EXPLOITATION OF INFORMATION PROPAGATION PATTERNS IN SOCIAL SENSING

BY

MD TANVIR AL AMIN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2017

Urbana, Illinois

ABSTRACT

Online social media presents new opportunity for sensing the physical world. The sensors are essentially human, who share information in the broadcast social media. Such human sensors impose challenges like influence, bias, polarization, and data overload, unseen in the traditional sensor network. This dissertation addresses the aforementioned challenges by exploiting the propagation or prefential attachment patterns of the human sensors to distill a factual view of the events transpiring in the physical world.

Our first contribution explores the correlated errors caused by the dependent sources. When people follow others, they are prone to broadcast information with unknown provenance. We show that using admission control mechanism to select an independent set of sensors improves the quality of reconstruction. The next contribution explores a different kind of correlated error caused by polarization and bias. During events related to conflict or disagreement, people take sides, and take a selective or preferential approach when broadcasting information. For example, a source might be less credible when it shares information conforming to its own bias. We present a maximum-likelihood estimation model to reconstruct the factual information in such cases, given the individual bias of the sources are already known. Our next two contributions relate to modeling polarization and unveiling polarization using maximum-likelihood and matrix factorization based mechanisms. These mechanisms allow us to automate the process of separating polarized content, and obtain a more faithful view of the events being sensed.

Finally, we design and implement 'SocialTrove', a summarization service that continuously execute in the cloud, as a platform to compute the reconstructions at scale. Our contributions have been integrated with 'Apollo Social Sensing Toolkit', which builds a pipeline to collect, summarize, and analyze information from Twitter, and serves more than 40 users.

Dedicated to my parents.

ACKNOWLEDGMENTS

All praise is due to the Almighty, who bestowed me the strength and determination to wade through the stress of graduate life. For almost six years, I have worked toward this dissertation that improves the quality and scalability of summarization services in social sensing. It has been an adventurous journey full of challenges, innovation, accomplishments, occasional failures, learning experiences, and papers!

Many thanks to my advisor, Tarek Abdelzaher, who motivated and inspired me to scientific rigor, high quality writing, and presentation style. Without his help, this thesis would not have been possible. I specially remember many long meetings, discussions, and brainstorming sessions that have been integral to the formation of my thesis. Moroever, his direction has helped me to lead a team of graduate student researchers developing Apollo Social Sensing Toolkit, and acquire leadership skills. We also designed and taught a Real-Time Systems course at UIUC. It gave me opportunity to create teaching material, and advise students in challenging projects. Also thanks to my committee members: Indranil Gupta, Aditya Parameswaran, and Mudhakar Srivatsa, for numerous help, suggestions, and support. Thanks to the University of Illinois Department of Computer Science, and the Graduate College for awarding me Chirag Foundation Graduate Fellowship. It relieved me from financial burden at the beginning of my doctoral program. Research published in this dissertation has been funded by grants from US Army Research Laboratory, DTRA, and National Science Foundation. Thanks to the funding agencies for their immensely helpful grant programs.

My collaborators at IBM Research, and US Army Research Laboratory: Mudhakar Srivatsa, Raghu Ganti, Charu Aggarwal, Lance Kaplan, deserve big thanks for their guidance and advice during my visits. I am grateful for the brainstorming sessions. Special thanks to professors at UIUC: Indranil

Gupta, Philip Brighten Godfrey, Klara Nahrstedt, and Matthew Caesar, who have advised me during course projects and research collaborations. Earliest iterations of an important contribution of my thesis 'SocialTrove' started as a course project with Indranil Gupta. My internships at Google and Facebook enriched me with real-world systems skills and engineering techniques. I thank my hosts Phil Gross, Yi Wang, Rajesh Nishtala, Andrii Grynenko for their guidance. My collaboration with Boleslaw Szymanski at RPI, Reaz Ahmed at UWaterloo was very useful.

I am thankful to Yusuf Sarwar Uddin who helped me quickly bootstrap research in our group. Numerous meetings with Shen Li have been very influential. I am thankful to Dong Wang and Hieu Le for their awesome support and advice during the initial phase of my PhD. I am also thankful to other labmates and collaborators, who helped shape up my research. I can not possibly include all the names here, but I would specially thank Muntasir Rahman, Panindra Seetharamu, Jongdeog Lee, Shuochao Yao, Prasanna Giridhar, Shaohan Hu, Shiguang Wang, Hongwei Wang, Huajie Shao, Yiran Zhao, Hengchang Liu, Lu Su.

I was born and grown up in beautiful Bangladesh. I earned a bachelors and masters degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology. I also had the chance to work there as a Lecturer, and teach courses to undergraduate students. It was one of the most wonderful experiences spanning professional and social aspects of my life, and certainly had a strong influence on making me the person I am today. It was my professors and colleagues there who taught me the fundamentals of Computer Science and Engineering, inspired me to pursue a PhD in Computer Science, and made me aware of the social and ethical responsibilities a scientist should hold. I want to take some time to thank my professors, colleagues, friends, and students there, specially Mohammad Kaykobad, Mostofa Akbar, Reaz Ahmed, Ashikur Rahman, Sohel Rahman, Abu Sayed Md. Latiful Hoque, Eunus Ali, Yusuf Sarwar Uddin, Khaled Mahmud Shahriar, Abul Hassan Samee, Abdullah Adnan, Shahriar Nirjon, Rifat Shahriyar, Sukarna Barua, Nashid Shahriyar, Arup Raton Roy, Tanaeem Moosa, Shihabur Rahman Chowdhury, Debajyoti Mondal, Shahriar Rouf Nafi, Mahbubur Rahman, Muntasir Mashuk, Masruba Tasnim, and many others. I was very fortunate to work with Mosharaf Chowdhury and Shahriar Nirjon, as programming contest teammates; they enriched my algorithmic analysis and problem solving skills, and mentored me during my research. I am thankful to my teachers Abu Bakar Siddiqui, Delwar Hossain, Sushanta Kumar Sarker, and Nimai Chandra Das. I am thankful to Mizanur Rahman and Shafiqur Rahman, who taught me how to teach students! I also worked with fantastic classmates and project partners, who made my student life vibrant and full of learning experiences. I would specially like to thank Helal, Sukarna, Fortune, Ayon, and Romel.

The journey to doctoral thesis is a long one. Without the love, inspiration, and support from my parents I could not complete this journey thousands of miles away from home. I dedicate this dissertation to my father Md Abdul Hai and my mother Khandakar Tohura. Many thanks to my elder brothers Abdullah Al Mahmud, and Shahidullah Al Faruq, whose inspiration and help guided me through tough times. My friends in Bangladesh: Abir, Tanvir, Bipul, and Chisty were integral to many aspects of my life. Moreover, it would be very hard for me to sustain here as a graduate student, without the help of my friends in Urbana-Champaign. To name a few, I am grateful to Ahmed Khurshid, Abul Hassan Samee, Yusuf Uddin, Reaz Mohiuddin, Tuba Yasmin, Shafiul Azam, Shakil Kashem, Imranul Hoque, Muntasir Rahman, Munawar Hafiz, Farhana Ashraf, Ahsan Arefin, Shameem Ahmed, Abdullah Al Nayeem, Anupam Das, Farhadul Hoque, Naheed Ferdous, Priyanka Sarker, Shegufta Ahsan, Persia Aziz, Sajjadur Rahman, Himel Dev, and the list continues. It is my family, friends, and the society that help me traverse life, and give ultimate meaning to me and my contributions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

β_i	Probability of a source to be independent in making claims
f_{ij}	Probability of source i making same or similar claims as source j
τ	Admission threshold
SC	Source-assertion network
S_i	Source i
C_j	Assertion j
SD	Source dependency network
z_j	Factual state of assertion j
y_j	Known polarity of claim j
θ	Vector of parameters associated with EM formulations
a^{kl}	Probability of a source from group k to make claims from group l
Y_i^k	Source i belongs to polarized group k
Z_j^l	Assertion j belongs to polarized group l
A	Matrix of source-assertion network
U	Matrix corresponding to sources, related to factorization
V	Matrix corresponding to assertions, related to factorization
T	Source dependency or influence network
k	Number of polarized groups
λ	Regularization factor
γ	Strength of social dependency factor

α	Step size for gradient descent
$size$	Ensemble size
B	Partitions generated by a specific factorization experiment
τ_{diag}	Distance threshold between partitions of a factorization experiment
τ_{edge}	Distance threshold between two factorization experiments
Δ	Batching interval
$d(\mathbf{u}, \mathbf{v})$	Distance between vectors $\mathbf{u}$ and $\mathbf{v}$
d_q	Query distance threshold
d_c	Threshold of diameter of a set of vectors, decides whether to cluster further

CHAPTER 1

INTRODUCTION

This dissertation addresses the problem of reconstructing independently observable states of the physical world from content shared or corroborated by human sensors in the online social medium. The problem is important because humans possess tremendous capability in terms of assessing different situations, much better than a hardware sensor can. When an important event like celebration, protest, election, disaster, sports, or even a revolution happens in the physical world, people can independently observe it. Increasing prevalence of gadgets with rich sensors, and the use of online social networks for instant information broadcast motivates them to share the observations in the form of text, location, photo, or video. Individuals discussing the events in this way, on average, understand the context. Their behavior reflects their understanding. By monitoring such collective behavior, it is possible to harness the *collective intelligence* of the social medium [1], that can be useful to assess the veracity of the information. Therefore, our goal is to faithfully reconstruct factual information about the events happening in the physical world. This direction unveils new opportunities. It becomes possible to automatically crowd-sense news without professional curation, free from bias or influence. To make our point, Table 1.1 shows evidences of events being reported to Twitter first, much before a traditional news service.

The problem is challenging for many reasons. First, participants are human, and individual participation is mostly voluntary and untrained. Therefore, traditional human errors like mistake, omission, or exaggeration are common. Second, unlike physical world, online platforms tend to have much lower cost of social interactions. They follow a broadcast (or multicast) based information dissemination model. People share both original and corroborated content in the form of text, pictures, or video, and others can easily discover those by following particular people or topics of interest. Therefore,

Table 1.1: Example tweets appearing earlier than news media

Event	Time (CT)	Tweet
Beijing Earthquake	1:37am 12 May 2008	EARTH QUAKE in Beijing? Yup... @keso I felt it too!!
Hudson river plane crush	2:36pm 15 Jan 2009	`http://twitpic.com/135xa` There's a plane in the Hudson. I'm on the ferry going to pick up the people. Crazy
Royal Wedding	5:04am 16 Nov 2010	The Prince of Wales is delighted to announce the engagement of Prince William to Miss Catherine Middleton
Raid on Osama Bin Laden	2:58pm 1 May 2011	Helicopter hovering above Abbottabad at 1AM (is a rare event)
Whitney Houston	6:30pm 11 Feb 2012	My sources say Whitney Houston found dead in Beverly hills hotel.. Not in the news yet
Boston Marathon Bombing	1:50pm 15 Apr 2013	Explosion at coply

it is easier to be partial, be influenced, believe rumors, or be persuaded by bias. Third, the social platforms receive content at a tremendously high rate. Applications would need to utilize machine clusters to run at scale. Moreover, much of this content can be irrelevant to particular question, or not represent observable physical states. In this dissertation, we present algorithms and systems to summarize the factual information at scale, from the large volume of human generated social media content.

Toward the goal of reconstructing the physical world, we identify that the fundamental challenge is to find mechanisms to handle correlated errors in the observations posted by human sources. In this dissertation, we present algorithms to solve this problem in different situations. We observe that people tend to follow others, and are influenced by them. Because of this property, people sharing information about an event may not have independently observed it, rather they are relaying information received from someone else. We present admission control techniques to drop dependent sources from a social-sensing scenario [2]. Further improvements jointly estimate the credibility of the sources and the informations given a social dependency network, using a maximum-likelihood estimation technique [3]. Next we observe that in case of situations involving conflicting interests of multiple opposing groups, the network of information propagation takes a polarized shape. People show preference for particular side of the conflict, and selectively share, omit, or follow information. We argue that the sources become less credible if the information confirms their bias. We show that polarity-aware algorithms can better reconstruct the ground truths [4, 5], given the attachment of the sources is already known. Next, we have developed unsupervised algorithms to model polarization in the social network [6], and automate the process of separating content of different polarities [7]. Using the polarity annotations obtained by our algorithm, we automate the generation of polarized social dependency network, which allows us to develop a real-time news service. To run this service at scale, we developed 'SocialTrove' [8], which utilizes a machine cluster to summarize social content. We have integrated our algorithms and services into 'Apollo Social Sensing Toolkit' [9]. Evaluations have been performed using Twitter as the social network.

The central theme to this dissertation is making use of the *crowd wisdom*. Human sensors have already assessed the events that transpire around them, and expressed their version. We show that taking hints from the selective share and corroboration properties of the sources allow us to reconstruct the observable states of the physical world. We, therefore, state the following:

Thesis Statement *Algorithms to distill facts from social media posts must observe and account for information propagation patterns on the medium. Relying exclusively on analysis of information propagation patterns can indeed distill accurate observations with high probability, even in the absence of (deep) analysis of content.*

1.1 Challenges

In this section, we further explain the problem and the challenges in the context of Twitter.

- Human generated observations are unstructured and contains various forms of conscious or unconscious variabilities. Two different volunteers may take the photo of same important event, but their photos would probably have different angles [10]. Two different sources may tweet about the same event, albeit in slightly different wording [8]. For

Table 1.2: Examples of factual and non-factual tweets

Factual tweets	Non-factual tweets
Hate crime soared to record levels in most areas after #Brexit vote.	Do you know that there may be a third force acting on this #Brexit thing?
After Hurricane #Irene hit Puerto Rico, the streets were so flooded that a shark managed to swim in a street.	I won't cry if Hurricane Irene annihilates the Jersey Shore.
In Egypt, the death toll in the clashes between police and pro-Morsi supporters in Cairo has risen to 34.	Good luck #Egypt! Peacefully Fight for what you know is right! We are thinking of you!
Warren Buffet led Berkshire Hathaway has tripled its holdings in #Apple.	The iPhone 7 charger/headphone situation is one of the worst things that's ever happened to me.

example, `Main street is flooded`, and `There is flood in main street` essentially represents the same state about the physical world.

- People share independently observable events in the social media. Additionally they also post slogans, personal stories, opinions, etc that do not constitute as states of the physical world. Our goal is to faithfully reconstruct the observable states. Table 1.2 shows example of such tweets in the left, and tweets that do not represent factual information on the right. Presence of unrelated information makes the problem harder. Note that tweets on the left can have binary states `True` or `False`.

- Some people are more credible in their reporting and, some are less credible by adding false or fictional information [3]. The social influence among the people also affects what they share [2, 4, 5]. In case of conflicts, dispute, or situations involving multiple parties with contrasting interests, people can become biased and color their observations according to their sides [4]. Table 1.3 shows example of tweets of different polarity on the left and right.

