetInf [G-Rodriguez et al., 2010]. We hence consider four scenarios: *HL-HL*, *HL-MT*, *MT-HL* and *MT-HL*, resp. denoting the network and infected set.

We select missing nodes using a sampling rate of $p = 0.7$, and choose $\beta = 0.1$—notably finding similar results for different values. We use two high-volume memes that were popular in 2008: "Lipstick on a pig" and "The state of the economy". It is important to emphasize here that SI is an abstraction; although the network here was learnt using a SI-inspired model [G-Rodriguez et al., 2010], the actual spread of information in the data may not precisely match our assumptions. Moreover the network extracted here using machine-learning algorithms is itself noisy. In spite of all this, NetFill discover interesting results.

For conciseness we only report results for *HL-MT*, noting these are representative for the other combinations. With respect to missing nodes, NetFill outperforms the baselines as seen in Fig. 2.5 (c). Just as for *AS-Oregon*, the precision and *MCC* are very low for Frontier and Simulation as they select almost the entire graph as C^-. NetFill's precision is more than 5× better than the baselines. Further with regards to the culprits, Fig. 2.5 (d) shows the solution discovered by NetFill has a Q score only a fraction higher than the ground truth.

Truly Missing Nodes: With the rationale that the *MemeTracker* dataset is itself a sample



(a) NetSleuth (b) NetSleuth

(c) Simulation (d) Simulation

(e) Frontier (f) Frontier

(g) NetFill (h) NetFill

Figure 2.4: How well do we perform on Grid? Performance of various methods on Grid-Con (left) and Grid-Disc (right). NetFill *performs best in identifying the missing nodes (green) and finding the seeds (black). Grey nodes are infected, false positives are yellow, and false negatives are cyan. Best viewed in color.*

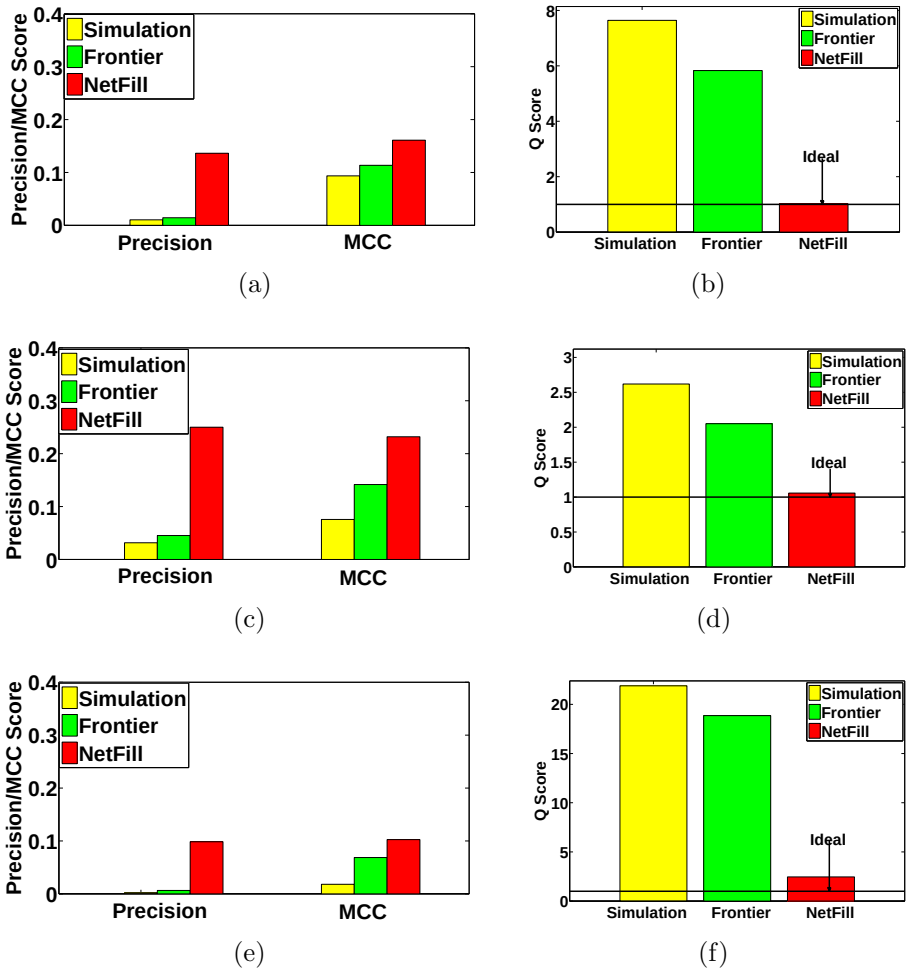


Figure 2.5: ***Performance of Methods on Real Data:*** NetFill *performs well on real data. Performance on AS-Oregon (top) and MemeTracker HL-MT (middle) and Flixster (bottom).* NetFill *has best precision, MCC (left, higher is better), and Q scores (right,* 1 *is ideal).*

of the true cascades in the web, we apply NetFill on the *complete* infected set to discover nodes that were missed when collecting the data. We used *HL-MT* using an expected sampling rate of $p = 0.9$.

At this sampling rate NetFill identifies 22 nodes (websites) as missing, including 'nbcbayarea.com' and 'chicagotribune.com'. By checking their archives we could verify that 6 sites indeed contain the meme "The state of the economy", whereas 2 others discuss politics and economic situation in USA in general. For the remaining websites, it was not possible to verify with certainty if they used the meme or not—some do not offer searchable archives, others are simply not online anymore. This both shows that our method works in practice, as well as that it can be used to improve the quality of cascade datasets, even under our basic assumptions.

***Flixster*:** Last, we consider the *Flixster* dataset by which we evaluate how well NetFill does when the data does *not* follow our assumptions. That is, unlike for *MemeTracker*, for *Flixster* is unclear whether SI is a meaningful model—it is interesting to see whether NetFill still outperforms the baselines. In *Flixster*, the infected set D is a group of people who rated a certain movie, the undirected graph constructed from the friend relationship. We consider movies of medium volume infection, $|D| \approx 3000$. We used a sampling rate of $p = 0.9$. The edge weights were computed using the methodology of [Goyal et al., 2010], and we set the infection probability β as the mean edge weight.

We show the results for a single movie in Figure 2.5 (e) — the results were similar for all applicable movies. As above, NetFill outperforms the baselines in precision, MCC, as well as Q-score with a wide margin—the difference being dramatic for the latter. These scores also show that NetFill is conservative when the data does not follow the model, which prevents overfitting and allowing it to find relatively simple descriptions of what happened.

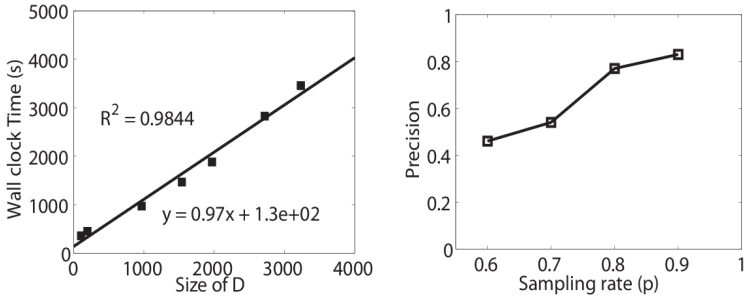
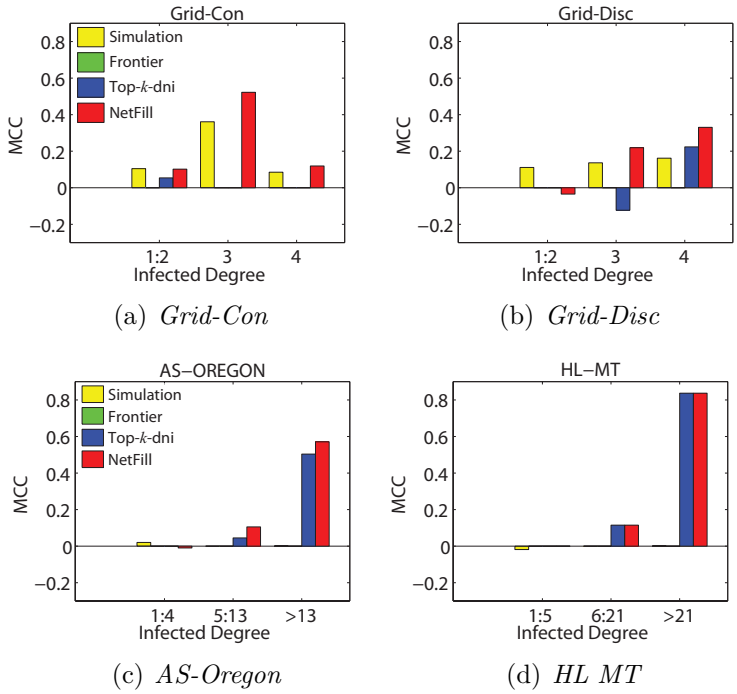


Figure 2.6: ***Scalability and Robustness of* NetFill:***Scalability (left) and Precision (right).* NetFill *is near-linear: Run times for infected footprints in AS-Oregon.* NetFill *is robust: Precision decays gradually with sampling rate on Grid.*

2.6.5 What if the Number of Missing Nodes are Known?

Usually it is difficult to predict the number of missing nodes accurately without the help of a domain expert. Hence we need methods that can automatically find the number of missing nodes as well as their identity. In the case we are given the number of missing nodes k, we can use a heuristic Top-k-DNI to find the identity of the missing nodes.