- 'Big Data' is inherent for crowd-sensing. According to Twitter, they receive over 500 million tweets per day [11]. For another example, around 100 million pictures are uploaded to instagram every day. Moreover, the tweets can be viewed as a stream with high arrival rate. The rate of generation of new information is variable depending on the events

Table 1.3: Tweets of different polarities

Pro	Anti
President awarded @jamala title of the Peoples Artist of Ukraine `(Pro-Jamala)`	#Oops Poroshenko accidently confirms on TV that Jamalas #Eurovision song 1944 is the same song "Crimea is ours" from May 2015. @EBU_HQ `(Anti-Jamala)`
Crowds March in Egypt to Protest Morsi Detention. `(Pro-Morsi)`	Amnesty International Egypt: Evidence points to torture carried out by Morsi supporters. `(Anti-Morsi)`
Huge #Brexit benefit is some control of immigration. If you want in, you have to have a job. `(Pro-Brexit)`	If #Brexit is a grand social experiment to see how stupid people can be and how low they can go, the answer is Very. Can we stop now please? `(Anti-Brexit)`
Syrians In #Ghouta Claim Saudi-Supplied Rebels Behind #Chemical Attack `(Pro-Government)`	Syria govt forces carried out coordinated chemical attacks on #Aleppo. Security Council should act `(Anti-Government)`

that are actually happening in the real world. Events like election, disaster, or major sports are likely to generate more involvement from people for a while. As a result, the observations can be highly transient and the value of the information can quickly damp out. It is practical for the applications to have near real-time requirements to act on the observations.

1.2 Contributions

This dissertation addresses the challenges mentioned in section 1.1. In the following sections, we explain the contributions.

1.2.1 Source Dependency in Social Sensing

A key challenge in reconstructing the physical states is selecting an independent set of observations, that truly corroborates a particular event in question. Because human sensors are influenced by others, we observe that presence of non-independent observations rank many of the non-factual or rumor tweets higher. In this work, we explore several simple distance metrics between sources, derived from their social dependency network. Distance may depend on factors such as whether one source is directly connected to another (e.g., one follows the other in Twitter), whether both are connected to a common ancestor (e.g., both follow a common source), or whether both are followed by the same people. By choosing the most dissimilar sources, we show that we can improve the reconstruction of events. This work is described in detail in Chapter 2.

1.2.2 Social Sensing with Polarized Sources

In this work, we develop a polarity aware fact-finder. The fact-finder addresses the problem of reconstructing accurate ground truth from unreliable human observations in polarized scenarios. By polarization, we refer to a situation where different groups of sources hold largely different beliefs that color their interpretation, and hence representation, of events they observe. Hence, multiple competing versions of such events are reported. The goal of our algorithm is to identify versions that are more likely to be consistent with ground truth. We abstract human observers as binary sensors [3] in that each reported observation is either true or false, and make *statistical credibility* assesments solely based on propagation patterns on different observations. Based on the polarity of the assertions, we extend EM-Social [3] algorithm to implement a polarity-aware fact-finder. Evaluations using polarized scenarios crawled using Twitter search API show that in the presence of polarization, our reconstruction tends to align more closely with ground truth in the physical world than the existing algorithms. The algorithm is implemented as an application module in Apollo Social Sensing Toolkit, and used to de-bias the crowd-sensed news feed service. It is described in detail in Chapter 3.

1.2.3 Evaluating Polarization Models in Social Networks

This work develops and evaluates models of information propagation on social media in the presence of polarization, where opinions are divided on issues of contention into multiple, often conflicting, representations of the same events, each reported by one side of the conflict. Multiple models are compared that derive from the hypothesis that individuals propagate more readily information that confirms their beliefs. We use these models to solve the inverse problem; namely, given a set of posts in a conflict scenario, automate their separation into opinions that favor each side, as well as pieces that appear to me more neutral. Specifically, we develop new maximum-likelihood estimation algorithms for separation of polarized Twitter posts. This work is described in detail in Chapter 4.

1.2.4 Unveiling Polarization in Social Networks

We present a matrix factorization based gradient descent algorithm to separate polarized content in social networks. We propose a model for polarized information networks, and show that the presence of polarized groups can be detected by considering dependence among posted observations. We explore different degrees of polarization and compare the quality of separation (of tweets of opposing polarity) across different algorithms, using real traces collected from Twitter. Evaluations using polarized scenarios crawled using Twitter search API show that our algorithm performs much better than using sentiment analysis or veracity analysis to solve the problem. The algorithm has been implemented as an application module in Apollo Social Sensing Toolkit. It is used to separate the assertions in different polarity groups, as an input to the polarity aware fact-finder. This work is described in detail in Chapter 5.

1.2.5 SocialTrove: A Summarization Service for Social Sensing

SocialTrove is a *general-purpose* representative sampling service that reduces redundancy in large data sets. The service allows application designers to specify an *application-specific* distance metric that describes a measure of similarity relevant to this application among data items. Based on that application-specific measure, the service hierarchically clusters incoming data streams in real time, and allows applications to obtain representative samples at arbitrary levels of granularity by returning cluster heads at appropriate levels of the cluster hierarchy. When data are large, if the observations are stored in a cluster-agnostic manner, retrieving a representative summary would require scanning the entire set of observations, thereby communicating with many machines and decreasing throughput. Instead, SocialTrove stores content in a similarity-aware fashion. Apollo Social Sensing Toolkit uses SocialTrove in the underlying data infrastructure level to cluster incoming data objects in an online fashion, and to serve data objects matching to a query, for subsequent consumption by the application modules. Evaluations using Twitter decahose streams show that SocialTrove supports higher query throughput compared to traditional indexing mechanisms, while maintaining a low access latency. SocialTrove is described in detail in Chapter 6.

1.2.6 Apollo Social Sensing Toolkit

The social-sensing and summarization algorithms presented in this dissertation have been integrated with 'Apollo Social Sensing Toolkit', which is a cloud-backed social sensing platform to create, execute, and customize social sensing tasks. consisting of levels (i) Social Sensors, (ii) Data Infrastructure, and (iii) Application Modules, and a runtime system. Apollo is scalable, and uses a distributed architecture to parallelize analytics workload in a machine cluster. Apollo Social Sensing Toolkit is described in detail in Chapter 7.

1.3 Impact

- The research outcomes have been integrated into Apollo Social Sensing Toolkit, a distributed platform for social sensing. Apollo is being used at US Army Research Lab, and for academic research in multiple departments at UIUC, RPI, CUNY, UCSF, UWisc, ND, PSU, and a few other universities.

  Current implementation of Apollo Social Sensing toolkit is deployed in UIUC Green Data Center [12]. There are 20 internal and 20 external users on `apollo3.cs.illinois.edu` and `apollo4.cs.illinois.edu`, regularly utilizing the toolkit for tracking current events and distilling high value content from large amounts of noisy social media content.

- SocialTrove and the social-sensing algorithms developed in this dissertation are also used as infrastructure for data prioritization in the recently proposed NDN stack that makes networks aware of hierarchical data names.

- Algorithms related to this dissertation have been mentioned in books 'Social Sensing: Building Reliable Systems on Unreliable Data' [13], 'Advances in Computer Communications and Networks – From Green, Mobile, Pervasive Networking to Big Data Computing' [14].

- Papers related to this dissertation have been included more than 15 times, in the reading list of graduate-level courses in different universities. The courses include Sensing in Social Spaces, Adaptive Computing Systems, Advanced Distributed Systems, Data-Driven CPS, etc.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 introduces the correlated error in social-sensing caused by the dependent sources. It presents source selection mechanism to select an independent set of sources. Chapter 3 explains the correlated error caused by the presence of polarization and bias in conflict situations. It presents a polarity-aware fact-finder algorithm to uncover the likely truths in the presence of two or more parties with conflicting interests. It requires the content to be separated into different polarities. Therefore, Chapter 4 studies polarization models, and proposes maximum-likelihood estimation algorithms. Chapter 5 presents matrix factorization based algorithms to partition the content into polarized groups. Based on the partitions, the algorithms presented in Chapter 3 is used to automatically generate a news-feed of observable facts about the conflict situations happening in the physical world. Chapter 6 introduces SocialTrove to address the issue of information overload. SocialTrove reduces redundancy from socially sensed observations by hierarchically clustering tweets in an online fashion, and provides a representative and diverse sample to build the information network required by the earlier algorithms. We have integrated the proposed algorithms and the systems into Apollo Social Sensing Toolkit, described in Chapter 7. Finally, we conclude the dissertation in Chapter 8, and explore avenues for future research.

CHAPTER 2

SOURCE DEPENDENCY IN SOCIAL SENSING

This chapter develops algorithms for improved source selection in social sensing applications that exploit social networks (such as Twitter, Flickr, or other mass dissemination networks) for reporting. The collection point in these applications would simply be authorized to view relevant information from participating clients (either by explicit client-side action or by default such as on Twitter). Social networks, therefore, create unprecedented opportunities for the development of sensing applications, where humans act as sensors or sensor operators, simply by posting their observations or measurements on the shared medium. Resulting social sensing applications, for example, can report traffic speed based on GPS data shared by drivers, or determine damage in the aftermath of a natural disaster based on eye-witness reports. A key problem, when dealing with human sources on social media, is the difficulty in ensuring independence of measurements, making it harder to distinguish fact from rumor. This is because observations posted by one source are available to its neighbors in the social network, who may, in-turn, propagate those observations without verifying their correctness, thus creating correlations and bias. A corner-stone of successful social sensing is therefore to ensure an *unbiased sampling* of sources that minimizes dependence between them. This chapter explores the merits of such diversification. It shows that a diversified sampling is advantageous not only in terms of reducing the number of samples but also in improving our ability to correctly estimate the accuracy of data in social sensing.

2.1 Overview

This chapter investigates algorithms for diversifying source selection in social sensing applications. We interpret social sensing broadly to mean the set of applications, where humans act as the sensors or sensor operators. An example application might be a participatory sensing campaign to report locations of offensive graffiti on campus walls, or to identify parking lots that become free of charge after 5pm. Another example might be a damage assessment effort in the aftermath of a natural or man-made disaster, where a group of volunteers (or survivors) survey the damaged area and report problems they see that are in need of attention. Social sensing benefits from the fact that *humans* are the most versatile sensor. This genre of sensing is popularized by the ubiquity of network connectivity offered by cell-phones, and the growing means of information dissemination, thanks to Twitter, Flickr, Facebook, and other social networks.

Compared to applications that exploit well-placed physical sensors, social sensing is prone to a new type of inaccuracy; namely, unknown dependence between sources, which affects data credibility assessment. This dependence arises from the fact that information shared by some sources (say via a social network such as Twitter) can be broadly seen by others, who may in turn report the same information later. Hence, it becomes harder to tell whether information received is independently observed and validated by the source or not. When individual data items are inherently unreliable, one would like to use the degree of corroboration (i.e., how many sources report the same data) as an indication of trustworthiness. For example, one would like to believe an event reported by 100 individuals more than an event reported by a single source. However, if those individuals are simply relaying what they heard from others, then the actual degree of corroboration cannot be readily computed, and sensing becomes prone to rumors and mis-information.

We investigate the effect of diversifying the sources of information on the resulting credibility assessment. We use Twitter as our social network, and collect tweets representing events reported during Egypt unrest (demonstrations in February 2011 that led the resignation of the Egyptian president) and hurricane Irene (one of the few hurricanes that made landfall near New York City in 2011). For credibility assessment, we use a tool developed earlier by the authors that computes a maximum-likelihood estimate of correctness of each tweet based on its degree of corroboration and other factors [15]. In our dataset, some of the tweets relay events that are independently observed by their sources. Others are simply relayed tweets. Note that, while Twitter offers an automatic relay function called “re-tweet”, there is nothing to force individuals to use it when repeating information they heard from others. It is perfectly possible to originate tweets with similar content to ones received without using the re-tweet function. In this case, information is lost on whether content is independent or not.

While it is generally impossible to tell whether or not content of two similar tweets was independently observed, our premise is that by analyzing the social network of sources, we can identify those that are "close" and those that are "not close". By using more diversified sources, we can increase the odds that the chosen sources offer independent observations, and thus lower our susceptibility to rumors and bad information.

We explore several simple *distance metrics* between sources, derived from their social network. Distance may depend on factors such as whether one source is directly connected to another (e.g., one *follows* the other in Twitter lingo), whether both are connected to a common ancestor (e.g., both follow a common source), or whether both are followed by the same people. By choosing the most dis-similar sources, according to these metrics, we show that we can indeed suppress more rumors and chain-tweets. The impact of different distance metrics on improving credibility assessment of reported social sensing data is compared.

The rest of this chapter is organized as follows. Section 2.5 describes earlier work done in field of source selection and fact-finding. Section 2.2 formulates our source selection problem and proposes a set of source selection schemes that diversify the sources admitted for purposes of data collection. Evaluation results demonstrating the effect of source selection on credibility assessment of collected data are presented in Section 2.4 followed by a summary in Section 2.6.

2.2 Source Selection in Social Sensing

Data in social sensing applications that exploit social networks (e.g., Twitter) can be polluted by users who report events that are not experienced or verified by themselves. This is because individuals are able to reproduce claims that they heard from others. We argue that if information can be collected from a diverse set of sources who have a weak "social" connection between them, there is a higher chance that the information collected thereby would be more independent, allowing a more informed judgment to be made regarding its reliability. In the following, we use the terms users, sources and nodes as well as the terms tweets, feeds, claims and observations interchangeably.

2.2.1 Online User Social Graph and Source Dependence

In an online community platform or online social network, each user maintains a virtual relationship with a set of other users. This relationship entails some degree of information sharing. For example, on YouTube, a user may subscribe for videos posted by another user so that the former gets a notification when the later uploads a new video. In Facebook, there is an explicit friend relationship and a membership of a fan-page of another well-known user. Google$^+$ has more granularity like friends, family members, acquaintances, and other groups, called circles. In this chapter, we consider a Twitter-based social sensing application, which allows a *follower-followee* relation. A user following another user means that the former intends to receive the posts made by the latter. We say that if user i follows user j, i is the follower and j is the followee. In Twitter, a user can arbitrarily choose which other users to follow, although the converse is not true. That is, a person can not make another user follow them (a person can, however, block another user from following).

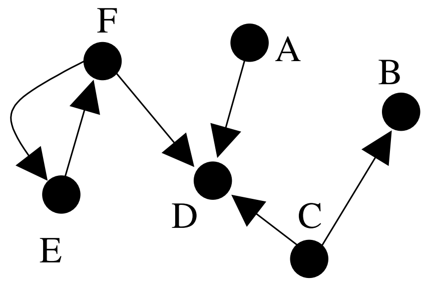


Figure 2.1: A social graph of Twitter uesrs. A directed edge means which source follows which.