We hinted in Section 2.5 that we could select nodes with high infected neighbors as our missing nodes C^-. The infected degree (d_{n_i}) of an uninfected node is defined as the number of infected neighbors. The idea behind the approach Top-k-DNI is that nodes with a high infected degree have had more chances to be infected by their infected neighbors. In other words the probability that they will be infected in the very next time step is very high. *If we are given* $C^- = k$, that is the number of missing nodes, we can simply select the top k nodes with the highest infected degree. Although this method seems to perform well insofar as finding the missing nodes is concerned it suffers from the following drawbacks



(a) *Grid-Con* (b) *Grid-Disc*

(c) *AS-Oregon* (d) *HL MT*

Figure 2.7: ***Performance (MCC) of methods across infected degrees*** d_{n_i}***:*** NetFill *performs better than baseline as well as* Top-k-DNI *in the lower buckets.*

1. **Incorrect number of seeds:** The number of seeds found by Top-k-DNI were very large in the case of *Grid-Con*, *Grid-Disc* and *AS-Oregon*. In these datasets we had simulated the SI process from a fixed set of seeds S and hence we expect the same number of seeds to be recovered.

2. **The missing nodes do not depend on the seeds:** As noted in Section 2.5, the missing nodes returned by the Top-k-DNI algorithm does not consider the seed set while choosing the missing nodes. Knowledge of the identity of the seeds can help us finding better missing nodes.

3. **Poor performance in lower infected degree buckets:** If we divide the *MCC* across different infected degree buckets (d_{n_i}) as in Figure 2.7, we see that Top-k-DNI performs very poorly in predicting nodes of lower infected degree. A zero or a negative value of *MCC* means that Top-k-DNI does no better than random chance. The reason the *MCC* scores are low is because Top-k-DNI predicts *every* node with a high d_{n_i} to be infected.

2.6.6 Scalability and Robustness

As mentioned before, Top-k-DNI is linear in the size of the graph in running time. We expect NetFill to be sub-quadratic in practice: Figure 2.6-left shows the wall-clock running time on increasing sizes of the infected subgraphs on the *AS-Oregon* network—it clearly shows that NetFill is *near-linear*, demonstrating its scalability.

We also varied the sampling rate to check robustness: Figure 2.6 right shows the change in precision for NetFill as p changes on the *Grid* network. Although the performance degrades with lower p as expected (more nodes are missing), the decay is smooth over a wide range of values.

2.7 Discussion

The experiments demonstrate that NetFill performs very well—it obtains high precision and *MCC* scores on both synthetic and real data, as well as for both simulated and real-world cascades—outperforming the baselines by a clear margin. Moreover, the close-to-ideal Q scores show that both the seed sets and sets of missing nodes discovered by NetFill are of very high quality: the total description length of the discovered solutions closely approximate the ground truth.

Interestingly, NetFill works well even for the *MemeTracker* and *Flixster* datasets, where virtually none of our assumptions are upheld—for instance, their cascades do not necessarily follow the (idealized) SI model, and the sample rates are unknown, which showcases the power of our method and formulation. Moreover, we simplified these graphs by treating them as undirected and unweighted; incorporating edge directions will likely improve quality, as certain nodes in networks like *MemeTracker* are known to disseminate information (e.g., www.cdc.gov) rather than be subject to infection. Though beyond the scope of this work, our MDL score can fairly straightforwardly be extended to weighted and directed graphs. How edge directions affect Equation 2.1 and our algorithms is an interesting open question.

The small-scale case study for *MemeTracker* also shows that our overall approach works in practice: NetFill recovered 8 *truly* missing infections from a real dataset. It is valid to argue that the SI model is somewhat simplistic for the type of information diffusion in this data. Investigating how NetFill can be extended towards the richer infection models like SIR and SEIR, as well as how to incorporate infection timestamps will make for engaging future research.

2.8 Conclusions

In summary, we studied the problem of finding missing nodes and concealed culprits in noisy infected graphs. We approach the problem using compression, and gave an efficient method, NetFill that approximates the ideal. NetFill automatically recovers both number and identities of missing nodes and seed nodes effectively. Our main contributions include:

(a) *Problem Formulation:* We defined the missing nodes and seeds problem in terms of MDL: the best sets of missing nodes and culprits describes the data most succinctly.

(b) *Fast Algorithm:* We provide a conceptually simple and fast algorithm, NetFill, which principally optimizes our score with an EM-like approach.

(c) *Extensive Experiments:* We showed NetFill outperforms baselines on both synthetic and real datasets and gives meaningful results even when our assumptions may not hold.

Chapter 3

Topic Classification using Temporal Features

In this chapter we utilize the way a contagion spreads in a network to infer the type of the contagion. In particular, we are interested in finding the topic of a keyword without analyzing its textual content. We do so by extracting temporal features from the popularity time series of the keyword. We apply SansText on both keywords and hashtags. This work was accepted in ASONAM 2014 [Sundareisan et al., 2014].

This chapter is structured as follows: First we introduce the problem in Section 3.1 and then we present a brief literature survey in Section 3.2. Next we look at the methodology we propose in Section 3.4. Finally we discuss the experimental setup and the different tasks that we want to solve in Sections 3.5, 3.6 and 3.7.

3.1 Introduction

Twitter and similar forms of social media have are now accepted as *surrogate data sources* that can provide insight into real-world events, e.g., box office sales, influenza outbreaks, and even the movement of earthquakes [Sakaki et al., 2010]. Rather than being passive indicators of such events, the effect of influence cascades on Twitter (e.g., propagation of real information or misinformation such as rumors) have shown that social media is very much an active participant in the progression of such events.

Our goal is to identify signals in Twitter that can serve as precursors to population-level events such as flu outbreaks, civil unrest, and elections. Traditional approaches to characterizing and forecasting such events rely on a fixed vocabulary of keywords or hashtags that are tracked through cascades and which are then input to machine learning models for classification. In practice, however, new hashtags emerge as required for a situation, new keywords

underscore emergent phenomena, and thus purely text-based approaches are inadequate for dealing with the multitude of possible events that could arise.

We demonstrate the use of phenomenological models to capture the dynamics of information propagation, in particular the use of SpikeM [Matsubara et al., 2012]model that leverages statistics about a partially revealed cascade to determine the class of events that the cascade is likely to signify, e.g., a political event versus a sports event. Modeling the rise and fall patterns quantitatively provides a text-free approach to detect and classify emerging events, hence our system's name SansText, which we believe to be of interest more generally.

We demonstrate the utility of this approach on a dataset of more than *2 million* tweets gathered from the Latin American countries of Argentina, Brazil, Chile, Colombia, Ecuador, El Salvador, Mexico, Paraguay, Uruguay, and Venezuela, over the past two years. Latin America is a rich testbed for our project because of the diversity of events that happen due to empowered citizenry and rapid permeation of digital media. In addition to using data on specialized protest events in Latin America, we demonstrate the applicability of our approach more generally on popular high-volume topics on Twitter.

Our contributions can be summarized as:

1. *Problem Formulation and Approach*: We formulate the domain classification problem using activity profiles, and propose a simple yet powerful and efficient low-cost approach based on learning an aggregate information diffusion model (see Section 3.4).

2. *General Topics*: We demonstrate the effectiveness of our approach via first classifying simple popular keywords to domains. We compare against several baselines, and also explore the robustness of our approach to different parameter sets of our model (see Section 3.6).

3. *Protest Data*: We demonstrate the effectiveness of SansText on multiple tasks of different granularity, using protest event data from South America. We also show that SansText can outperform the baselines and is robust to limited data (see Section 3.7).

3.2 Related Works

Although much work has been done on finding topics from tweet text (c.f. [Hong and Davison, 2010]), we briefly review closely related work in the context of the dynamics of information diffusion and other more general time-series methods, as the focus of our thesis is on activity profiles.

Information Diffusion: [Centola, 2010, Gruhl et al., 2004] study the structural properties in the spread of information in networks, including the blogspace. In [Jin et al., 2013a],

the authors model the spread of news, information and rumors in the twitter network. [Romero et al., 2011] observed that hashtags diffuse as a complex contagion, and the nature of information diffusion differs with the topics. They did a study on hash-tags in twitter and found that there is a significant variation in the ways that widely-used hash-tags on different topics spread. The differences in their study manifested mostly in the particular probabilities of infection (the so-called 'P-K' curve) based on the number of friends infected. While their study does provide some insight into the diffusion process, it is not predictive, and it is not built for forecasting of events. In [Bogdanov et al., 2013], the authors study how information propagation is effected by user interests in the twitter network. While in [Rattanaritnont et al., 2012], the authors used the cascade ratio and the tweet ratio to understand how cascades of various topics diffuse in Twitter.

The preceding work looks more at the *structural* aspects of propagation. In contrast there is also work on studying just the *temporal* aspects of information propagation. Crane and Sornette [Crane and Sornette, 2008] studied the rise-and-fall patterns of Youtube video views in a population and found that there were 4 classes, based on a self-excited Hawkes process [Hawkes and Oakes, 1974]. Similarly, Yang et al. [Yang and Leskovec, 2011] explores the temporal patterns associated with online content and found there were 6 classes (associated with different *sources*). Matsubara et al. [Matsubara et al., 2012] showed that all these patterns can be generated from a single unified model SpikeM, which is succinct and yet powerful. SpikeM has been used before to model some malware propagations as well [Papalexakis et al., 2013]. In [Shen et al., 2014], the authors propose a probabilistic framework to model and predict the popularity dynamics of individual items within a complex evolving system. However, these works do not focus on domain/topic classification problems. In this thesis we show how to use such models for challenging domain classifications *without using semantics of the tweet text* itself.