We leverage this relationship in Twitter to form a *social graph* among users. We represent each user by a vertex in the graph. A directed edge from one vertex to another denotes that the latter follows the former. We use the notation $i \to j$ to denote an edge in the graph (meaning that user i follows user j). Sometimes, a user may not directly follow another, but can follow transitively via a set of intermediate followees. We refer to this as a follow chain. We use $i \rightarrow^k j$ to denote such a chain with k edges in between. Obviously, $i \rightarrow j = i \rightarrow^1 j$. If i follows j via more than one path, $i \rightarrow^k j$ denotes the one with the least number of hops. We also use $F(i)$ to denote the set of users that a node i follows, that is, the set of followees of node i.

It is reasonable to argue that if source i directly follows source j, reports posted by j would be visible to i, making the information posted by i potentially not original. Another possibility could be that both source i and j have another source in common that both of them follow (i.e., they have a common followee). In that case, the common followee may impact both of them, making their observations mutually dependent. In order to extract reliable information from user-generated tweets, our intention is to gather tweets from *independent* sources to maximize the odds of originality of the information (or equivalently minimize the chance that these users influenced one another). The question is how to reduce potential dependence among users as a given the follower-followee relationships between them. In the following, we formulate this source selection problem.

2.2.2 Source Selection Problem Formulation

We construct a *dependence graph* consisting of sources as vertices and directed edges between vertices as an indication whether or not a source is potentially dependent on another source (e.g., receives their tweets). Weights assigned to edges reflect the degree to which such influence can happen. These weights depend on the characteristics of the social network and the underlying relationship among sources in the social graph. In the context of Twitter, we simply use the follow relationship between sources. If we consider the follow relationship to be the only way sources could be dependent, the proposed dependence graph is identical to the Twitter social graph itself. In general, it is reasonable to assume that other forms of dependence may also exist.

Let $\mathcal{G} = (V, E)$ be the dependence graph, where an edge ij indicates source i is potentially dependent on j. Each edge ij is assigned a *dependence score*, f_{ij}, that estimates the probability of such dependence. That is, with probability f_{ij}, source i could make the same or similar claims as source j. Many factors affect these dependence scores. For example, when a source directly follows another source, it is more dependent on its followee than a source that follows the same followee via a longer follow chain. The number of common followees between a pair sources can also be an indication of dependence between them. If a given pair of nodes have a large number of common followees, they are prone to be more dependent than a pair that have fewer common followees or no followees at all. Whatever the cause of dependence between sources is—that we describe in the subsequent subsection in more detail—we aim to choose a subset of sources that have the least amount of dependence among them.

In the rest of the chapter, we re-draw the dependence graph, $\mathcal{G}$, as a complete graph with transitive dependencies collapsed into a single edge. Hence, f_{ij} exists for every pair of sources i and j (f_{ij}, and is zero only if no influence exists between them. We are interested in estimating the probability that a source makes an *independent* claim, when its claims can be potentially influenced by those made by others. We define an overall *independence score* for each source that gives the probability that it is *not* influenced by other sources in making a claim. This score, denoted by $\beta(i)$ for source i, can be approximated as:

$$\begin{aligned}\beta(i) &= P[i \text{ is independent in making claims}] \\ &= \prod_{j=1}^{n} P[i \text{ is not dependent on } j] \\ &= \prod_{j=1}^{n} (1 - f_{ij}) \qquad (2.1)\end{aligned}$$

One important property of the independence score (that we shall henceforth refer to as the β-score) is that a source cannot have this score in isolation. It is rather a functional form of dependence on other sources. From the definition, we observe that $\beta(i) = 1$ means that source i is absolutely independent (not dependent on any other sources in consideration). We also notice that the β-score declines for a source if the source is influenced by more other sources. To diversify the collection of sources, we consider only a subset of sources whose sum of independence scores is maximum subject to the constraint that no individual source has an independence score below a certain threshold. Let this threshold be τ. That is, we want to compute the subset of selected sources $S \subseteq V$ that maximizes the sum of β-scores. Therefore, we have:

$$\max \quad \sum_{i \in S} \prod_{j \in S} (1 - f_{ij}) \tag{2.2}$$

$$\text{s.t.} \quad \prod_{j \in S} (1 - f_{ij}) \geq \tau, \forall i \in S \tag{2.3}$$

Note that, individual sources can also have an *influence* factor associated with them that can be inferred from the number of followers. If a source has many followers, it may mean that this source produces observations that other users find reliable. This is a source ranking problem and has been addressed in prior work. In this chapter, we do not address source ranking. Instead, we verify the promise that *diversifying* the sources can improve the performance of a subsequent ranking algorithm.

The optimization problem stated by Equation (2.2) can be shown to be an IP (Integer Programming) problem, and is therefore NP-Hard. We can use a greedy approximation by building the solution incrementally. The greedy algorithm assumes that all candidate sources are available apriori so that the source selection can pick a subset of them. Sometimes the set of sources is not known beforehand. Rather, new sources are discovered as they arrive incrementally. In that case, an *online* algorithm seems more appropriate.

In this chapter, we consider a system where a stream of tweets arrives at a processing station. Our source selection scheme acts as an admission controller that needs to make an online assessment regarding whether or not a new source is to be selected based on the relationships it has with respect to other sources selected earlier. If the source is selected, all tweets that originate from that source are admitted, and will be passed to the actual processing engine as they arrive. Otherwise, the source is not admitted and all tweets from that source will be dropped on arrival. Hence, our online admission controller is a simple gate that admits tweets based on which source they are coming from. An advantage of admission control as described above is that it is fast and easy. In particular, it is based on sources and not on the content of tweets. In principle, better admission controllers can consider content as well, but they will be significantly slower. Hence, in this chapter, we restrict our notion of data sampling to the granularity of entire sources, making it a source selection scheme. In the following, we compare performance of different source selection schemes.

2.3 Online Admission Control

The online admission controller makes a decision regarding each tweet upon its arrival to the system. If the source associated with the tweet is already admitted, the tweet is passed to the next step. If not, the candidacy of the source is evaluated in terms of how independent this source is with respect to the earlier admitted sources. The admission controller computes the β-score of the incoming source and then accepts it only if its β-score remains above an admission threshold, τ. Otherwise, it is denied. Let S be the set of sources that have been admitted so far. The source denial rule, as per Equation (2.3), is:

$$\textit{Denial rule for source } i\textit{:} \quad \prod_{j \in S} (1 - f_{ij}) < \tau \tag{2.4}$$

For a certain definition of f_{ij} and the associated admission threshold, τ, we can formulate a set of different admission controllers as we describe in the following. In all admission control schemes, if not otherwise stated, admission decisions are final: once admitted, a source is not revoked from the admitted set. In the following discussion, let i be the source who is seeking admission.

1. No direct follower:

$$\begin{aligned} f_{ij} &= \begin{cases} 1 & \text{if } i \text{ follows } j \\ 0 & \text{otherwise} \end{cases} \\ \tau &= 1 \end{aligned}$$

Deny, if the source is a direct follower of another admitted source. Recall that if source i follows any of the earlier admitted sources in S, that is, for some $j \in S$, $f_{ij} = 1$, it leads to $\beta(i) = 0$, thus violating the admission condition.

2. No direct follower as well as no common followee:

$$\begin{aligned} f_{ij} =& \begin{cases} 1 & \text{if } i \to j \vee F(i) \cap F(j) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \\ \tau =& 1 \end{aligned}$$

Deny, if the source directly follows someone in the set or has at least one followee in common with another admitted source.

3. No descendants:

$$\begin{aligned} f_{ij} =& \begin{cases} p^k & \text{if } i \to^k j, 0 < p < 1 \\ 0 & \text{otherwise} \end{cases} \\ \tau =& 1 \end{aligned}$$

Deny, if the source is a follower of another admitted source possibly via a set of intermediate followees.

4. No more than k followees:

$$\begin{aligned} f_{ij} =& \begin{cases} p & \text{if } i \text{ follows } j \\ 0 & \text{otherwise} \end{cases} \\ \tau =& (1-p)^k \end{aligned}$$

for some constant $p, 0 < p < 1$.
Deny, if a source is a direct follower of more than k admitted sources.

No common followee with more than k sources:

$$\begin{aligned} f_{ij} =& \begin{cases} p & \text{if } F(i) \cap F(j) \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \\ \tau =& (1-p)^k \end{aligned}$$

Deny, if a source has at least one followee in common with at least k other admitted sources.

No more than k common followees:

$$f_{ij} = \begin{cases} 1 & \text{if } |F(i) \cap F(j)| \geq k \\ 0 & \text{otherwise} \end{cases}$$
$$\tau = 1$$

Deny, if a source has at least k followees in common with another admitted source.

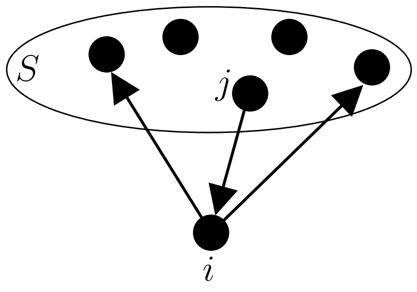


Figure 2.2: Admission control scheme. Assuming the same dependence score f between pair of sources, $\beta(i) = (1-f)^2$ and $\beta(j)$ is declined by a factor $(1-f)$.

4. β-controller: This controller selects sources that progressively improve the sum of β-scores as per Equation (2.2), while satisfying the constraint (2.3) for each individual admitted source. This controller considers transitive *follower-followee* relationships among sources and defines the following dependence function:

$$f_{ij} = \begin{cases} p^k & \text{if } i \rightarrow^k j \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

for some constant $p < 1$. We used, $p = \frac{1}{2}$.

Let $B(S)$ be the sum of β-scores of admitted sources. In other words, $B(S) = \sum_{j \in S} \beta(j)$. Let i be the new source. The scheme computes:

$$\beta'(i) = \prod_{j \in S \cup \{i\}} (1 - f_{ij}), \forall i \in S \cup \{i\} \tag{2.6}$$

$$B(S) = \sum_{j \in S} \beta(j) \tag{2.7}$$

$$B'(S) = \sum_{j \in S \cup \{i\}} \beta'(j) \tag{2.8}$$

The scheme then admits i only if $\beta'(i) \geq \tau$ and $B'(S) > B(S)$. Note that, when a new source is admitted, the scores of some earlier admitted sources may decrease (this is because they may be followers of this newly admitted source). Upon admittance of the new source, those scores are updated. Among possible choices, we consider two versions of β-controllers, with $\tau = 0, 1$. The one with $\tau = 0$ does not check individual β-scores but admits sources as long as they improve $B(S)$, whereas $\tau = 1$ denies a new source if it has any link with any of the earlier admitted sources (i.e., $\beta < 1$) and also fails to improve $B(S)$.

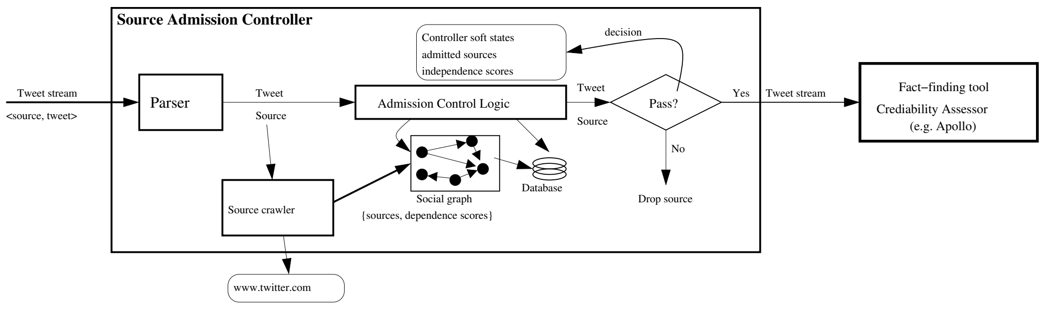


Figure 2.3: Schematic model of the admission controller with Apollo's pipeline.

2.3.1 Complexity of Admission Controllers

Once accepted, a source is not rejected later, and vice versa. So the decision about a particular source can be stored in a hash table. Once a source arrives, whether that source had already been explored or not, can be checked in $O(1)$ time and the stored decision can be used. If the incoming node is previously unexplored, the admission controller needs to decide about it. For the first three controllers, this decision requires $O(out(i))$ computations, where $out(i)$ is the outdegree of i in the dependence graph. The method is simply to check whether any of those outdegree vertices belong to the set of already decided sources. β-controllers consider ingoing edges also, so they take $O(out(i) + in(i))$ computation steps per admission decision. In short, the admission cost of a new source is at worst in the order of its degree in the dependency graph. But it is O(1) lookup for all the tweets that come from it thereafter. Moreover, social graphs tend to have a power law degree distribution, so very few nodes will require a high computation time for the decision.

Once admitted, a source is not discarded later, so the decision about a particular source can be cached in a hash table. Once a tweet arrives, whether that node had already been explored or not, can be checked in $O(1)$ time and that cached decision can be used. If the incoming node s_i is previously unexplored, the admission controller needs to compute $\beta(s_i)$ which takes $O(|TW(s_i)| + |TR(s_i)|)$ computation time, where $TW(s_i)$ is the set of those edges in the transitive closure of the social graph which originate from s_i, and $TR(s_i)$ is the set of those edges in the transitive closure of the social graph which end i s_i. So, computational complexity for taking a decision about a new source depends on the sum of its indegree and outdegree in the transitive closure of the social graph. For n sources, the total time for decision becomes $O(\sum_{i=1}^{n}[|TW(s_i)| + |TR(s_i)|]) = O(e_t)$ according to handshaking lemma of graph theory, where e_t is the number of edges in the transitive closure of the social graph. However, when $\tau = 1$, this computation can be done in the original social graph rather than its transitive closure. Over time, as the tweets arrive, total time spent for decision in that case becomes $O(e)$, where e is the number of edges in the social graph. For a stream of t tweets ($t > n$), the total time spent is $O((t - n) * O(1) + O(e)) = O(t + e)$. So, average time spent per tweet is $O(\frac{e}{t})$. In a running system like Twitter, there will always be some new users joining, but the fraction of new users will be insignificant. After all the existing sources have been explored, total number of admitted sources will remain nearly constant. Each tweeter user in general generates a lot of tweets; so, as the number of tweet t increases, effective complexity becomes $O(1)$, which is the time needed for lookup in the hash table.

2.3.2 System Design and Implementation

Our admission controller is used in association with a fact-finding tool called Apollo [16]. It receives a stream of tweets from which it derives credibility scores of sources and claims (i.e., tweets) using an expectation-maximization (EM) technique [15]. Once the iterations converge, Apollo outputs the top credible sources and top credible tweets made by those sources.

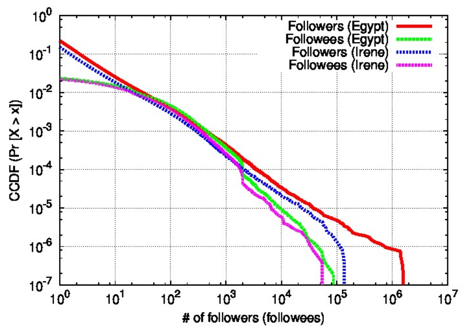
Apollo assumes that all sources are independent. Our admission controller filters out tweets before they are fed into the Apollo engine such that the surviving ones are more likely to be independent indeed. Figure 2.3 shows the design of the whole pipeline.

The pipeline is implemented as a set of stages processing a stream of tweets in JSON format. A parser extracts various information components from each tweet entry. There are two main components to extract: user information, usually a unique Id and screen name of the source who tweeted the current tweet, and the tweet string itself. The admission controller maintains a source information base that is updated as it encounters new sources. Upon encountering a new user, the "source crawler" contacts to the Twitter server and collects the Twitter record of that particular user, which includes additional information such as the user's screen name, location, profile url, the number of followers and the number and identities of followees this user has. If not otherwise restricted by any privacy setting for this user, the crawler also collects the complete list of followees (i.e., the other users that this user follows in Twitter's user space). As more and more sources are encountered, a social graph among users is constructed. This social graph is stored in a database and is an essential element for source admission control.

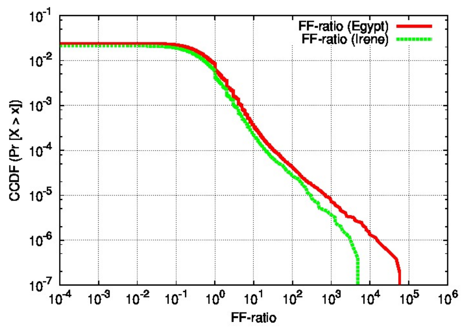
An admission controller logic unit implements the admission control rules described in Section 2.3. It computes dependence scores between pairs of sources and admits new sources as permitted by the corresponding admission rules. When an incoming source is admitted, the associated tweet entry is passed to the next processing stage within Apollo.

2.4 Evaluation

We evaluated our source selection schemes using two Twitter datasets. One is for Egypt unrest, collected in February 2011, during a massive public uprising in Cairo. Another dataset is from hurricane Irene, one of the costliest hurricanes on record in the Northeastern United States, collected in August 2011, when it made landfall near New York City. In both cases, we



(a)



(b)

Figure 2.4: (a) Complementary distribution (CCDF) of follower and followee count per user, (b) CCDF of ff-ratio per user, in Egypt dataset.

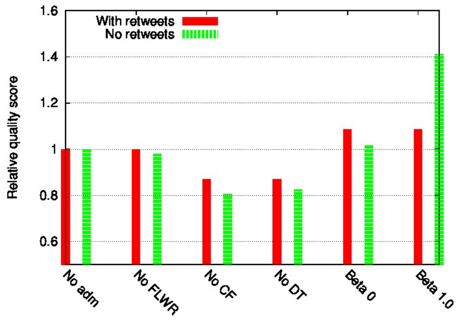
collected hundreds of thousands of tweets posted by users as the events unfolded during those times. The datasets are summarized in Table 2.1. We were interested in extracting a smaller subset of high quality reports on the progress of these events as computed by the find-finder engine, Apollo. The question is whether a significant improvement occurs in distilling the most important tweets due to the source diversification process described earlier in this chapter.

Table 2.1: Statistics of two datasets

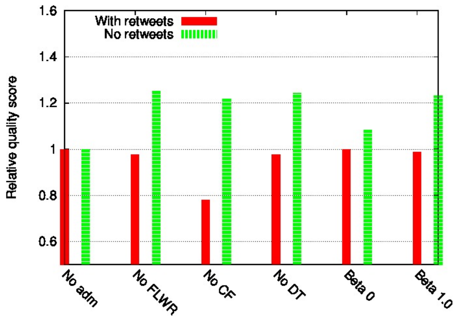
Dataset	Egypt unrest	Hurricane Irene
Time duration	18 days	$\approx$ 7 days
# of tweets	1,873,613	387,827
# of users crawled	5,285,160	2,510,316
# of users actually twitted	305,240	261,482
# of follower-followee links	10,490,098	3,902,713

In Twitter, both the number of followers and followees per user observe a power law distribution (i.e., heavy tail distribution). More precisely, there exists a very large number of users who have only a few followers, whereas a few sources may have an extremely large number of followers. The same is true for the number of followees. Figure 2.4a plots the complementary cumulative distribution (CCDF) of the number of followers and followees per source across all users recorded in the Egypt dataset and Irene dataset. The CCDF depicts what fraction of users have the number of followers or followees greater than the corresponding value on the x-axis.

In Figure 2.4a, we observe that the number of followers per user, in both



(a) Egypt dataset



(b) Irene dataset

Figure 2.5: Relative quality scores across different admission control schemes.

datasets, is larger than the number of followees per user. Hence, the followee curve in the plot lies beneath the follower curve. Clearly, when the entire social network is considered, the totals will be the same. However, in our data collection, we see only those who tweet. Hence, we invariably sample the subset of more active users, creating the imbalance between follower and followee counts. We plot the ratio of follower count to followee count (*ff-ratio*) in Figure 2.4b. We see that in both datasets only a very small fraction of users have non-zero follower and followee count (1.7% for Egypt dataset and 2.4% for Irene dataset). More than half of these have more followers than followees (ff-ratio > 1). Very few users have an order of magnitude more followers than followees. These are mostly popular entities, such as celebrities, international organizations, and news media.

The goal of the evaluation was to answer two related questions: First, what is the impact of source diversification on data credibility assessment when the social network is well-connected? Second, what is the impact if the social network is very sparse? Since both of our datasets were sparse, to answer the first question, we artificially removed from one of the datasets (namely, the Egypt dataset) all users who did not have any links (together with their tweets). Tweets from the remaining sources were considered. The Irene dataset was kept as is, and used to answer the second question (i.e., demonstrate the impact of our admission controllers in the case when the underlying social network is sparse). Conceptually, our admission controllers, by their very design, exploit links between sources for diversification. Hence, in the absence of many links, their effect should not be pronounced.

Next, we present results from various admission controllers that we described in Section 2.3. We compare no admission control to several admission control schemes; namely, *no follower* (No FLWR), *no common followee* (No CF) and *no descendant* (No DT), and *β-controller* (Beta). We evaluate the improvement, attained by these admission controllers, in Apollo's ability to rank tweets. Performance was assessed by the fraction of top-ranked tweets that were "good" in that they reported "relevant and true facts". To identify relevant and true facts, we asked volunteers to grade the top-ranked tweets by placing them in one of the following two categories:

- *Fact*: A claim that describes a physical event that is generally observable by many individuals *independently* and can be corroborated by sources external to the experiment (e.g., news media).

- *Other:* An expression of one's personal feeling, experiences, or sentiments. Remarks that cannot be corroborated. Unrelated random text and less meaningful tweets.

Apollo was run with each of the admission control options on consecutive windows of data, called *epochs*, and used to return the top 5 tweets from each epoch. For the Egypt dataset, we divided the timeline into 18 epochs, and collected the top 5 tweets from each, resulting in a total of 90 tweets graded per experiment (i.e., per admission control option). For the Irene dataset, we choose 150 tweets (top 5 tweets from each of 30 epochs). We built a web interface, where volunteers could grade these tweets without revealing which ones were selected in which experiment (i.e., with which admission controller). Once tweets were graded, a *quality score* for each experiment was computed denoting the fraction of tweets that have been identified as *fact*. If more than one volunteer graded the same results and differed in classifying a tweet, we used the average score.

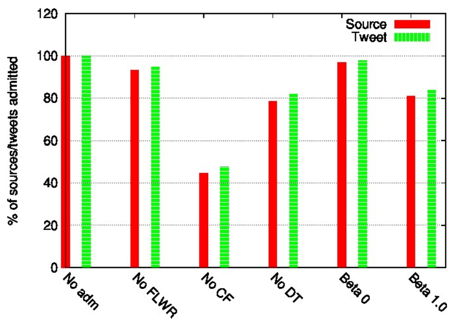
Figure 2.5 presents the *relative* quality scores of various admission control schemes with respect to the "no admission control" scheme. We present results with two Apollo options, i) with retweets and ii) without retweets. The former option has no effect on the dataset. The latter option discards all tweets that are explicitly tagged by their sources as "retweets" (i.e., a repeat of tweets posted earlier). This discarding is in addition to tweets already dropped by admission control. We observe that, in both datasets, experiments with no-retweet option produce higher quality scores. This is because they eliminate "chain-tweeting", where users relay sentiments and opinions of others. In the absence of such re-tweets, highly corroborated tweets (that percolate to the top) more often reflect situations that independently prompted the respective individuals to report. Such a synchronized reaction typically reflects a higher importance of the reported situation.

In our plots, "Beta 1.0" stands for β-controller with threshold, $\tau = 1.0$. We observe that in general β-controllers result in better quality scores. This observation supports our hypothesis that diversifying sources does indeed improve the quality of information distillation. In contrast, the performance of the other admission controllers is mixed. For the Egypt dataset, simple admission heuristics such as 'no follower', 'no common followee' and 'no descendant' generally offer slightly lower quality scores compared to no admission control. For the Irene dataset, they produce lower scores when retweets are included but higher scores in the no-retweets case.

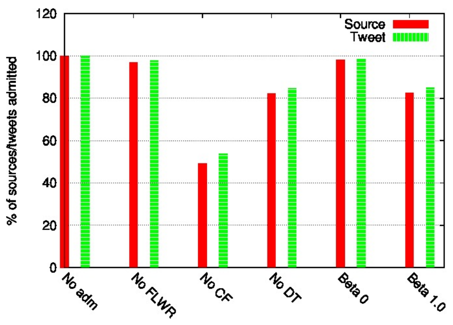
Note that, since the Irene dataset has limited connectivity, β-controllers have a more limited impact. They performs similarly to the no admission control case for the with-retweets option, and slightly better for the no-retweets option. This is expected, since sparse social networks offer little opportunities for further diversification.

With retweets option, however, there are a couple of small discrepancies. For example, "Beta 0" improves the result, but "Beta 0.5" does not, again "Beta 1.0" does. While dropping admitted sources has positive improvements for "Beta 0.5", the same does to hold for "Beta 1.0". These discrepancies in the reported results are mainly due to the fact that in all experiments we had a single stream of tweets, which did not allow us to repeat these experiments for a set of different tweet streams but on the same experiment condition. We could have been able to make more generalizable comments on results if we had more tunable experiment setups. We will accommodate this in our future work.

For Irene dataset, we were needed to run a couple of different other admission controllers, due a particular issue with the dataset. The Irene dataset has a very small connected social network: most sources have no edges with others. This makes a very large number of sources to be *trivially* admitted by the admission controllers we described earlier. To circumvent this, we incorporate individual attributes of sources in addition to their pair-wise relationships in admitting sources. We specifically used *ff-ratio* (which we

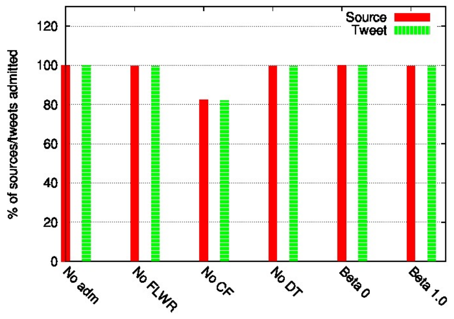


(a) With retweets

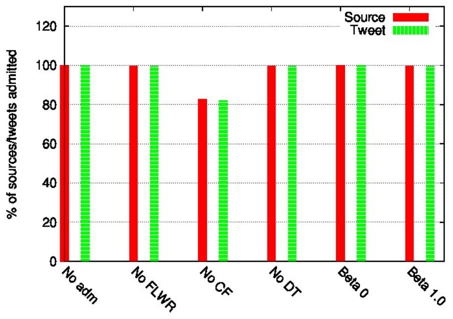


(b) Without retweets

Figure 2.6: Admission controller statistics for different admission schemes (Egypt dataset).



(a) With retweets



(b) Without retweets

Figure 2.7: Admission controller statistics for different admission schemes (Irene dataset).

refer to as α factor). Recall that ff-ratio specifies the ratio of follower count to followee count of a user. Generally, sources with higher α scores prone to be more independent than those with smaller values. When α score is considered, the admitted sources not only need to satisfy β-constraint, but their α scores need to be higher than a certain threshold. That gives a variant of "Alpha Beta" admission controllers. We show the results in Figure 2.5b.

Figure 2.6 and Figure 2.7 show the percentage of sources and tweets that each admission controller admits for the two datasets. It is apparent that some admission schemes are more pessimistic in the sense that they admit fewer sources (and tweets thereby) than others. For the Egypt dataset, on an average, 15–20% tweets are pruned by the admission controllers. For the Irene dataset, however, admission rates across various admission controllers are much higher because of the disconnected nature of the underlying social network.

2.5 Related Work

Social sensing has received much attention in recent years [17]. This is due to the large proliferation of devices with sensing and communication capabilities in the possession of average individuals, as well as the availability of ubiquitous and real-time data sharing opportunities via mobile phones with network connection and via social networking sites (i.e., Twitter). A few early applications include CarTel [18], a vehicular data collection and sharing system, BikeNet [19], an application allowing bikers to share their biking experiences on different trails, PhotoNet [20], a data collection service for pictures from disaster scenes, CenWits [21], a search and rescue scheme for hikers, CabSense [22], a participatory sensing application using taxi car fleets, Urban sensing [23, 24], and ImageScape [25], an application for sharing diet experiences. It has been suggested [17] that people-centric genre of sensing should also cover humans as the sensors themselves, as opposed to being sensor carriers and operators. There are many sensing challenges in human context such as accommodating energy constraints of mobile sensing devices [26], protecting the privacy of participants [27], and promoting social interactions in different environments [28].

Srivastava et al. [17] suggested that humans are the most versatile sensors. One consequent problem lies in the decreased quality of collected data, since humans are not as reliable as well-calibrated sensors. Moreover, there are new challenges that stem from the fact that observations may propagate among such "sensors", leading to correlated noise and bias. A significant amount of literature therefore deals with extracting useful information from a vast pool of unreliable data.

Prior to the emergence of social sensing, much of that work was done in machine learning and data mining. The techniques were inspired by generalizations of Google's PageRank [29], Hubs and Authorities [30], Salsa [31], Hub Synthesis [32] etc. These algorithms are designed to find authoritative web-page to answer a web search query. However, it was noted that authority does not always translate to accuracy, and data crawled from authoritative web sources may contain conflicting information, wrong information, or incomplete information. Therefore techniques were proposed that represent information by a source-claim network [33, 34] that tells who said what. The basic idea is that the belief in correctness of a claim is computed as the sum of trustworthiness of sources who made that claim, and the trustworthiness of a source is, in turn, obtained from the beliefs in correctness of the claims it makes. An iterative algorithm then tries to reason on this graph to extract the most trustworthy information given the degree of corroboration and inferred source reliability. Generally these techniques are called *fact-finders*, a class of iterative algorithms that jointly infer credibility of claims as well as trustworthiness of sources. Notable fact-finding schemes include TruthFinder [35], 3-Estimates [36], and AccuVote [37, 38].