To summarize, none of the above works deal with topic classification of online cascades using just activity profiles of keywords, based on an information diffusion model.

Time-series Analysis: This is an old topic with many textbook approaches [Box and Jenkins, 1990]. Most methods like AR, ARIMA etc. are linear methods (and we describe some of them as baselines later). Non-linear methods tend to be hard to interpret (for example it is hard to relate them to actual physical models) and include [Chakrabarti and Faloutsos, 2002] where the authors propose a fast and completely automated non-linear forecasting system which can provide estimation of vital forecasting parameters. Forex-foreteller (FF) [Jin et al., 2013b] uses a linear regression model to make currency forecasts with high recall over precision. It explores correlative links between news and financial market fluctuations. It uses a language model to classify incoming news articles to build the forecasting model.

3.3 Background

Here we briefly describe the recently proposed SpikeM model [Matsubara et al., 2012] for modeling the popularity of a hashtag cascade. Although it was proposed only for hashtags, as we will show later, we re-purpose it for more general keywords.

We are interested in the macroscopic properties of hashtag cascades in the network. The model assumes that if a user has used the hashtag in his tweet, that user has been infected. Once infected the user always stays in the infected state (as he or she already knows about the hashtag).

The model assumes a total number of N un-informed population ('bloggers') that can be informed ('infected') by the hashtag. Let $U(n)$ be the number of such bloggers that are *not* infected at time n; $I(n)$ be the count of bloggers that got infected up to time $n-1$; and $\Delta I(n)$ be count of bloggers infected exactly at time n. Then $U(n+1) = U(n) - \Delta I(n+1)$ with initial conditions $\Delta I(0) = 0$ and $U(0) = N$.

Additionally, we let β as the 'infectivity' (essentially popularity) of a particular hashtag. We assume that the popularity of a hashtag at any particular person drops as a specific power-law based on the elapsed time since the hashtag infected *that person* (say τ) i.e. $f(\tau) = \beta\tau^{-1.5}$. Finally, we also have to consider one more parameter for the model: the "external shock", or in other words, the first appearance of a hashtag: let n_b the time that this initial burst appeared, and let $S(n_b)$ be the size of the shock (count of infected bloggers).

Finally, to account for periodicity, we define a periodic function $p(n)$ with three parameters: P_a, as the strength of the periodicity, P_p as the period and P_s as the phase shift.

Putting it all together, the SpikeM model is

$$\Delta I(n+1) = p(n+1)\left(U(n)\sum_{t=n_b}^{n}\left(\Delta I(t) + S(t)\right) f(n+1-t) + \epsilon\right)$$

where $p(n) = 1 - \frac{1}{2}P_a\left(\sin\left(\frac{2\pi}{P_p}\left(n + P_s\right)\right)\right)$, and ϵ models noise.

3.4 Methodology

3.4.1 Problem Formulation

Informally, the general problem we aim to solve is to classify a keyword into the correct domain, depending only on the temporal characteristics of the activity profile of that keyword. Note that there can be more sophisticated methods which can be employed for this purpose which leverage the actual tweet text as well. Nevertheless we believe our framework gives a different low-cost approach, which is surprisingly powerful and gives robust results and hence is interesting in its own right. More formally our problem stated as:

GIVEN: The temporal activity profile for a keyword on Twitter

FIND: The correct domain class label for the keyword.

In this thesis, the particular 'domains' and 'keywords' are motivated by two real-world examples: popular high-volume topics like politics, sports etc., and more specialized 'protest-types' which categorize real-world protest events in Latin America. We will describe these in more detail later in Sections 3.6.1 and 3.7.1.

3.4.2 Proposed Approach

Romero et al [Romero et al., 2011] discuss the differences in the mechanics of information diffusion, particularly in the so-called 'probability of infection' curve, across different domains. With this in mind, we posit that even the temporal propagation signature of hashtags from different domains are likely to be different, and these differences affect the popularity time series for each hashtag. Further, model parameters obtained for the popularity time series for hashtags from the same domain exhibit similarities, which can be learnt by using appropriate algorithms.

The SpikeM [Matsubara et al., 2012] Model provides an analytical tool for modeling popularity time series. It fits an exponential rise and a power law fall to data, and takes into account the periodicity of the activities too, as described in Section 3.3. It is able to model real Twitter data well, where data shows exponential rises and power law falls, along with periodic trends with peaks during weekends. This model has the added advantage that model parameters consider orthogonal aspects of the spread of the infection. Hence we propose to

The SpikeM model has seven parameters. A list of the parameters with a brief explanation is given in Table 3.1. If $X(n), n = 1 \cdots T$ is the sequence of count of keyword occurrences we want to model, we minimize the following:

$$\min_{\boldsymbol{\theta}} \sum_{n=1}^{T} \left(X(n) - \Delta I(n)\right)^2$$

where $\boldsymbol{\theta} = \begin{bmatrix} N & \beta & S_b & P_a & P_s & P_p \end{bmatrix}^T$ is the vector of model parameters. We use Levenberg-Marquardt [Levenberg, 1944] to learn the parameters.

Using the SpikeM parameters learnt from keyword activity profiles as features and training data, our framework SansText learns a classifier that can classify keywords to domains. Classification accuracy is used as a metric to judge the ability of a classifier to classify hashtags to domains.

We find that, though SpikeM parameters can be used successfully to predict the topic of a keyword, only a sub-set of them are relevant to the classification problem. Thus, we further analyze the importance of each SpikeM parameter to domain-wise classification. Some

S_b	The value of the external shock applied
β	The infectivity parameter of the virus
n_b	The time at which the external shock S_b was applied
P_a	The amplitude of the periodic part of the time series
P_s	The phase shift of the periodic part of the time series
P_p	The periodicity of the time series
ϵ	The error term

Table 3.1: List of parameters used in the SpikeM *model*

parameters like n_b, which determines the day the news broke out in the social network, may not be a good predictor for all topics. On the other hand, the value of N, which is the number of people interested in the topic (the inherent 'audience'), turned out to be useful.

We next describe our extensive set-up for SansText in more detail.

3.5 Experimental Setup

3.5.1 Overview

The motivation for our SansText approach is to classify entire tweets based on a small number of keywords. The most commonly used keywords are 'hash-tags', which generally denote particular contexts, and have been extensively used in Twitter studies before. For example, tweets that have the hash-tag "#manchesterutd" would talk about the soccer club Manchester United and hence the tweet can be classified as belonging to the topic sports. Another way to find such keywords is to use the help of domain experts to find out which keywords are popular in a particular topic (especially in context of protests datasets).

We use both these methods while collecting the data for our experiments. Once these keywords are collected the next step is to collect tweets that have this keyword. We use a large sample of all South American tweets for this purpose. Geographic targeting was done through both geo-tagging and a user referencing their own location in the tweet text or their profile. We use these collected tweets and aggregate them to count the number of occurrences of a keyword in a time period. Thus we obtain a volume time series for each keyword which we can fit with the SpikeM model and collect the parameters of the model. The next step is to use these parameters as features for the classification problem of finding which topic the keyword/hash-tag belongs to. We use different methodologies for getting the ground-truth labels for keywords for each of the experiments, which we will describe later. A subtle point is that we do not use parameter n_b from set of parameters in Table 3.1 for classification, as it represents the day the initial spike was observed and hence is not an *inherent* property of the spreading cascade. It is easy to see that this makes some of our

predictions *harder*, as there are some topics occur only in certain times of the year e.g. flu occurs mainly in March-Sept (in S. America), and hence just by looking at the timestamps we can guess if the topic of a keyword is flu (although this approach may not work for more perennial). Moreover, as our focus in this thesis is on actual dynamics, we do not use this feature.

After learning the parameters we test SansText against multiple intuitive and non-trivial baselines listed in 3.5.2 using classifiers described in 3.5.3. We describe this process in more detail for each of our experiments later in Sections 3.6.1 and 3.7.1.

To throughly evaluate our approach, in our experiments we pose the following research questions:

1. Does the choice of the time interval of the aggregation change our results?
2. Are SansText parameters a good feature set for domain-wise classifications?
3. Which SansText parameters are important to the classification problem for?
4. Can we use SansText when the keywords are spread across multiple topics?
5. How much data do we need to make a successful classification?

3.5.2 Baselines

We compared SansText to a trivial baseline (Majority) and three non-trivial baselines each of which essentially give us features for each time-series. We give these baselines the same data that SansText gets i.e. a set of time series. We describe each of them briefly:

Majority: The Majority baseline always gives the output of the class with maximum frequency no matter what the input is. Hence, we are not performing any 'learning' in this method. We use this baseline to indicate which of the methods actually learn useful patterns in the dataset.

Euclidean: The Euclidean distance is the simplest distance between two series x and y of length n and is defined as $\sqrt{\sum_{i=0}^{n}(y_i - x_i)^2}$. A distance of zero implies that the two series are exactly the same. The attributes chosen for classification were the distances to every other keyword. The intuition for this baseline stems from the idea that all keywords from the same topic would have small distances. We would like to point out that such a method has also been used frequently [Keogh and Kasetty, 2002]. Apart from the natural problems of the euclidean distance (as it is too 'rigid'), the other disadvantages of this baseline are that for every new keyword we need to calculate the distances to all other keywords in the dataset and that we need to save all $\binom{n}{2}$ distances for classification.