Several extensions were developed to improve fact-finding results, such as incorporating prior knowledge [39, 40], and accounting for the source's expertise in different topics [41]. A maximum likelihood estimation approach was developed that is the first to compute an *optimal solution* to the credibility assessment problem [15]. The solution is optimal in the sense that the resulting assignment of correctness values to claims and sources is the one of maximum likelihood. A confidence interval was also computed to describe the quality of the maximum-likelihood hypothesis [42]. In this chapter, we attempt to improve quality of fact-finding results by improving its input through increasing the odds of independence between the selected sources. Directly considering the source dependencies in the EM formulation resulted in EM-Social algorithm [3]. Further improvements resulted in [4, 43, 44].

In the context of fact-finders, Qi et al. [45] consider dependency between the sources and assess credibility of the sources at a group level, where a group is formed by inferring latent dependency structure among the sources. Vydiswaran et al. [46] propose fact-finders in the context of free-text claims as opposed to structured information. Lehmann et al. [47] propose DeFacto – a fact-validation mechansim for RDF triples. Yu et al. [48] propose multi-dimensional fact-finding framework using slot filling validation technique. Li et al. [49] propose confidence-aware algorithm when there is a long-tail distribution of the sources to the claims. They incorporate signals from multiple sources, systems, and evidences, using a knowledge graph, and combine with multi-layer linguistic analysis of the content. Cao et al [50] address relative accuracy in the absence of true values. Sensoy et al. [51] propose frameworks based on Description Logic and Dempster-Shafer theory to reason about uncertain information obtained from different sources. Pal et al. [52] address the problem of integrating unreliable information over time. They model the real-world history as hidden semi-Markovian process (HSMM), the unreliable sources as observations of the hidden states, and propose Gibbs Sampling and EM algorithms to jointly infer the history and their mapping to the sources. Li et al. [53] propose optimization-based and MAP-based algorithms to estimate credibility from evolving data. Zhi et al. [54] propose EM algorithms to identify the existence of true answer to particular questions in slot filling tasks. These algorithms primarily work on structured information, while the data received from social media is mostly free-form. The lack of addressing for social factors such as influence or bias required new solutions.

The problem of source dependency has been addressed to improve the accuracy of information fused from multiple web-sites and data sources [33,36, 55–57]. Dong et al. [38] consider source dependency by the complex copying relationship between sources. The problem is considered in the context of fact-finders in [58]. Use of source-dependency to account for conflicting data was presented by Dong et al. [37], Blanco et al. [59]. Liu et al. [60] present Solaris, an online data fusion system and computes expected, maximum, and minimum probabilities of a value to be true. Sarma et al. [61] consider the problem from the perspective of cost minimization and coverage maximization. In their model, correlated errors from the dependent sources are not explicitly considered. Zhao et al. [62] propose Latent Truth Model. They model the sources using two types of errors, namely false positives and false negatives, and merge multi-valued attribute types using a Bayesian approach. The problem of estimating real-valued variables in the presence of conflicts has been addressed in [63]. Li et al. [64] propose iterative mechanisms using loss functions to estimate source reliability in the presence of conflicting information. Pasternack and Roth [65] propose Latent Credibility Analysis, a principled mechanism to identify credibilities in the presence of conflict. Dong et al. [66] focus on providing explanation for the fusion for the purpose of understanding the output. Pochampally et al. [67] propose fusion techniques that consider source quality and data correlations. Li et al. [68] propose scalable methods for copy detection.

Problems discussed in these works are related to our general problem of finding facts, however the solutions are not applicable. The mechanisms assume the presence of formatted and structured knowledge triplets, on which the possible copy and level of dependency is estimated. They consider the source dependency by unveiling possible data copies. They generate a relatively independent set of high quality sources to perform data fusion using voting. Earlier works [15] observed that voting is not optimal for finding facts from social media posts. The reason is the unknown odds of different sources to post correct information. To estimate the source models optimally consistent with the observations, we use Expectation-Maximization (EM) based formulations.

Additionally, the problem of detecting copied information is relevant in fusing information from different web based information sources. This is because often a particular information or a subset gets copied, and sometimes there are multiple versions with conflicting claims. In case with Twitter or similar social media analysis, posts are often small and are often about events transpiring in the physical world that quickly change. It is easy to access social media, and it is also easy to publish. Therefore, detecting possible copy of a particular post is less relevant, rather the aggregate dependency between the sources is more important for the analysis. The timestamp of the posts, retweet information, and the similarity in expressed information is used to estimate social dependency network.

The problem of information source selection has also been discussed in web data retrieval [69–72] and in query sampling [73–75]. Dai et al. [76] utilize the mutual dependency between sources to identify anomaly in the context of users-rating-books and users-clicking-advertisement scenario. These efforts reason on the attributes of sources as well as the content that those sources generate. In contrast, ours is a content-agnostic approach that relies only on relationships among sources.

Gupta et al. [77] propose heuristic methods to detect credible events in the context of tweets. They perform 'PageRank-like' iterations for authority propagation on a multi-typed network of events, tweets, and users, followed by an event-graph analysis using event similarities. Xu et al. [78] perform urban event analysis using search engine results. Ye et al. [79] propose truth discovery in the context of crowdsourcing. Meng et al. [80] propose optimization framework to find true values in the context of crowd-sensing tasks, such as air quality sensing, gas price, or weather condition estimation. [81] addresses crowdsensing in disaster response. Ipeirotis et al. [82] utilize a classical EM formulation to estimate worker quality in Amazon Mechanical Turk in the presence of worker bias. Participant recruitment is addressed in [83]. Aydin et al [84] address multiple-choice question answering via crowdsourcing. Su et al. [85] propose generalized decision aggregation in distributed sensing systems. Scalability was addressed in [86], and approximate truth discovery mechanisms by scale reduction was proposed. FaitCrowd [87] proposes Bayesian algorithm for truth discovery in crowdsourding. Miao et al. [88] propose privacy-preserving truth discovery in the context of crowd-sensing. [89] proposes additive and multiplicative models for crowdsourced claim aggregation. [90] detects spatial events via truth discovery in crowdsourcing. [91] exploits implication relations in the data for truth discovery. Li et al. [92] propose truth discovery for reliable medical diagnosis using crowdsourcing. Feeds from Twitter have been used for event detection [93, 94], explaining anomaly in traffic events [95,96], or exploring sports events [97]. Mukherjee et al. [98] combine some linguistic features with contextual information to estimate user credibility in the context of medical drugs using probabilistic inference. Sakaki et al. modeled Twitter users as social sensors to report earthquake in Japan [99].

Most of the related works discussed in this section utilize a network-based approach to detect facts or key events. A different school of thought uses semantic analysis, sentiment analysis, or natural language processing techniques to determine whether a tweet is likely fact [100–106]. In general, these solutions require a set of ground truth annotations to train a model specific to a language, situation, or context. Twitter is a global and multilingual media. Events transpiring in the physical world are dynamic in nature, and the signature to detect key facts may require contextual knowledge [101,104,105]. While our solutions can benefit from the presence of situation specific models, we did not want to depend on those. Therefore, in this dissertation, we primarily restrict ourselves to statistical techniques like EM or matrix factorization. Our algorithms look at source-claim structures, or information propagation patterns that can reliably find key facts even in the absence of (deep) content analysis.

EM-based fact-finders in the context of social sensing have been further extended. Wang et al. [107], Yao et al. [44] propose recursive fact-finders in the context of Twitter. The problem when the social media participants can post conflicting claims has been addressed in [108]. Later in this dissertation, we extend the EM-Social [3] formulation to polarized situation in social media [4]. Huang and Wang [109] propose confidence-aware and link-weight based [110] EM formulation for fact-finding in the context of social sensing. Hierarchy of the claims has been exploited in [43]. Ouyang et al. [111] propose EM algorithms to eliminate bias from crowdsensed local business and services discovery. Recent literature also extend the fact-finder formulation in specialized situations like place discovery [112], spatial-temporal-social constrained [113], theme-relevant [114], topic-aware [115], etc.

The social network of trust, influence, or dependency among the users have been explored in different application contexts. Wang et al. [116] propose iterative reinforcement mechanisms to identify fake reviews or review spammers using a social review graph. Agarwal et al. [117] propose mechanisms to estimate influence of the sources to the community in a blogging environment. TIDY [118] proposes a trust-based approach to information fusion by diversifying the set of sources.

The social dependency network can be used to detect rumors. Nel et al. [119] propose a method using the information publishing behavior of the sources and clustering sources with similar behavior. Shah and Zaman [120] propose "rumor centrality" as a maximum likelihood estimator to detect the source of rumors. Jin et al. [121] applied epidemiological models to study information cascades in Twitter resulting from both news and rumors. Castillo et al. [102] develop a method that uses source, content, and propagation patterns to classify rumors from non-rumors. The work on rumor-detection is largely complementary to ours. We do not explicitly detect rumor source in the dependency network. However, the mechanism to corroborate claims by multiple independent sources tend to suppress rumors.

While estimating factual information from the social media posts, we do not consider malicous sources. The error models considered in this dissertation consists of error caused by source dependency, polarization, or bias. It is also possible for certain sources to collude, act malicious and deliberately flood the network with false information. Algorithms for detecting sybil nodes [122, 123] might be used in such cases to drop the bad sources from consideration, before applying fact-finder techniques.

In this chapter, we use Apollo [9], a generic fact-finding framework that can incorporate different suitable fact-finding algorithms as plug-ins for a versatile set of applications. We use the aforementioned maximum-likelihood estimator [15, 124] as the fact-finding algorithm in Apollo. We demonstrate that the performance of fact-finding can be significantly improved by using simple heuristics for diversifying sources so that it uses sources that are less dependent on one another. Further improvements resulted in algorithms described in [3,43,44].

While our work on diversifying sources would not be needed if one could accurately account for dependence between them in data credibility assessment, we argue that, in general, estimating the degree of dependence between sources is very hard. For example, if one source follows another on Twitter and both report the same observation, it is hard to tell whether the second report is simply a relay of the first, or is an independent measurement. Given the ambiguity regarding the originality (versus dependence) of observations, we suggest that diversifying the sources is a useful technique whether or not credibility assessment can take dependence into account.

We implemented our source selection scheme as an online admission controller that is included as an upfront plug-in to the Apollo execution pipeline. Results show that our admission control can both speed up data processing (by reducing the amount of data to be processed) and improve credibility estimates (by removing dependent and correlated sources).

2.6 Summary

In this chapter, we considered a fact extraction problem from a large collection of user-generated tweets during two recent events, namely the Egypt unrest and hurricane Irene. We demonstrated that diversifying the sources can improve the results of extracting high quality information (i.e., facts or credible claims) from human-generated content. Human sources on social networks may describe events that do not constitute their own independent observations. This lack of independent corroboration may affect the accuracy of extracting information. We built different online admission controllers that filter tweets based on their sources and feed them into the fact-finding engine, Apollo. We observed that those admission controllers that used local social graph features such as the direct neighborhood of the source in question had inconsistent performance, whereas admission controllers that used more global features tended to perform better. In the current implementation, as a proof-of-concept, we leveraged the "follow" relationship between online users in Twitter as an indication of dependence between them. Other attributes that might potentially make sources dependent, such as geographic locations or communities to which users belong, will be investigated in the future.

The admission control mechanism improves the accuracy of distilled information. However, because we are dropping sources, we might be losing important and significant information. Therefore, the correlated error model caused by the dependent sources as described in this chapter has been further improved in [3]. The algorithm is known as `EM-Social`. EM-Social considers the dependency between the sources in a maximum-likelihood formulation. The retween pattern of the sources is used to estimate a social network of dependencies. This network provides information about parent-child relationship between the sources. When a parent claims a particular assertion, and a child also claims it, the claim from the child is given less weight depending on its repeat ratio. On the other hand, we consider those cases as independent where the child claims an assertion, while the parent didn't claim it. Experiment results showed that such mechanisms that integrated the source dependency inside the maximum-likelihood formulation improved the accuracy of the distilled facts by 10% to 20% for different scenarios.

CHAPTER 3

SOCIAL SENSING WITH POLARIZED SOURCES

This chapter addresses correlated errors in social sensing when the sources can be polarized. Such might be the case, for example, in political disputes and in situations involving different communities with largely dissimilar beliefs that color their interpretation and reporting of physical world events. Reconstructing accurate ground truth is more complicated when sources are polarized. The chapter describes an algorithm that significantly improves the quality of reconstruction results in the presence of polarized sources. For evaluation, we recorded human observations from Twitter for four months during a recent Egyptian uprising against the former president. We then used our algorithm to reconstruct a version of events and compared it to other versions produced by state of the art algorithms. Our analysis of the data set shows the presence of two clearly defined camps in the social network that tend of propagate largely disjoint sets of claims (which is indicative of polarization), as well as third population whose claims overlap subsets of the former two. Experiments show that, in the presence of polarization, our reconstruction tends to align more closely with ground truth in the physical world than the existing algorithms.

3.1 Overview

This chapter addresses the problem of reconstructing accurate ground truth from unreliable human observations. It extends recent crowd-sensing literature [125] by investigating reliable information collection from *polarized sources*. By polarization, we refer to a situation where different groups of sources hold largely different beliefs that color their interpretation, and hence representation, of events they observe. Hence, multiple competing versions of such events are reported. The goal of our algorithm is to identify versions that are more likely to be consistent with ground truth.

We apply our solution to extracting information from Twitter. We view Twitter as a participatory sensing system, where participants voluntarily report events they observe. The view of social networks acting as sensor networks was proposed in a recent survey on human-centric sensing [126]. We do not perform natural language processing on tweets (such a contribution would fall into another venue). Rather, in this chapter, we explore the merits of making *statistical credibility* assessments solely based on propagation patterns of different observations, as well as their degree of corroboration, *regardless* of their semantics.

There are two different schools of thought in information credibility assessment on Twitter. The first uses a machine learning approach that attempts to model human judgement of credibility. In this approach, classifiers are trained to recognize credible tweets as would be judged by a person (e.g., by a mechanical turk worker). Several recent papers proposed classification features of increasing degrees of sophisticatation that lead to increasingly good matches between human and machine credibility annotations [101, 102, 127].

The second school of thought comes from sensing literature and adopts an estimation-theoretic perspective. It assumes a unique ground truth that is realized in the physical world, and views humans as unreliable sensors who report such ground truth with possible errors and omissions. Statistical (estimation-theoretic) techniques are then used to determine the likelihood that these sensors are correct, given the correlations between them (e.g., that arise from social ties and retweets). An example of this approach in a recent expectation maximization algorithm that jointly estimates the unknown source reliability as well as the statistical tweet credibility [125]. The work was extended to account for non-independent sources [3] and non-independent claims [128].