DTW: Dynamic Time Warping (DTW) is also a popular robust distance metric between two time series. It is extensively used in areas like speech processing [Rabiner and Juang, 1993]. The main difference between DTW and EUCLIDEAN is that DTW calculates the distance by taking into account that the two series can have peaks at different times. In other words DTW can insert or delete some elements in the series so as to minimize the distance between them. Just like EUCLIDEAN, we use the distances from each of the keywords as parameters for the classification problem. We used the recent fast implementation by Rakthanmanon et. al. [Rakthanmanon et al., 2013] for finding out these distances[1]. Just like EUCLIDEAN the major disadvantages of this baseline are that for every new keyword we need to calculate the distances to all other keywords in the dataset and that we need to save all $\binom{n}{2}$ distances for classification.

Fourier: The Fourier Transform decomposes a given function into a sum of periodic functions of the form $e^{i\pi n}$ [Briggs and Emden, 1995]. The Discrete Fourier Transform (DFT) uses functions of the form $e^{\frac{i*2\pi k}{N}}; k \in I; 0 \leq k \leq N$ [Briggs and Emden, 1995]. Thus, the DFT gives a finite set of coefficients for a discrete time series.

For discrete time series of equal lengths, the same functions are used as the basis when computing the DFT. Each coefficient generated in the DFT can be treated as a feature of the given time series. Intuitively, the DFT allows us to identify the significant periodicities in a time series, thus allowing learning and classification based on similarities in the periodic nature of time series. The disadvantage of this baseline is that we have to save coefficients atleast half the size of the time series. To compute the DFT, we use the Cooley-Tukey FFT algorithm [Cooley and Tukey, 1965], as implemented in the NumPy numerical analysis package for Python.

3.5.3 Classifiers

We use the following popular classifying methods for our experiments:

1. *Multilayer Perceptron*: In this method, the weights between 3 layers of a neural network are learnt using back propagation.
2. *C4.5*: This is a classic tree based method which uses Information gain as the criteria to split the attributes.
3. *Random Forests*: A set of different classification trees are constructed using different subsets of parameters to learn the model.
4. *Bagging*: It is a method that is used to reduce the variance in the model that we learn by using the same algorithm on different training sets generated by random sampling. We use as the base classifier as REPTree.

[1]Code can be found here: `http://www.cs.ucr.edu/~eamonn/UCRsuite.html`

5. *Logistic Regression*: Uses the logistic function in order to find a boundary between various classes.

A more detailed description and comparisons between these supervised learning methods is in [Caruana and Niculescu-Mizil, 2006]. In all our tests we use Weka's [Hall et al., 2009] implementations of these algorithms with the default parameters to the algorithms, unless stated otherwise. Additionally, in all our experiments we use 10-fold cross validation to report the classification accuracy.

3.6 Experiments on Popular Dataset

3.6.1 Data Collection

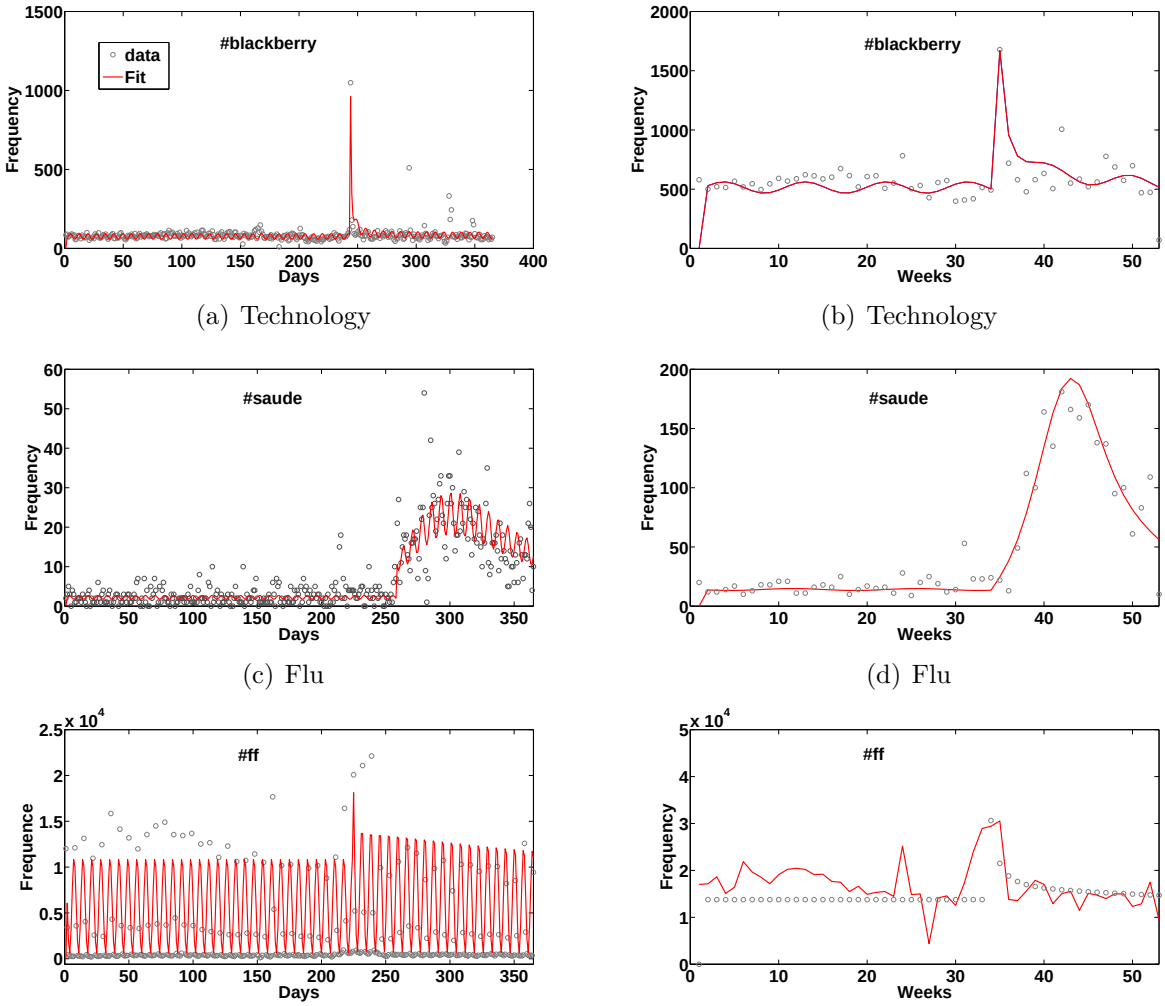
In this study, we are interested in tweets from these popular domains: Political, Flu, Sports, Technology and Idioms (these domains have also been used before in prior studies). We define these domains and give examples in Table 3.2. Using Datasift's collection service[2], we collected a list of top 300 hash-tags (by volume) from June 2012 to May 2013 and divided it into these domains. The division was carried out by using majority vote among three of the authors (similar to methodology used in [Romero et al., 2011]). We show the division of the hash-tags in the Appendix. The domains Flu, Idioms, Technology, Sports and Political had 11, 10, 11, 12 and 14 hash-tags respectively. After collecting these hash-tags we extract the timestamps of each of occurrence of the hash-tag in a tweet (again from June 2012-May 2013). We then aggregate these occurrence numbers by day and by week to generate a time series of the mentions of the hashtag. Note that every hash-tag would have its own time-series. We then used our model to find the set of parameters that fit the time series. We show one of these plots for three different domains for both the weekly and the daily settings in Figure 3.1.

3.6.2 Results

3.6.2.1 Does the Choice of the Time Interval of the Aggregation Change our Results?

SansText exploits the particular rise-fall nature of the time-series for fitting the model. Hence it is possible that too high/low a granularity will bury/wash out the patterns. We performed all the experiments in this section with both the daily setting as well as the weekly setting—both of these settings show similar results. Part of the reason is that SansText has

[2] `www.datasift.com`



(a) Technology (b) Technology

(c) Flu (d) Flu

(e) Idiom (f) Idiom

Figure 3.1: How well does SansText *model the data? We compare the learnt model with the data for Daily (top row) and weekly (bottom row) granularity for different domains, and plot volume per unit time against time. We observe that* SansText *(blue) does a good job for fitting the exponential rise and power-law fall of the real data (red).*

Topic	Description
Idioms	Hash tags that are a group of words or their abbreviations connected together. These hash tags have a conversational meaning like #ff stands for "Follow Friday" a trend which recommends whom to follow this Friday on Twitter.
Flu	Hash tags that related to being hit by Flu like symptoms of flu. Example #fiebre which means Fever.
Technology	Hash tags which relate to Technology like #apple (the company).
Sports	Hash tags which relate to sports like #londres2012, the Olympics in 2012.
Politics	Hash tags which talk about politics like #caprilespresidente which talks about Henrique Capriles Radonski for the next president of Venezuela.

Table 3.2: Description of topics used in Popular Dataset.

Class	Description
Non Violent Government Policies	Policies by the government that resulted in non-violent protests.
Non Violent Energy and Resources	Protests over energy or resources that were non-violent eg: Hike in gas prices.
Violent Energy and Resources	Protests over energy or resources that were violent.
Non Violent Other	Non-violent protests that have reasons other than energy, resources, govt. policy, housing and employment
Violent Other	Violent protests that have reasons other than energy, resources, govt. policy, housing and employment

Table 3.3: Class definition for 5 event type classification problem in Protest Dataset

an explicit periodicity parameters, so it is tolerant to an extent to such natural aggregation levels. Hence due to lack of space, we describe the results of just the daily setting.

3.6.2.2 Are SansText Parameters a Good Feature Set for Domain-wise Classification ?

In short from Figure 3.3(a), SansText outperforms all the baselines. Hence the parameters that we use are a good set of features for classification in the Popular Dataset.

We used the parameters that we get from the model fittings as attributes and used classifiers listed in section 3.5.3. Since we use 5 different classification algorithms for each method we only report the values for the best performing one (it could be different for different methods). We show the % improvement of the methods over the weakest method in Figure 3.3(a). So SansText performed 173% better than Majority insofar as classification accuracy

	N	β	S_b	ϵ	P_a	P_s	P_p
N	1.00	$\emptyset$	$\emptyset$	0.86	$\emptyset$	$\emptyset$	$\emptyset$
β	$\emptyset$	1.00	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
S_b	$\emptyset$	$\emptyset$	1.00	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
ϵ	0.86	$\emptyset$	$\emptyset$	1.00	$\emptyset$	$\emptyset$	$\emptyset$
P_a	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	1.00	0.36	$\emptyset$
P_s	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	0.36	1.00	$\emptyset$
P_p	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	1.00

Figure 3.2: ***Correlation of parameters in Popular Dataset:*** *Are the seven parameters correlated in the Popular Dataset ? As we can see there is little correlation between the parameters. All values in the range (-0.3, 0.3) are shown as $\emptyset$. Please see section 3.6.2.3 for explanation on why N and ϵ have a high correlation.*

Method	% improvement
Majority	0
Fourier	103
Euclidean	155
DTW	134
SansText	173

(a) Popular Dataset

	% improvement
SansText	165
Fourier	27
DTW	0
Euclidean	54
Majority	33

(b) Protest **Task 1**

	% improvement
SansText	40
Fourier	0
DTW	20
Euclidean	11
Majority	5

(c) Protest **Task 2**

Figure 3.3: ***Performance on datasets:*** *How well do we perform compared to the baselines? As we can see we are performing way better than any other baseline in all the three cases*

is concerned. Since Political was the class with the highest frequency of hashtags, the Majority will always predict every hashtag to be Political.

As expected Fourier does not perform well, as while DFT does well with periodicities, it does not do well with spikes (as we need co-efficients from across the frequency spectrum because of a spike in the time-domain). A bit surprising result was the Euclidean fared better than DTW in the daily setting while not in weekly setting. This is because in the daily setting the time-series have multiple local peaks (due to similar periodicities) which align across the time-series (while the major rise-fall peak itself may not align). Euclidean will get a better distance, whereas DTW tries to align the main peak and makes mistakes in the local ones. On the other hand, in the weekly setting, there are fewer periodicities, and hence aligning the main peak is more important, which DTW does it successfully. This also shows that for any method to be successful we should treat the periodic nature of the time series different from the actual rise and fall for the method to be robust across different settings—which is exactly achieved by SansText.

3.6.2.3 Which SansText Parameters are Important to the Classification Problem ?

In short, from the correlation matrices in Figure 3.2 and the % improvement when we remove parameters in Figure 3.4(a), we can infer that *all* the parameters are useful for classification.

To answer this question we investigated two things: (a) we removed each parameter one by one and found out its effect on the classification accuracy; and (b) we measured the correlation coefficient of each of the features. We show the results of this experiment for the algorithm that produced the best results in the previous section in Figure 3.4(a). We compare this result with the correlation coefficient between our parameters. We are interested in finding out if there exists redundancy in the set of attributes that we use for classification. If any two of the attributes are highly correlated then we can reduce the feature set. We computed the Pearson's correlation coefficient between all the 7 attributes we used for classification. The correlation matrix is presented in Figure 3.2.

We observe from Figure 3.4(a) that only removing β increases the classification accuracy (albeit very slightly). But we observe that β is not correlated with any other parameter as shown in the Figure 3.2. We also observe that only two parameters have a high value of correlation(> 0.5). These parameters are N and ϵ. We argue that by removing either one of these would affect the classification accuracy as shown in Figure 3.4(a). The reason that the Pearson's coefficient is large between them is because when there are a lot of people talking about a topic(N) there is a lot of noise(ϵ) in the dataset. From the C4.5 tree we learn for this dataset, we infer that Sports hashtags generally have a high noise ($\epsilon > 250$). Also Technology hashtags have a large shock ($S_b > 14000$) or have a lot of fan following ($\epsilon > 100$).

3.6.2.4 How Much Data do We Need to Learn SansText Parameters ?

In this section we will try to find out how much of the time series data is needed to learn SansText parameters for the Popular Dataset.

We took the hash-tag #saude from the topic Flu for this part of the experiment. We tried learning the parameters for SansText for data till n_b, data till n_b-3 and data till n_b+3 and plotted the resulting fits in Figure 3.7(left). We observe that all the plots look similar till n_b+3. This suggests that we learn parameters in SansText in a robust way. With more data the Root Mean Square Error (RMSE) decreases as follows: 22.14(green), 19.25(blue) and 16.91 (red curve). This also implies that we do not need data till the peak in order to learn the parameters.

Parameters Used	% improvement
$\Theta \setminus P_a$	-5.2
$\Theta \setminus P_s$	-2.6
$\Theta \setminus P_p$	-23.1
$\Theta \setminus N$	-7.7
$\Theta \setminus \beta$	5.1
$\Theta \setminus \epsilon$	-20.6
$\Theta \setminus S_b$	-20.6

(a) Popular Dataset

Parameters Used	% improvement
$\Theta \setminus P_a$	-45.5
$\Theta \setminus P_s$	-4.6
$\Theta \setminus P_p$	-27.2
$\Theta \setminus N$	-18.2
$\Theta \setminus \beta$	9
$\Theta \setminus \epsilon$	0
$\Theta \setminus S_b$	-18.2

(b) Protest **Task 1**

Parameters Used	% improvement
$\Theta \setminus P_a$	-9
$\Theta \setminus P_s$	-11.5
$\Theta \setminus P_p$	-6.6
$\Theta \setminus N$	-16.9
$\Theta \setminus \beta$	3
$\Theta \setminus \epsilon$	-8.9
$\Theta \setminus S_b$	-13.9

(c) Protest **Task 2**

Figure 3.4: ***Ablation Test:*** *As we can see by removing the parameters one by one, we reduce the classification accuracy in most cases for the 3 datasets. Here* $\Theta = \{N, \beta, \epsilon, P_a, P_s, P_p, S_b\}$*, the list of parameters used in* SansText*.*

	N	S_b	ϵ	P_a	P_s	P_p	β
N	1	$\emptyset$	0.40	$\emptyset$	$\emptyset$	$\emptyset$	-0.30
S_b	$\emptyset$	1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
ϵ	0.40	$\emptyset$	1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
P_a	$\emptyset$	$\emptyset$	$\emptyset$	1	-0.39	0.38	$\emptyset$
P_s	$\emptyset$	$\emptyset$	$\emptyset$	-0.39	1	$\emptyset$	$\emptyset$
P_p	$\emptyset$	$\emptyset$	$\emptyset$	0.38	$\emptyset$	1	$\emptyset$
β	-0.30	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	1

(a) Protest **Task 1**

	N	S_b	ϵ	P_a	P_s	P_p	β
N	1	0.32	$\emptyset$	0.45	$\emptyset$	$\emptyset$	$\emptyset$
S_b	0.32	1	$\emptyset$	$\emptyset$	0.32	$\emptyset$	-0.33
ϵ	$\emptyset$	$\emptyset$	1	0.48	$\emptyset$	$\emptyset$	$\emptyset$
P_a	0.45	$\emptyset$	0.48	1	-0.36	$\emptyset$	$\emptyset$
P_s	$\emptyset$	0.32	$\emptyset$	-0.36	1	$\emptyset$	-0.27
P_p	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	1	$\emptyset$
β	$\emptyset$	-0.33	$\emptyset$	$\emptyset$	-0.27	$\emptyset$	1

(b) Protest **Task 2**

Figure 3.5: ***Correlation between parameters of*** **SansText** ***in Protest Dataset:*** *As we can see from the correlation matrix for* ***Task 1*** *and* ***Task 2*** *that there is no correlation between the parameters which means that we can't remove any one of them. All values in the range (-0.3, 0.3) are shown as* $\emptyset$*.*

3.7 Experiments on Protest Data

3.7.1 Data Collection

For the second dataset, we have access to a Gold Standard Report (GSR) of protests organized by an independent third party (MITRE). The GSR is a database generated by human analysts who scour newspapers in Latin America for reported happenings of civil unrest. This report has a list of all protests along with other metadata like where the protest occurred, the type of the event, the type of population involved, the date of the event, and the date of reporting. We used the keywords collected for these events by [Hua et al., 2013]. The authors in [Hua et al., 2013] used location based collection of tweets surrounding the event with a date range of ± 10 days and used TF-IDF to find the important keywords. Hence these keywords are important around the time and the location of the event. A subtle point is that each event could have multiple keywords associated with it. With these sources for our data we try to solve two tasks

3.7.1.1 Task 1: Keyword to Event Type Mapping

We have a set of keywords that are known to be related to protests. We can monitor each one of them and find out if they are gaining popularity or not. If a protest related keyword is known to show popularity over a period of time we want to find out if the temporal pattern it displays can help us predict which event type the developing protest belongs to. Since every keyword can belong to multiple event types, we try to find out if SansText can be used to predict the event type in such a scenario.