We adopt the latter school of thought. In this work, we are more interested in understanding the *physical world* (i.e., in sensing) as opposed to understanding what humans perceive as credible. Following this model, we abstract human observers as binary sensors [3] in that each reported observation is either true or false. The novelty of this contribution lies in considering sources that are polarized. Intuitively, polarization affects our model of *correlations* in (human) sensor outputs: when sources (viewed as unreliable binary sensors) share a more significant bias towards a topic, their observation (bit) errors on that topic are more correlated. On the other hand, when they do not share a bias, their errors are independent. Note that, when sources are correlated, corroboration among them carries less statistical weight than when they are independent. Hence, when statisically assessing the likelihood of error in an observation reported by multiple sources, it is important to know whether the topic of that observation matches the bias of the sources or not. The answer determines whether such sources should be regarded as correlated or not, leading to a *topic-dependent* source correlation model. Later in the chapter, we explore the above intuition more formally to arrive at a polarity-informed maximum-likelihood estimate of statistical credibility for each reported observation.

Another advantage of the estimation-theoretic approach adopted for credibility assessment in this chapter is that the resulting estimator has a known error bound. This bound was computed in prior work [129], and remains applicable to ours. Hence, not only do we compute truth estimates but also arrive at confidence intervals in source reliability.

We evaluate our solutions using real-world traces collected from Twitter. We recorded observations from Twitter for four months during a recent uprising against the former Egyptian president. We manually annotated a fraction of tweets depending on their degree of support to the deposed president as *pro*, *anti*, or *neutral*. We henceforth call these tweets *claims*, with no implication as to their degree of credibility. We then studied the propagation patterns of these different groups of claims and adapted our previous fact-finder to recognize polarization. The fact that different topics propagate on different dissemination trees is intuitive and has already been pointed out in prior literature [100]. The contribution is novel in its investigation of the specific case of polarized sources and in accounting for polarization in maximum-likelihood credibility assessment.

The investigation of our particular data set revealed the presence of two clearly defined camps in the social network that tend to propagate only one group of claims, as well as a population that tends to propagate selected claims with less correlation with their polarity. We estimated their respective polarity-dependent propagation networks. Each network was then used to compute correlations among sources for the purposes of computing their error-independence properties. For comparison, we also estimated the propagation network constructed when content polarity is not taken into account, as done in previous estimation-theoretic work on truth estimation [3]. We observed that the latter network matches the respective polarity-dependent propagation networks when describing the graph neighborhood of strongly polarized sources, but diverges when describing the neighborhoods of sources that are more neutral. This causes the previous approach to infer incorrect correlations for neutral sources. We show that these false correlations lead to degradation in truth estimation in favor of polarized information. Our new approach avoids this pitfall.

The rest of this chapter is organized as follows. In Section 3.2, we present a case study for this work that shows how polarized certain situations can be. In Section 3.3, we propose a model for polarized sources, claims, and bias-aware social networks. In Section 3.4, we present a formulation of the problem and derive algorithms to solve it. Experimental evaluation is presented in Section 3.5. Related work is reviewd in Section 6.6. Finally, we present conclusions and future work in Section 3.7.

3.2 The Case of a Polarized Network

We analyzed traces obtained from Twitter during a recent uprising in Egypt that resulted in deposing the president. The collected tweets expressed either a positive or negative sentiment towards the deposed president. These tweets were first clustered such that tweets making the same observation (typically the same sentence or very similar sentences) were put in the same cluster. Each such cluster was viewed as a *single claim*. By observing the time at which different sources contributed their tweet to a given cluster, it was possible to identify a propagation cascade of the corresponding claim through the social network. Table 3.1 presents statistics of the tweets collected.

To asses polarization, it is required to classify the claims into *pro*, *anti*, and *neutral* classes. We chose to manually annotate the largest cascades as those represent the claims that have been observed or propagated the most; therefore more likely to cover important or popular events. Figure 3.1 shows that the distribution of cascade sizes is approximately heavy tailed. This observation suggests that considering a small number of top claims is sufficient to represent a large number of tweets. We experimented by manually annotating 400 and 1000 largest cascades, and the results were

Table 3.1: Summary of the tweets collected

Query	`Egypt` $\vee$ `Morsi` $\vee$ `Cairo` $\vee$ Location: 100 miles around Cairo
Number of tweets	4.3M
Total size	17 GB
Tweets containing `Morsi`	900K
English tweets containing `Morsi`	600K
Number of cascades	193K

similar. We describe the case with 1000 largest cascades. Collectively, these cascades accounted for roughly 44K sources and 95K tweets.

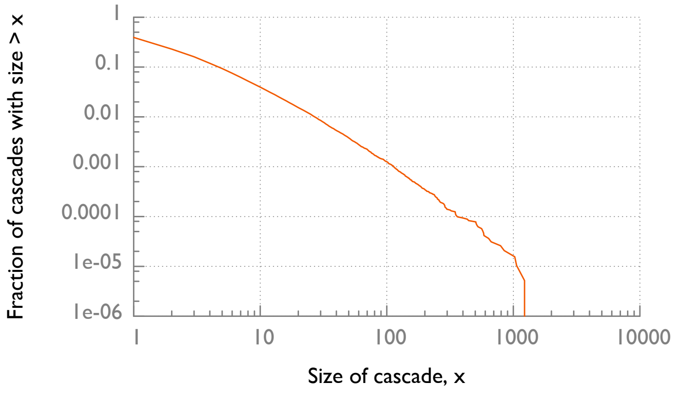


Figure 3.1: Complementary cumulative distribution of cascade sizes

Figure 3.2 plots the distribution of the probability of a source to tweet *pro* in the top 1000 cascades. The figure illustrates a very interesting property of these cascades. Namely, it is evident that there are three visibly different types of sources. The first type, accounting for 50% of the sources, has a near 1 probability of tweeting *pro*. The second type, accounting for more than 20% has a near zero probability of tweeting *pro* (i.e., mostly tweets *anti*). The rest of the sources tweet both polarities. They are located in the middle of the plot. We call them “neutral” sources. The figure suggests that the community is clearly polarized. This observation motivates us to ask the questions: Does this polarization affect the accuracy of reconstruction of physical world events via social sensing? How reliable are previous data cleaning approaches in the presence of polarized sources? How to circumvent their shortcomings?

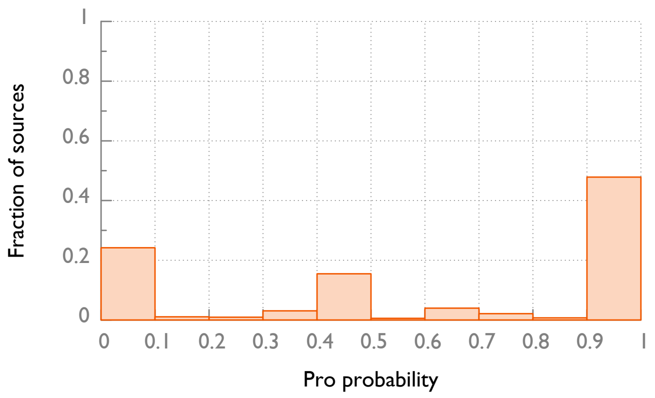


Figure 3.2: Distribution of pro tendency of sources

We show in our evaluation that, in general, community polarization is strong enough to confuse previous algorithms, and therefore polarity-aware credibility analysis algorithms are necessary.

3.3 A Model for Polarized Social Networks

This section presents a model of *polarized social networks* acting as sensor networks. In the following subsections, the models for claims, (polarized) sources, and their dependencies are described.

3.3.1 Modeling Polarized Claims and Sources

Consider m sources who collectively make n claims (i.e., generate n cascades). The relation between the claims and their sources can be represented by a source-claim network, SC, which is a bipartite graph. We conveniently represent it using a $m \times n$ matrix, such that $SC_{i,j} = 1$ if source S_i makes claim C_j (i.e., contributes to j^{th} cascade), and 0 otherwise.

We consider a binary model, where each claim can be *True* or *False*. This categorization is orthogonal to polarity. To model polarized claims, we introduce a topic indicator, y_j, for each claim C_j, that takes one of the values from topic set $T = \{pro, anti, neutral\}$. This topic represents the polarity of claim C_j. A vector y is defined as the polarity vector for all claims.

In general, a source may make claims of different polarity. We define the reliability of a source as the probability of making correct claims. Note, however, that when making claims that agree with the source's own bias, the source might become less selective and have a higher probability of making false claims. In contrast, when making claims that are orthogonal to the source's bias, the source might get more conservative and the probability of correctness increases. This suggests that source reliability is a vector, with one entry per topic. Hence, we model a source S_i by a vector r_i of dimension $|T|$, where $r_{i,t}$ denotes the reliability of the source when making claims of polarity T_t.

3.3.2 Modeling Polarity-aware Source Dependencies

Prior work on credibility assessment in social sensing [125] developed an algorithm that takes a source-claim network, SC, as input, and jointly estimates both reliability of sources and statistical credibility of claims. The algorithm was then adapted to take into account dependencies between sources [3]. As mentioned earlier, such dependencies imply correlated errors that need to be accounted for in statistical analysis.

A dependency between two sources is a directional quantity. It is estimated by observing the probability that one source propagates information obtained from the other (i.e., joins a cascade given that the other source joined it earlier). Representing such correlations by directional links between the respective source nodes, a propagation graph is constructed that constitutes the inherent social (influence) network. Netrapalli and Sanghavi [130] formulate the problem uncovering the latent influence network (or information propagation graph), given a sufficient number of cascades. We use their algorithm to generate social networks given the set of sources, tweets, and their timestamps.

An alternative method of finding the latent network is to take the Twitter-provided follower-followee graph. However, the follower-followee graph is not always a good representation of actual information propagation paths exercised by users. For example, as most of the tweets are public, when an event of significance transpires in the physical world, interested individuals may search for top tweets and act on those. This method does not require following any particular person and therefore the follower-followee relationship is an incomplete proxy for the underlying information propagation network.

Another possibility is to construct the propagation graph directly from retweets. For example, if source A retweets source B, k times, insert a weighted directed link (A, B, k) in the network. The problem with this approach is that in large cascades it is not clear who exactly (of those who tweeted the same claim earlier) a source was influenced by. Hence, the retweet relation does not necessarily reflect the correct influence topology. The influence network estimation approach proposed by Netrapalli and Sanghavi [130] avoids this problem, which is why we adopt it in this work.

A further advantage of using the approach of Netrapalli and Sanghavi [130] for estimating the influence propagation network is that we no longer care whether something is a retweet, or a separately authored tweet of similar content. All that matters for this algorithm are the clusters of tweets (of similar content), each forming a cascade, and the timestamp of each tweet in each cascade. Hence, the approach is not restricted to uncovering influence propagation via the Twitter medium itself. A source may influence another externally (e.g., via a different communication medium). The external link can still be uncovered as long as both sources make tweets of similar content.

To model polarity-aware source dependencies, we generate $|T|$ different influence propagation networks, using the aforementioned algorithm [130], by observing claims of a single polarity at a time to infer a single network. The set of these networks is collectively referred to as SD^B, where element SD^B_t is the network generated by considering only the claims of polarity T_t. We call the corresponding networks *pro*, *anti*, and *neutral* networks. For comparison, we also construct a generic network, SD, by considering all claims regardless of their polarity. In Section 3.5, we empircally evaluate the quantitative differences between SD^B and SD.

Please note that the pro (anti, neutral) network is *not* a network of only the pro (anti, neutral) sources, rather it is a network created using only the pro (anti, neutral) claims. As a result, these networks may contain overlapping sources if such sources make claims of different polarities. The terms *pro source*, *anti source*, and *neutral source*, when used, therefore refer to the predominant disposition of a source as opposed to exclusive membership of one of the networks.

3.4 Ground-truth Estimation in Polarized Networks

This section formulates the problem of ground truth estimation in polarized networks and describes the algorithm we use to solve it.

3.4.1 Problem Formulation

Based on the model described in section 3.3, the problem is to estimate the statistical credibility of each claim given the source claim network, SC, the polarity of each claim, specified in the vector, y (where y_j is the polarity of claim C_j), and the inferred set of influence propagation networks, SD^B, one per polarity. Let z_j be the unknown ground truth value of claim C_j (stating whether it is true or false). Formally, we want to compute:

$$\forall j, 1 \leq j \leq n : \Pr(z_j = True | SC, y, SD^B) \tag{3.1}$$

3.4.2 Solution

As discussed earlier, the bias of a source may cause it to be less selective in making claims of one polarity compared to another. For example, the source might indiscriminately propagate claims that agree with its bias, while being selective in making other claims. Hence, source reliability (probability of making claims that are true) may depend on claim polarity. Let the reliability of source, S_i, when making claims of polarity, T_t, be denoted r_{i,T_t}. For simplicity, in this chapter, we assume that the source reliability values for different polarities are independent. The polarities of interest are $T = \{pro, anti, neutral\}$. Hence, we can break down Expression (3.1) into three independent subproblems; namely, computing the credibility of *pro*, *anti*, and *neutral* claims, respectively. This is formally expressed as finding the probabilities below:

$$\forall j, y_j = pro : \Pr(z_j = True | SC_{y_j=pro}, SD^B_{pro}) \tag{3.2}$$

$$\forall j, y_j = anti : \Pr(z_j = True | SC_{y_j=anti}, SD^B_{anti}) \tag{3.3}$$

$$\forall j, y_j = neu. : \Pr(z_j = True | SC_{y_j=neu.}, SD^B_{neu.}) \tag{3.4}$$

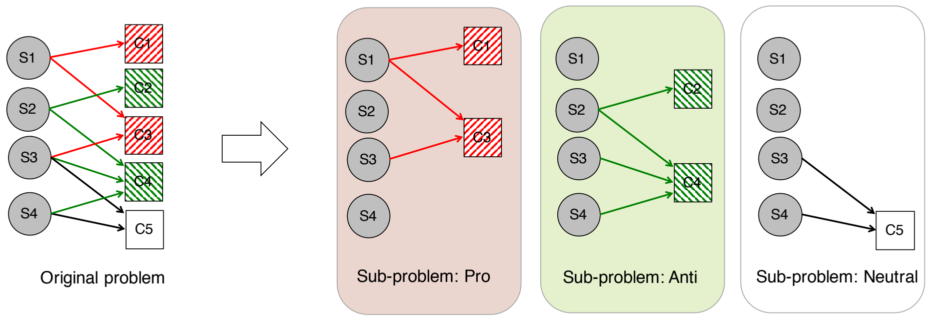


Figure 3.3: Executing polarity aware fact-finder

where $SC_{y_j=pro}$, $SC_{y_j=anti}$, and $SC_{y_j=neu.}$ are the subgraphs of the source claim network, SC, with claims of only the specified polarity present (or equivalently, the array SC with claim columns of other polarities removed). The independence assumption between source reliability parameters $r_{i,pro}$, $r_{i,anti}$, and $r_{i,neutral}$ makes it possible to solve for variables (3.2), (3.3), and (3.4) separately, essentially breaking the original problem into three independent subproblems, one for each polarity. In the subproblem corresponding to polarity, T_t, we consider the source claim subnetwork $SC_{y_j=T_t}$ and the inferred influence propagation network $SD^B_{T_t}$, then solve jointly for source reliability r_{i,T_t} and statistical claim credibility, z_j, where $y_j = T_t$.

Figure 3.3 illustrates the formation of the subproblems. Here S_1 to S_4 are the sources, and C_1 to C_5 are the claims. There is an edge in (S_i, C_j) in the bipartite network if source S_i authored claim C_j. The *pro* claims are shown in red, the *anti* claims are shown in green, and the *neutral* claims are shown in white. The proposed polarity-aware algorithm identifies each 'class' of claims, and considers the independent subproblems that contain all the claims of that particular class and the sources that make them. The solution to each subproblem results in credibility scores for the claims in that particular class, as well as one element of the polarity-aware reliability vector of the sources.

More specifically, each subproblem is solved using the expectation maximization algorithm presented in [3]. Starting with an initial guess of source reliability parameters, expressed as the vector θ_0, the algorithm performs the iterations:

$$\theta_{n+1} = \arg\max_{\theta}\{E_{z|SC_{y_j=T_t},\theta_n}\{\ln \Pr(SC_{y_j=T_t}, z^t|SD^B_{T_t}, \theta)\}\} \tag{3.5}$$

where z^t is the vector of latent variables z_j (claim credibility), for all claims, where $y_j = T_t$. The above breaks down into three steps:

- Compute the log likelihood function
  $\ln \Pr(SC_{y_j=T_t}, z^t | SD^B_{T_t}, \theta)$
- The expectation step
  $Q_\theta = E_{z^t|SC_{y_j=T_t},\theta_n}\{\ln \Pr(SC_{y_j=T_t}, z^t|SD^B_{T_t}, \theta)\}$
- The maximization step
  $\theta_{n+1} = \arg\max_\theta\{Q_\theta\}$

where the last two steps are solved iteratively until they converge, yielding updated source reliability estimates and claim credibility, z_t (for claims of polarity T_t).

3.4.3 Polarity Classification of Claims

Our polarity aware model assumes that there exists a mapping y from claims to polarities. This mapping is required to divide the set of tweets into $|T|$ parts. We manually annotated the top 1000 largest cascades (most propagated claims). However, to use our polarity aware credibility estimation algorithm as a crowd-sensing tool, it is important to include all the claims in the analysis. Therefore, an algorithm to classify each incoming tweet into a particular polarity is required.

We attempted to use readily available learning-based sentiment analysis tools for this purpose that look at the content of the tweets and classify them into positive and negative sentiments. It was not sufficient because the polarity of a tweet is not necessarily correlated with its sense or sentiment being positive or negative. For example, "The government is working for the people", and "The opposition is working against the people" have positive and negative sentiments respectively; but polarity of both of these claims are likely to be pro-government.

It is possible to design an advanced classifier for this purpose that uses learning techniques or natural lanuage processing methods to classify the tweets into *pro*, *anti*, and *neutral* classes. However, such a classifier requires extensive domain-specific knowledge and its design depends on the choice of polarity classes and their context. Moreover, simple learning-based tools often suffer from low quality and require extensive training. A domain-specific classifier that looks at the content and determines the polarity is therefore hard to generalize.

Instead, given our seed of manual annotations, we used an iterative algorithm that propagates tweet annotations to source annotations, and then from source annotations back to tweet annotations, repeatedly. Sources that predominantly make tweets of a given polarity are identified from the manually annotated tweets and other tweets of the same sources are given the same polarity. This algorithm is clearly an approximation. Nevertheless, even this approximate polarity annotation can lead to an improvement in fact-finding, compared to polarity-unaware analysis. Later in this dissertation, we automate the polarity detection mechanism [7], in chapter 4 and chapter 5.

3.5 Experiments

In this section, we describe the experiments performed to determine how community polarization affects statistical credibility estimation in social sensing. Our experiments use the traces obtained from Twitter during the recent uprising in Egypt resulting in deposing the president (summarized in Table 3.1). The crawling started in July, 2013 and continued for four months.

3.5.1 Polarization Analysis

A key hypothesis of our work is that a better solution to the credibility estimation problem is obtained by breaking all tweets by polarity and solving independently for credibility of tweets in each polarity class, T_t, given the polarity-specific source-claim matrix, $SC_{y_j=T_t}$, and the polarity-specific influence propagation network, $SD^B_{T_t}$. This is as opposed to amalgamating all tweets regardless of polarity into one source claim matrix, SC, and using a single influence propagation network, SD, as inputs to the credibility estimation.

To appreciate the difference between the two solutions, some analysis of the resulting networks is needed. For this analysis, we read the text of the largest

1000 claims and manually annotated them as *pro*, *anti*, or *neutral*. The annotation revealed that there are 199 *pro* cascades and 109 *anti* cascades in the top 1000 largest cascades. By utilizing the timestamps of when each source forwarded a given claim, we estimated the inherent social propagation network for each type of claims using the algorithm proposed by Netrapalli and Sanghavi [130].

This resulted in 15,714 edges in the *pro* network SD^B_{pro}, 8,460 edges in the *anti* network SD^B_{anti}, and 33,946 edges in the *neutral* network $SD^B_{neutral}$. We also estimated the generic network SD using all 1000 cascades together. There are 55,329 edges in that network.

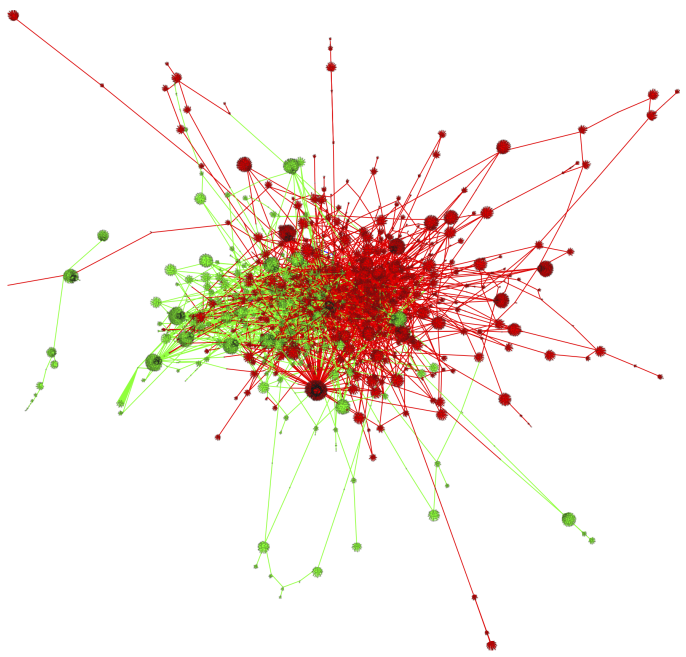
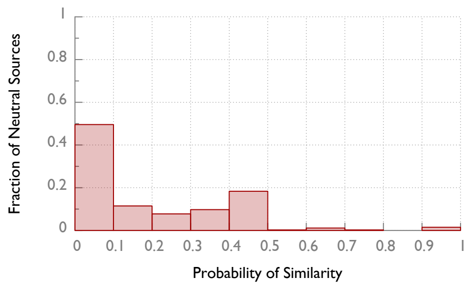


Figure 3.4: An overlay of two polarized social networks. Pro shown in red and anti shown in green

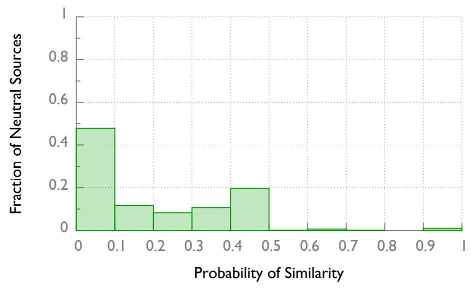
Figure 3.4 shows the *pro* network, SD^B_{pro}, in red, and the *anti* network, SD^B_{anti}, in green, overlayed together. The neutral network is not shown to maintain visual clarity.[1] This plot suggests that two polarized groups exist with their own different propagation links.

With that preparation, we are ready to answer the question: is considering one amalgamated influence propagation network the same as considering a polarity-specific network, when estimating the credibility of tweets?

[1]Source are further restricted to only the top 400 cascades for clarity.



(a) Neutral sources: pro network vs. generic network

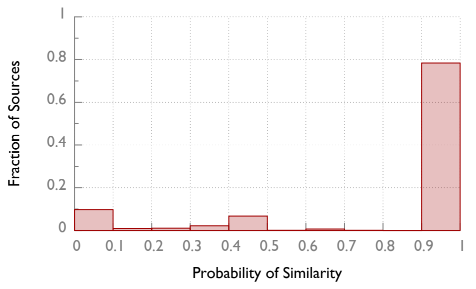


(b) Neutral sources: anti network vs. generic network

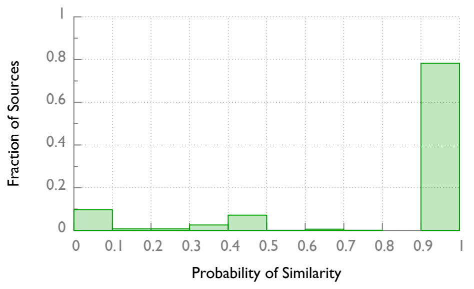
Figure 3.5: Distribution of neighborhood similarity of neutral sources between polarized and generic network

The answer is no. It turns out that the neighborhood of neutral sources is not correctly represented in the amalgamated network. This results in improper modeling of source dependencies, which affects credibility estimation when such sources propagate *pro* or *anti* tweets. To see the difference in source dependency estimation when neutral sources propagate *pro* or *anti* tweets, consider Figure 3.5, which compares the neighborhood of neutral nodes in the amalgamated influence propagation network, SD, versus that in the *pro* or *anti* network (SD^B_{pro} or SD^B_{anti}). The degree of similarity is measured by the jaccard similarity coefficient between the two sets of neighborhoods. The similarity distribution between SD^B_{pro} and SD is shown in Figure 3.5a. The similarity distribution between SD^B_{anti} and SD is shown in Figure 3.5b. It is seen that more than 98% of the sources have different neighborhoods in the amalgamated SD network compared to the SD^B_{pro} and SD^B_{anti} networks. This means that the amalgamated network does not properly capture their dependencies. Further inspection suggests that it exaggerates them, leading the statistical estimation algorithm to rely less on such sources (to avoid correlated errors).

The same cannot be said of polarized sources. Figure 3.6 shows that the generic network SD does not confuse the neighborhood of the strongly polarized sources. Figure 3.6a shows the distribution of neighborhood similarity between SD^B_{pro} and SD, and Figure 3.6b shows the distribution of neighborhood similarity between SD^B_{anti} and SD. The generic network SD correctly determines the neighborhood for around 80% of the polarized sources. This is expected. Those sources forward mostly one polarity of claims. Hence, the



(a) Pro sources: pro network vs. generic network



(b) Anti sources: anti network vs. generic network

Figure 3.6: Distribution of neighborhood similarity between polarized and generic networks

estimation of influence propagation returns the same results whether all or only those claims are considered.

The above figures offer an intuition into the shortcomings of the amalgamated approach from the perspective of credibility estimation: the approach tends to "disenfranchise" neutral sources.

3.5.2 Fact Finding

We compare the accuracy of our polarity-aware credibility estimation algorithm to its predecessor [3] that does not consider the polarity of tweets. We identify our algorithm by the word 'Polarized' and the other algorithm by the word 'Combined'.

To evaluate the fact-finding performance, we executed three experiments by selecting the largest n cascades, for $n \in \{400, 1000, 5000\}$. Summaries of the datasets used in each experiment are presented in Table 3.2. In each experiment, we classified the claims into the three polarity classes and ran polarity-aware and polarity-unaware estimators. In each case, the fact-finder computed the credibility of input claims $\in [0, 1]$ and the reliability of their sources $\in [0, 1]$.

Figure 3.7 shows the relation between the output of different algorithms in different experiments. The circle and triangle pointed curves show the fraction of claims that are believed as facts by the combined and the polarized algorithm, respectively. We find that the combined algorithm is less judgmental and believes more claims to be true. The square pointed curve

Table 3.2: Summary of the dataset of the experiments

	Experiment 1	Experiment 2	Experiment 3
Cascades	400	1000	5000
Pro claims	105	199	379
Anti claims	50	109	371
Neutral claims	245	692	4,250
Number of sources	31,480	43,605	68,206
Number of tweets	62,864	94,871	184,452
Pro tweets	17,603	22,750	27,114
Anti tweets	8,509	11,691	19,411
Neutral tweets	36,752	60,430	137,927
Source-claim edges (total)	43,024	68,092	140,170
Source-claim edges (pro)	13,057	17,152	22,773
Source-claim edges (anti)	6,770	9,302	16,380
Source-claim edges (neutral)	24,197	41,638	101,017
Pro network edges	12,160	15,714	23,942
Anti network edges	6,292	8,460	19,037
Neutral network edges	19,735	33,946	92,683
Combined network edges	36,472	55,329	130,092

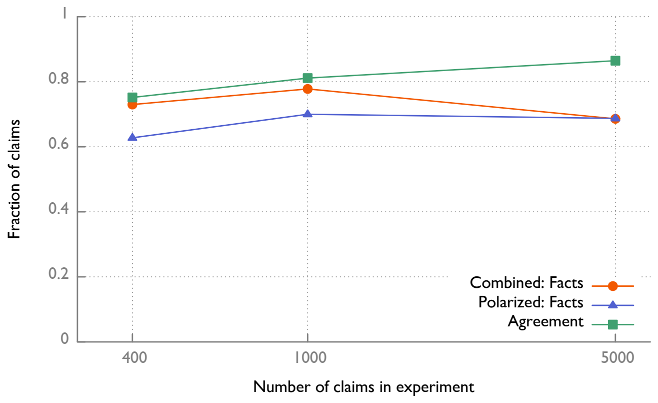


Figure 3.7: Number of claims believed as facts by different algorithms

shows the agreement between two schemes. The agreement is computed as the jaccard similarity between the two sets of claims believed as facts by the two algorithms. It is evident that the two algorithms converge more as the number of claims increase. We conjecture that this is because polarized claims were retweeted more and had larger cascade sizes. Hence, the smaller experiments had more polarized claims, offering a larger difference in results between the two approaches.

From Table 3.2, the probability of an arbitrary claim to be polarized is nearly 39% in the 400 claims experiment, while its nearly 31% in the 1000 claims experiment, and only 15% in the 5000 claims experiment. We also classified the tweets for a 10,000 claims and a 25,000 claims experiment, where the probability of a claim to be polarized went further down to 11% and 7%, respectively.

Finally, we evaluated the quality of information obtained by the polarized algorithm and the combined algorithm. Here, we present the comparison for the 1000 claims experiment. In this experiment, the polarized algorithm selected 128 pro, 76 anti, and 498 neutral claims as true (a total of 700 claims). The combined algorithm selected 147 pro, 88 anti, and 543 neutral claims (a total of 778 claims). Of the two sets, 662 claims were common in both cases, resulting in a degree of agreement of 81.13%.