3.7.1.2 Task 2: Event to Event Type Mapping

We find that many events have more than one keyword associated with them. Just as done in task one, we monitor the popularity of various keywords which are known be related to protests. When we observe a set of keywords trending from one geographic location at around the same time, we will collect all their mentions and treat them as an event. In this task, we try to predict what event type an event belongs to. We consider every event as the aggregation of all the keywords associated with it. Thus for our classification problem each event is just a time series.

We describe some of the event types that are possible in Table 3.3. We used the event types as the topics for the classification problem. Note that every keyword can belong multiple event types. We collect all the tweets that use these keywords in the ± 10 days of the event. We then fit our model on this time series and find all the parameters. We then use these parameters as features to predict the event type using the classifiers described in section 3.5.3.

As the results we obtain in both tasks were similar we will describe the results of both the tasks together. We show the keywords and their classes in the Appendix.

3.7.2 Results

3.7.2.1 Can we Use SansText When the Keywords are Spread Across Multiple Topics?

In short from the % improvement Figure 3.3(b), we can infer that SansText works even when the keywords are spread unto different topics. We perform better than all other baselines.

Till now we used keywords that belong to a single class. There are instances in which a keyword can belong to different classes. Since SansText does not look at the context with which the keyword was used, every keyword-event type instance is a new data point in out setting. We propose to find the event type by looking at how the behavior changes when

a keyword is used for different event types. Note that this sort of classification would be a hard task if we were to use traditional methods like NLP since we would have to use the context around the keyword in the tweet text. We can see in Figure 3.6(top row) that even though the keyword 'cantar' belongs to various event types the pattern observed in each one of them is different. We try to use these differences in classifying the keyword 'cantar'.

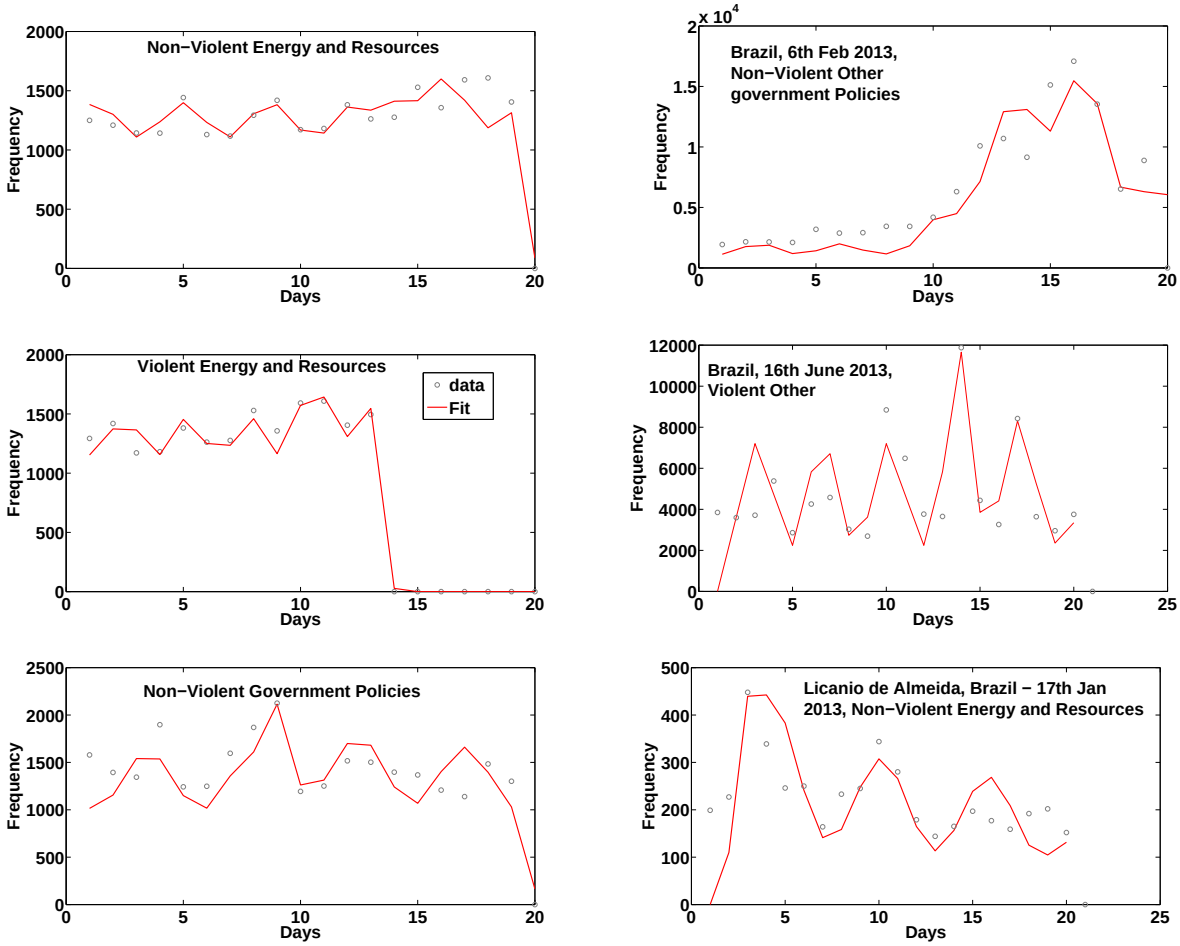


Figure 3.6: How well does SansText *model the data in Protest* ***Task 1*** *(left) and Protest* ***Task 2*** *(right)? We plotted the volume per unit time for the same keyword for different event types. As we can see for* ***Task 1*** *the patterns are not similar. We use this dissimilarity in* ***Task 1*** *to classify the keyword 'cantar' to an event type. While for* ***Task 2****, we show 3 events. We mention the type of protest with a text in the the figure.*

Since we use 5 different classification algorithms for each method we only report the values for the best performing one (it could be different for different methods). For **Task 1**, Figure 3.3(b) shows that SansText gives an improvement of 165% over the weakest baseline i.e. DTW. SansText performs atleast twice as better than most of the baselines while we perform 72% better than the next best baseline i.e. Euclidean. While for **Task 2**, Figure 3.3(b) shows that SansText performs better than all the baselines.

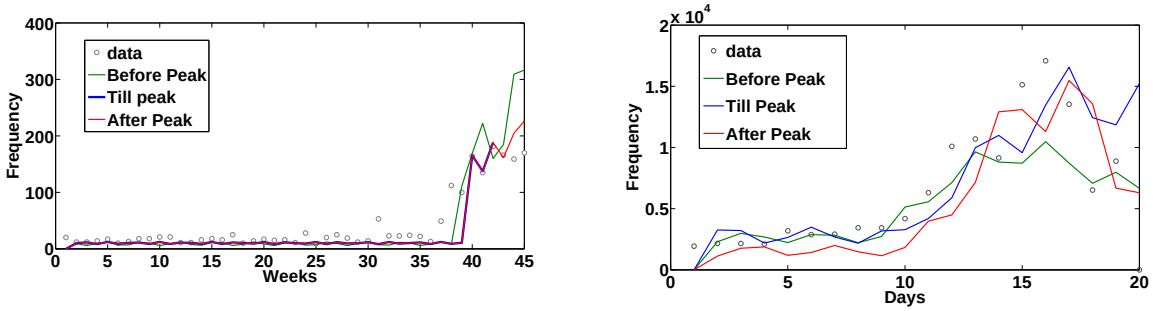
3.7.2.2 Which SansText Parameters are Important to the Classification Problem?

As we can see from the correlation matrices in Figure 3.5 and the % improvement while removing parameters one by one in Figure 3.4(right), we can infer that all the parameters used are important. Hence we can't remove even one of them from SansText.

To answer this question we removed each parameter one by one and found out its effect on the classification accuracy (i.e. an ablation test). We show the results of this experiment for the algorithm that produced the best results in the section above i.e. Multilayer Perceptron in Figure 3.4(right) for both **Task 1** as well as **Task 2**. We compare this result with the correlation coefficient between our parameters. Again, if any two of the attributes are highly correlated then we can reduce the feature set. We found the Pearson's correlation coefficient between all the 7 attributes we used for classification (see Figure 3.5). We observe from Figure 3.4(right) that like in the Popular Dataset only removing β increases the classification accuracy. But we observe that β is not correlated with any other parameter as shown in the Figure 3.5. We observe from Figure 3.5 that there is no correlation between any of the parameters and hence we can not remove any of the parameters from SansText.

From the C4.5 classification tree we can infer that Non-Violent protests of type 'Energy and resources' have low fluctuations in temporal data if the spike is ignored($P_a < 0.4$). While keywords belonging to Non-Violent Other category had less than 29000 tweets in this dataset.

3.7.2.3 How Much Data do we Need to Learn SansText Parameters?



(a) Popular Dataset (b) Protest Dataset

Figure 3.7: ***Robustness in* SansText *for data:*** *How much data do we need to fit the* SansText *parameters? As we can see all the three plots are similar. This implies that we are robust in determining the* SansText *parameters accurately.*

In this section we will try to find out how much of the time series data is needed to learn SansText parameters for the Protest Dataset.

We took the keywords for the event that corresponded to the Non-Violent Government

Policies on 6th Feb 2013 in Brazil. We tried learning the parameters for SansText for data till n_b, data till n_b-3 and data till n_b+3 and plotted the resulting fits in Figure 3.7(right). Clearly, we will learn better given more data, but we find that SansText's performance is robust and does not degrade much. With more data the Root Mean Square Error(RMSE) decreases as follows: 2130.2(green), 1997.83(blue) and 1771.38(red curve) . This also implies that we do not need data till the peak in order to learn the parameters.

3.8 Conclusion

We have demonstrated that activity profiles for hashtags from different domains are modeled well under our framework SansText, and showcased its utility for domain classification without requiring any textual analysis. We have demonstrated its effectiveness and superiority over baseline methods in both a general topical domain and in a specialized domain, viz. protest modeling. Inference of the parameters enables the forecasting of keyword and event popularity and classification into different granularities, as evidenced by our results over Latin American tweets.

Chapter 4

Conclusions

In conclusion, this thesis attempts to use propagation dynamics with epidemiologically inspired models in order to solve data mining tasks like finding missing data, finding patient zeros and topic classification sans text.

In Chapter 2, we studied the problem of finding missing nodes and concealed culprits in noisy graphs. We approach the problem using compression, and gave an efficient method NetFill, to approximate the ideal. NetFill can automatically, and with high precision and *MCC*, recover both number and identities of missing nodes—as well as the number and identities of the seed nodes.

In Chapter 3, we demonstrated that activity profiles for hashtags from different domains are modeled well under our framework SansText, and showcased its utility for domain classification without requiring any textual analysis. We have demonstrated its effectiveness and superiority over baseline methods in both a general topical domain and in a specialized domain, viz. protest modeling. Inference of the parameters enables the forecasting of keyword and event popularity and classification into different granularities, as evidenced by our results over Latin American tweets.

Future Work: There exist many avenues for future work. Prime examples include considering richer infection models, such as SIR, considering multiple snapshots instead of just one, and in particular to explore efficient methods for holistically optimizing the MDL score with regard to both C^- and C^+.

Even by treating the *MemeTracker* and *Flixster* datasets as undirected and unweighted NetFill provides good results as seen in Section 2.6. Incorporating weights would mean that every edge has a different probability of infection β for the SI process.

Directed edges can be easily incorporated in calculating the MDL. However we do need to analyze how adding direction to the edges would change Equation 2.1. We feel that

considering directed edges would improve the quality of the nodes we find as some nodes in networks like *MemeTracker* are known to disseminate information (www.cdc.gov) rather than get infected from other nodes.

Also we need to consider the time at which nodes are infected. This information would help us to construct more accurate ripples in datasets where time of the infection is also reported (like *MemeTracker*).

Finally the SI model may be a bit too simplistic to model information diffusion in some datasets. Extension of the method to accommodate SIR, SEIR etc would be interesting and useful.

For the second problem we studied in Chapter 3, future works may look into learning even the exponent parameter (currently 1.5) in our framework and using it for classification. Other unanswered questions include (a) Can we use the parameters used in SansText to predict the popularity of the protest even before it occurs ? (b) Can we predict if a keyword belongs to a protest type or not? (c) Can we find a set of parameters from the popularity graphs that can be optimally used in SansText to make a better prediction?

Bibliography

[Anderson and May, 1991] Anderson, R. M. and May, R. M. (1991). *Infectious Diseases of Humans*. Oxford University Press.

[Bikhchandani et al., 1992] Bikhchandani, S., Hirshleifer, D., and Welch, I. (1992). A theory of fads, fashion, custom, and cultural change in informational cascades. *J. Polit.Econ.*, 100(5):992–1026.

[Bogdanov et al., 2013] Bogdanov, P., Busch, M., Moehlis, J., Singh, A. K., and Szymanski, B. K. (2013). The social media genome: Modeling individual topic-specific behavior in social media. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 236–242, New York, NY, USA. ACM.

[Borgatti et al., 2006] Borgatti, S. P., Carley, K. M., and Krackhardt, D. (2006). On the robustness of centrality measures under conditions of imperfect data. *Soc. Netw.*, 28(2):124 – 136.

[Box and Jenkins, 1990] Box, G. E. P. and Jenkins, G. (1990). *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated.

[Briesemeister et al., 2003] Briesemeister, L., Lincoln, P., and Porras, P. (2003). Epidemic profiles and defense of scale-free networks. *WORM*.

[Briggs and Emden, 1995] Briggs, W. L. and Emden, H. V. (1995). *The DFT - an owner's manual for the discrete Fourier transform.* SIAM.

[Caruana and Niculescu-Mizil, 2006] Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM.

[Centola, 2010] Centola, D. (2010). The Spread of Behavior in an Online Social Network Experiment. *science*, 329(5996):1194.

[Chakrabarti and Faloutsos, 2002] Chakrabarti, D. and Faloutsos, C. (2002). F4: Large-scale Automated Forecasting Using Fractals. In *Proceedings of the Eleventh International*

Conference on Information and Knowledge Management, CIKM '02, pages 2–9, New York, NY, USA. ACM.

[Chakrabarti et al., 2004] Chakrabarti, D., Papadimitriou, S., Modha, D. S., and Faloutsos, C. (2004). Fully automatic cross-associations. In *KDD*, pages 79–88.

[Chakrabarti et al., 2008] Chakrabarti, D., Wang, Y., Wang, C., Leskovec, J., and Faloutsos, C. (2008). Epidemic thresholds in real networks. *ACM TISSEC*, 10(4).

[Chen et al., 2010] Chen, W., Wang, C., and Wang, Y. (2010). Scalable influence maximization for prevalent viral marketing in large-scale social networks. *KDD*.

[Cilibrasi and Vitányi, 2005] Cilibrasi, R. and Vitányi, P. (2005). Clustering by compression. *IEEE TIT*, 51(4):1523–1545.

[Cooley and Tukey, 1965] Cooley, J. W. and Tukey, J. W. (1965). An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297–301.

[Costenbader and Valente, 2003] Costenbader, E. and Valente, T. W. (2003). The stability of centrality measures when networks are sampled. *Soc. Netw.*, 25(4):283–307.

[Cover and Thomas, 2006] Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory*. Wiley-Interscience.

[Crane and Sornette, 2008] Crane, R. and Sornette, D. (2008). Robust dynamic classes revealed by measuring the response function of a social system.

[Faloutsos and Megalooikonomou, 2007] Faloutsos, C. and Megalooikonomou, V. (2007). On data mining, compression and Kolmogorov complexity. In *Data Min. Knowl. Disc.*, volume 15, pages 3–20. Springer-Verlag.

[Faloutsos et al., 1999] Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the internet topology. In *SIGCOMM*.

[G-Rodriguez et al., 2010] G-Rodriguez, M., Leskovec, J., and Krause, A. (2010). Inferring networks of diffusion and influence. In *KDD*.

[Ganesh et al., 2005] Ganesh, A., Massoulié, L., and Towsley, D. (2005). The effect of network topology on the spread of epidemics. In *IEEE INFOCOM*.

[Goldenberg et al., 2001] Goldenberg, J., Libai, B., and Muller, E. (2001). Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*.

[Gomez-Rodriguez et al., 2011] Gomez-Rodriguez, M., Balduzzi, D., and Schölkopf, B. (2011). Uncovering the temporal dynamics of diffusion networks. In *ICML*.

[Goyal et al., 2010] Goyal, A., Bonchi, F., and Lakshmanan, L. V. (2010). Learning influence probabilities in social networks. WSDM '10.

[Goyal et al., 2011] Goyal, A., Lu, W., and Lakshmanan, L. V. S. (2011). Simpath: An efficient algorithm for influence maximization under the linear threshold model. *ICDM*.

[Gruhl et al., 2004] Gruhl, D., Guha, R., Liben-Nowell, D., and Tomkins, A. (2004). Information diffusion through blogspace. In *WWW*.

[Grünwald, 2007] Grünwald, P. (2007). *The Minimum Description Length Principle*. MIT Press.

[Habiba and Berger-Wolf, 2011] Habiba and Berger-Wolf, T. (2011). Working for influence: effect of network density and modularity on diffussion in networks. *ICDM DaMNet*.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).

[Hawkes and Oakes, 1974] Hawkes, A. G. and Oakes, D. (1974). A Cluster Process Representation of a Self-Exciting Process. *Journal of Applied Probability*, 11(3):493–503.

[Hayashi et al., 2003] Hayashi, Y., Minoura, M., and Matsukubo, J. (2003). Recoverable prevalence in growing scale-free networks and the effective immunization. *arXiv:cond-mat/0305549 v2*.

[Hong and Davison, 2010] Hong, L. and Davison, B. D. (2010). Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA. ACM.

[Hua et al., 2013] Hua, T., Lu, C.-T., Ramakrishnan, N., Chen, F., Arredondo, J., Mares, D., and Summers, K. (2013). Analyzing Civil Unrest through Social Media. *Computer*, 46(12):80–84.

[Jin et al., 2013a] Jin, F., Dougherty, E., Saraf, P., Cao, Y., and Ramakrishnan, N. (2013a). Epidemiological Modeling of News and Rumors on Twitter. In *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, SNAKDD '13, pages 8:1–8:9, New York, NY, USA. ACM.

[Jin et al., 2013b] Jin, F., Self, N., Saraf, P., Butler, P., Wang, W., and Ramakrishnan, N. (2013b). Forex-foreteller: Currency Trend Modeling Using News Articles. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1470–1473, New York, NY, USA. ACM.

[Kempe et al., 2003] Kempe, D., Kleinberg, J., and Tardos, E. (2003). Maximizing the spread of influence through a social network. In *KDD '03*.

[Keogh and Kasetty, 2002] Keogh, E. and Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 102–111, New York, NY, USA. ACM.

[Kephart and White, 1993] Kephart, J. O. and White, S. R. (1993). Measuring and modeling computer virus prevalence. *IEEE SP*.

[Kossinets, 2006] Kossinets, G. (2006). Effects of missing data in social networks. *Soc. Netw.*, 28(3):247–268.

[Kumar et al., 2003] Kumar, R., Novak, J., Raghavan, P., and Tomkins, A. (2003). On the bursty evolution of blogspace. In *WWW*, pages 568–576. ACM Press.

[Lakhina et al., 2003] Lakhina, A., Byers, J. W., Crovella, M., and Xie, P. (2003). Sampling biases in ip topology measurements. In *In IEEE INFOCOM*, pages 332–341.

[Lappas et al., 2010] Lappas, T., Terzi, E., Gunopulos, D., and Mannila, H. (2010). Finding effectors in social networks. In *KDD*.

[Leskovec et al., 2006] Leskovec, J., Adamic, L. A., and Huberman, B. A. (2006). The dynamics of viral marketing. In *EC*, pages 228–237.

[Leskovec et al., 2009] Leskovec, J., Backstrom, L., and Kleinberg, J. (2009). Meme tracking and the dynamics of news cycle. In *KDD*.

[Leskovec et al., 2007a] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. S. (2007a). Cost-effective outbreak detection in networks. In *KDD*, pages 420–429.

[Leskovec et al., 2007b] Leskovec, J., McGlohon, M., Faloutsos, C., Glance, N., and Hurst, M. (2007b). Cascading behavior in large blog graphs: Patterns and a model. In *SDM*.

[Levenberg, 1944] Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathmatics*, II(2):164–168.

[Maiya and Berger-Wolf, 2011] Maiya, A. and Berger-Wolf, T. (2011). Benefits of bias: Towards better characterization of network sampling. In *KDD*.

[Matsubara et al., 2012] Matsubara, Y., Sakurai, Y., Prakash, B. A., Li, L., and Faloutsos, C. (2012). Rise and fall patterns of information diffusion: model and implications. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 6–14, New York, NY, USA. ACM.

[Matthews, 1975] Matthews, B. (1975). Comparison of the predicted and observed secondary structure of {T4} phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451.

[Nishiura et al., 2011] Nishiura, H., Chowell, G., and Castillo-Chavez, C. (2011). Did modeling overestimate the transmission potential of pandemic (h1n1-2009)? sample size estimation for post-epidemic seroepidemiological studies. *PLoS ONE*, 6(3):e17908.

[Papalexakis et al., 2013] Papalexakis, E. E., Dumitras, T., Chau, D. H. P., Prakash, B. A., and Faloutsos, C. (2013). Spatio-temporal mining of software adoption & penetration. In *ASONAM*, pages 878–885.

[Pastor-Santorras and Vespignani, 2001] Pastor-Santorras, R. and Vespignani, A. (2001). Epidemic spreading in scale-free networks. *Phys. Rev. Let. 86*, 14.

[Prakash et al., 2011] Prakash, B. A., Chakrabarti, D., Faloutsos, M., Valler, N., and Faloutsos, C. (2011). Threshold conditions for arbitrary cascade models on arbitrary networks. In *ICDM*.

[Prakash et al., 2010] Prakash, B. A., Tong, H., Valler, N., Faloutsos, M., and Faloutsos, C. (2010). Virus propagation on time-varying networks: Theory and immunization algorithms. *ECMLPKDD*.

[Prakash et al., 2012] Prakash, B. A., Vreeken, J., and Faloutsos, C. (2012). Spotting culprits in epidemics: How many and which ones? In *ICDM*. IEEE.

[Rabiner and Juang, 1993] Rabiner, L. R. and Juang, B.-H. (1993). *Fundamentals of speech recognition.* Prentice Hall signal processing series. Prentice Hall.

[Rakthanmanon et al., 2013] Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E. (2013). Data Mining a Trillion Time Series Subsequences Under Dynamic Time Warping. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 3047–3051. AAAI Press.

[Rattanaritnont et al., 2012] Rattanaritnont, G., Toyoda, M., and Kitsuregawa, M. (2012). Characterizing topic-specific hashtag cascade in twitter based on distributions of user influence. In *Proceedings of the 14th Asia-Pacific International Conference on Web Technologies and Applications*, APWeb'12, pages 735–742, Berlin, Heidelberg. Springer-Verlag.

[Richardson and Domingos, 2002] Richardson, M. and Domingos, P. (2002). Mining knowledge-sharing sites for viral marketing.

[Rissanen, 1983] Rissanen, J. (1983). Modeling by shortest data description. *Annals Stat.*, 11(2):416–431.

[Romero et al., 2011] Romero, D. M., Meeder, B., and Kleinberg, J. (2011). Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 695–704, New York, NY, USA. ACM.

[Sadikov et al., 2011] Sadikov, E., Medina, M., Leskovec, J., and Garcia-Molina, H. (2011). Correcting for missing data in information cascades. In *WSDM*. ACM.

[Sakaki et al., 2010] Sakaki, T., Okazaki, M., and Matsuo, Y. (2010). Earthquake shakes Twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

[Salathé et al., 2012] Salathé, M., Bengtsson, L., Bodnar, T. J., Brewer, D. D., Brownstein, J. S., Buckee, C., Campbell, E. M., Cattuto, C., Khandelwal, S., Mabry, P. L., and Vespignani, A. (2012). Digital epidemiology. *PLoS Comput Biol*, 8(7):e1002616.

[Shah and Zaman, 2010] Shah, D. and Zaman, T. (2010). Detecting sources of computer viruses in networks: theory and experiment. In *SIGMETRICS*, pages 203–214.

[Shah and Zaman, 2011] Shah, D. and Zaman, T. (2011). Rumors in a network: Who's the culprit? *IEEE TIT*, 57(8):5163–5181.

[Shen et al., 2014] Shen, H.-W., Wang, D., Chaoming, S., and Barabsi, A.-L. (2014). Modeling and predicting popularity dynamics via reinforced poisson processes. In *The Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI.

[Stewart and Sun, 1990] Stewart, G. W. and Sun, J.-G. (1990). *Matrix Perturbation Theory*. Academic Press.

[Sundareisan et al., 2014] Sundareisan, S., Rao, A., Khan, M. A. S., Ramakrishnan, N., and Prakash, B. A. (2014). Sanstext: Classifying temporal topic dynamics of twitter cascades without tweet text. In *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - 2014 (to appear)*. ACM.

[Tong et al., 2010] Tong, H., Prakash, B. A., Tsourakakis, C. E., Eliassi-Rad, T., Faloutsos, C., and Chau, D. H. (2010). On the vulnerability of large graphs. In *ICDM*.

[Vreeken et al., 2011] Vreeken, J., van Leeuwen, M., and Siebes, A. (2011). Krimp: Mining itemsets that compress. *Data Min. Knowl. Disc.*, 23(1):169–214.

[Wu et al., 2011] Wu, L., Ying, X., Wu, X., and Zhou, Z.-H. (2011). Line orthogonality in adjacency eigenspace with application to community partition. In *IJCAI*.

[Yang and Leskovec, 2011] Yang, J. and Leskovec, J. (2011). Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186. ACM.

Appendix - Division of Hashtags

Division of Hashtags/Keywords among topics

In Section 3.6.1, we used hashtags for different topics for the Popular dataset. We list the topics-wise division of hashtags below in Table A1.

Sports	Tech	Idiom	Flu	Political
#barcelona	#apple	#3palabrasdolorosas	#saude	#caprilespresidente
#deportes	#blackberry	#dudasquenomedejandormir	#salud	#caprilesvenezuelayelmundoestacontigo
#emelec	#fb	#ff	#paciente	#chavez
#f1	#google	#hedicho	#medicina	#copalibertadores
#futbol	#instagram	#mafaldaqoutes	#hospital	#educacionliberadora
#halamadrid	#iphone	#matirasmasudas	#gripe	#elecciones2012
#libertadores	#soundcloud	#siguemeytesigo	#flu	#fraudeenvenezuela
#londres2012	#technologia	#sumapuntos	#fiebre	#guatemala
#millos	#tuitutil	#sumapuntosque	#enfermedad	#hayuncamino
#olimpia	#twitter	#sumapuntossi	#dolores	#news
#realmadrid	#virtualmall		#estres	#noticias
	#youtube			#seguidores
				#tropa
				#veracruz
				#vota

Table A1: Hashtags for Popular Dataset

In Section 3.7.1, we used keywords that belong to different protest types in **Task 1**. We show the division of the keywords below in Table A2.

Non-Violent Other Government Policies	Non-Violent Energy and Resources	Violent Energy and Resources	Non-Violent Other	Violent Other
aeroporto	cantar	cantar	aeroporto	aeroporto
cantor	Carlo	Dutra	Dutra	Carlo
carlo	Dutra	estimativo	exibir	cobrador
existir	existir	existir	marcado	Dutra
friograndedesul	marcha	guarda	marcha	estimativo
guarda	minuto	marcado	matar	exibir
marcha	Rio Grande do Sul	marcha	Rio Grande do Sul	guarda
matar	sociedade	pedra	Sorocaba	marcado
minuto	Sorocaba	torcedor		matar
	torcedor			pedra

Table A2: Keywords for Protest ***Task 1***