The interesting cases, however, are those where the algorithms disagreed. We considered two sets of claims on which there was disagreement. Set A contained the claims that the polarized algorithm believed to be true with a probability 1, but the combined algorithm did not. There were 38 such claims. Conversely, set B contained the claims that the combined algorithm believed to be true with probability 1, but the polarized algorithm does not. There were 116 such claims.

The two sets were merged and presented to a human grader without the information on which claim came from which set. The grader was instructed to carefully research and verify each claim using historic data. Verified facts received a score of 1. Fabricated claims and lies received score of -1. Non-factual claims such as expressions of emotion, slogans, and sentiments were discarded (received a score of 0). After grading was done, we separated the sets again and calculated the scores for each algorithm. The results are presented in Table 3.3.

If we count non-factual claims (i.e., expressions of emotion, etc) then, when the algorithms disagree, 66% of the claims believed by the polarized algortihm are true, compared to 62% for the combined algorithm. More interestingly, the polarized algorithm believes only 2.6% false claims (that received a -1 score), while the combined algorithm believes 8.6% false claims. If we discard non-factual claims from the total (after all, they do not refer to binary facts), then when the algorithms disagree, 96% of the claims believed

Table 3.3: Quality of exclusive information

Definition	Set A Claims exclusive to Polarized	Set B Claims exclusive to Combined
Total	38	116
Factual	26	82
Non-factual (0)	12	34
True (1)	25	72
False (-1)	1	10
Factual true	96%	88%
Sum of scores	$25 - 1 = 24$ out of 38	$72 - 10 = 62$ out of 116

by the polarized algortihm are true, compared to only 88% for the combined algorithm. Equivalently, the probability of error is reduced (in our case) from 12% to 4%, or by a factor of three!

Finally, combining all scores to get a single overall quality indicator, our bias-aware crowd-sensing algorithm improves the quality by more than 18%.

The results shown above are a step forward. They demonstrate that when sources are polarized, we should consider separately the *pro*, *anti*, and *neutral* claims in performing credibility analysis. Such separation prevents estimation of false dependencies between neutral sources, based on amalgamated retweet patterns. By separating the content and considering only polarity-specific dependencies, errors are reduced.

3.6 Related Work

Crowd-sensing is an increasingly popular area of research, where humans are acting as the sensors generating observations. It extends more traditional participatory sensing models where humans carry the sensor devices that collect data. Human-generated observations have a different error model than traditional sensors, which introduces many interesting questions and challenges.

Wang *et al.* [125] addressed the question of reliable sensing in the context of human observers. He proposed a model where human observations are treated as binary claims that can be either true or false. The question of estimating credibility of a particular claim can be trivially addressed by voting (i.e., a claim with a larger propagation is deemed more credible). However, this simple approach is highly suboptimal when sources have different degrees of reliability. Wang's approach [125] jointly estimated source reliability and claim credibility for independent sources. When source are generally not independent, source diversification heuristics were studied that select tweets from only a subset of sources to maximize a measure of mutual independence [2]. A more principled solution that models source dependencies directly and accounts for them in the maximum likelihood framework was described in [3]. Our work builds on this foundation, while accounting for the polarized sources.

Information propagation through social or other complex networks has been studied extensively [131–134]. Netrapalli and Sanghavi [130], Myers and Leskovec [135], and Rodriguez *et al.* [136] model the propagation of information through social networks as epidemic cascades and use different ways to estimate the propagation graph from multiple cascades. This work nicely complements ours, since the latent influence propagation network is one of the inputs to our maximum likelihood (credibility) estimator. A related problem is community detection. Several efforts addressed the issue of detecting different communities in social networks [137, 138]. These methods can be used to confirm that influence cascades indeed propagate largely within corresponding community boundaries.

Topic-based models to infer user influence and information propagation have been studied in different contexts. Lin et al. [139] proposed a probabilistic model to infer the diffusion of topics through social networks. Pal and Counts [140], and Eytan *et al.* [141] propose methods to infer topic-based authorities and influential nodes in the context of online social platforms and microblogs. The concept of social media genotype to model and predict user activity in social media platforms was propsoed by Bogdanov *et al.* [100]. The genotype is a set of features that defines user behavior in a topic-specific fashion. Like us, they argue that a single static network is not a good indicator of user activity. Instead, they derive topic-aware influence backbones based on user genotypes, which we exploit in understanding how different polarities (topics) of information follow different paths in the social network. They focus on predicting user activity, while we are interested in improving the quality of fact-finding.

Finally, our work is related to the more general genre of crowd-sourcing; using the crowd to perform useful tasks [142, 143]. Unlike our work, where participants are unaware of their participation, this genre of research considers a more controlled and structured environment, where people are generally paid to participate in advertised tasks.

3.7 Summary

The chapter addressed truth recovery from tweets in the case of a polarized network. It was shown that polarization impairs credibility estimation. The problem was solved by developing a new polarity-aware estimation methodology that improves quality of results by 18%. Several extensions of the current framework are possible. For example, we assume that polarities are already known. Advanced classifiers that aggregate both content and provenance information may prove useful to reduce the need for manual polarity annotation. Although we adopt an estimation-theoretic perspective for credibility assessment, our algorithm can be easily extended to incorporate additional machine learning analysis or natural language processing on the text, which may improve the fact-finding performance. The idea of polarities can be extended to topics with arbitrary relations and overlap. Also, while this work considered sources that are polarized, it did not regard them malicious. An intent to decieve by an intelligent adversary presents a harder challenge. These extensions are delegated for future work.

CHAPTER 4

EVALUATING POLARIZATION MODELS IN SOCIAL NETWORKS

In this chapter, we develop and evaluate models of information propagation on social media in the presence of polarization, where opinions are divided on issues of contention into multiple, often conflicting, representations of the same events, each reported by one side of the conflict. Multiple models are compared that derive from the hypothesis that individuals propagate more readily information that confirms their beliefs. We use these models to solve the inverse problem; namely, given a set of posts in a conflict scenario, automate their separation into opinions that favor each side, as well as pieces that appear to me more neutral. Specifically, we develop new maximum-likelihood estimation algorithms for separation of polarized Twitter posts. We show that our solutions allow for such opinion separation to occur in an unsupervised fashion, and without machine interpretation of the disseminated content, while offering analytic means for quantifying accuracy of results. Our empricial evaluation, based on multiple Twitter data sets, confirms the accuracy of content separation, offering a new capability for viewing unbiased representations of events, or isolating the positions of different sides of a conflict.

4.1 Overview

This chapter studies models of polarization on broadcast-based public social media that support microblogging. For scenarions where polarization is present, we develop algorithms that separate media content into different positions, held by the respective sides of the conflict, as well as isolate what appears to be more neutral content. We show that such separation can occur in a manner that is both (i) language-agnostic (i.e., without machine interpretation of content), and (ii) fully unsupervised (e.g., without *a priori* knowledge of the individual sources and their beliefs, and without the use of labeled data or remote supervision techniques that train ahead of time based on existing text corpora).

The work is motivated by the rise of social media platforms, such as Twitter, that democtratize information broadcast, offering everyone not only the opportunity to share opinions at an unprecedented scale [144], but also to find, tune-in to, and copy opinions of like-minded individuals [145]. These affordances for information sharing, input selection, and downstream content propagation set the stage for the formation of online echo-chambers [146], where different groups of like-minded individuals propagate often-conflicting information that confirms their individual beliefs [4, 5, 147]. We call such a situation, *polarization*.

The work follows on recent interest in polarization in social media [4, 5, 7, 148–150], where the feasibility of unsupervised separation of different opinions was first established [7]. Their approach offered a heuristic based on matrix factorization. Other works offer methods to quantify polarization using distribution of opinions and graph partitioning algorithms [149, 150]. In contrast, we study generative models of information dissemination and their application in a maximum-likelihood framework to solve the opinion separation problem. Our models are based on one underlying hypothesis, confirmed in recent social media studies [148, 151–154], that individuals tend to propagate more readily information that confirms their beliefs. Hence, in aggregate, they form propagation topologies that offer different "impedance" to different types of content. By observing which content propagates more readily on which topologies, it is possible to jointly isolate the different sides of a conflict (nodes in the topology) together with their beliefs (posts propagating on the edges). Neutral content is not specific to a single side and hence propagates indiscreminately in both [148, 155]. These models of information propgation, in turn allow solving the inverse problem. Namely, they lead to maximum-likelihood estimation algorithms that, given a set of posts, automatically separate the opinions espoused by individual sides, as well as content that appears to be neutral. The capability for such separation can have many applications. For example, it could be used as an automated means for identifying less biased representations of events (by focusing on neutral content), or presenting the online positions of different sides.

The problem of modeling polarization, described above, is different from many other recent analysis challenges in social media. For example, it is different from *fact-finding*. Recent literature developed algorithms to reconstruct physical events from posted observations by separating credible and false observations [3]. Such algorithms have been shown to result in a biased reconstruction of events when incorrect posts have large support in the population [4], as might occur due to shared biases, which in essence creates correlated errors. Work, presented in this chapter, can mitigate such an effect by identifying potentially biased content.

The work also addresses a different concern from sentiment analysis [156, 157]. While current solutions to sentiment analysis aim to recognize the sentiment expressed in individual posts [158,159], we are more concerned with separating the positions of different sides with respect to issues of concern, regardless of their sentiment towards the individual issues.

It is also different in focus from community detection. Much of the community detection work rests on variations of the observation that individuals, by homophily, interact more within their community than across communities. While indeed the different sides of a conflict can be thought of as different communities based on their information propagation characteristics, the existence of neutral content complicates the separation process. Neutral content, by definition, propagates orthogonally to the conflict divide, thus leading to significant links across clusters [155]. Moreover, the boundaries of formed clusters shift depending on discussed content [148]. For example, we show that following a recent democratic election that resulted in significant polarization, the clusters blended together again, as the topic of discourse shifted. Our problem is thus more about separating potentially biased *content*, as opposed to binning sources.

Finally, the work does not require prior knowledge or supervision. Indeed, our results can be further enhanced by exploiting prior knowledge of individual sources or by using language and vocabulary models that help interpret the content. The advantage of not using any prior source-specific or language-specific information, however, is that the algorithmic techniques developed in this chapter become generally applicable to any population (we do not need to know the sources ahead of time) and any language (we do not need to know the language or dialect used in the discourse). As such, the cost and prerequisites of developing a working system are significantly reduced.

The rest of this chapter is organized as follows. Section 4.2 explains the polarization problem and provides the background. Section 4.3 formulates modeling polarization as a maximum likelihood estimation problem, and provides a candidate solution using EM mechanism. Section 4.4 and section 4.5 provides alternate formulations. Because it is not clear which solution is better, we perform a comparative study using controlled simulations in section 4.6. Section 4.7 implements our algorithm, evaluates with real-world datasets collected from Twitter, and compares with other candidate mechanisms. Section 4.8 reviews related literature, and finally section 4.9 discusses the findings and provides directions for future research.

4.2 The Polarization Problem

To illustrate polarization on social media, consider the following excerpts from posts on Twitter (i.e., tweets) describing a series of clashes between police and demonstrators in Bahrain, a Southwest-Asian state on the Arabian peninsula. Some of the excerpts are very anti-police: "#Bahrain This is How #Cops #Kill #People With #Automatic #Shotguns", "#BAHRAIN: This Is How Police Kill and Shoot the Citizens Demanding For FREEDOM", and "Resistance by revolutionary youth across #Bahrain last night after the police kill an 18 year old". Other posts are more emphathetic with the police: "Those Kids were instruments used by #Bahrain Protesters 2 Kill civilians & Police", "Bahrain shia protesters use molotov to kill police", and "Protesters in Bahrain attacking police and kill them". Finally, some have a neutral tone: "Violent confrontation between the revolutionary youth and the riot police troops in the streets of #Manama #Bahrain".

The question addressed in this chapter is the following: can one develop a generative model of information propagation (in the presence of polarization) that allows developing an unsupervised algorithm to automatically distinguish the above three categories of tweets, without prior training, and without knowledge of language or vocabulary? Importantly, can such an algorithm offer assurances in results? The main idea behind the generative model is that people tend to propagate information that matches their beliefs. Hence, by observing how information propagates, it may be possible to solve the inverse problem; namely, recover which side the information favors,

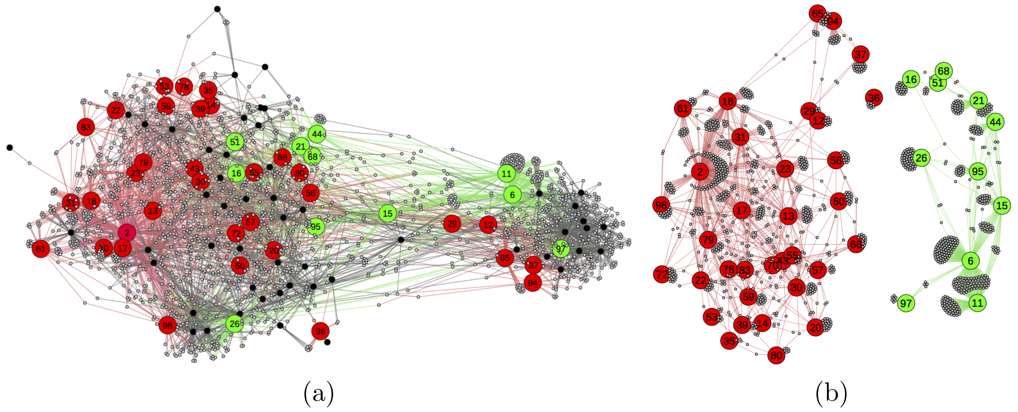


Figure 4.1: (a) Crawled data does not reveal polarization structure, (b) Manually filtering out the nonpolarized sources and assertions reveal polarization.

or tell if it is impartial.

The problem is complicated by the fact that polarized sources, besides sharing information that confirms their bias, usually also share neutral information about the situation [4, 5, 148, 153]. Moreover, neutral sources may exist that selectively share elements from both points of view [7, 160]. Therefore, no clean separation exists between clusters of sources based on what they propagate.

Figure 4.1a illustrates the bipartite graph between the sources and assertions they make (i.e., showing who shares what) collected in Egypt after a democratically-elected president was deposed. For visual clarity, only the largest 100 cascades of information propagated through the social media are shown. Manually graded assertions that were in favor of the president, and against the president, are shown in large red and green circles, respectively. The black circles are the neutral assertions. The smallest grey circles denote the sources. Figure 4.1b shows the same network after removing all neutral sources and claims. The remaining graph clearly separates well-connected clusters of nodes that espouse the different sides of the story. The edges in those subgraphs represent the opinions partial to the respective side. Note however that those clusters are different than what would have been obtained by directly clustering the original graph in Figure 4.1a. The challenge is to identify polarized opinions automatically by analysis of the original graph, without the benefit of manual labeling, prior training, or use of language-

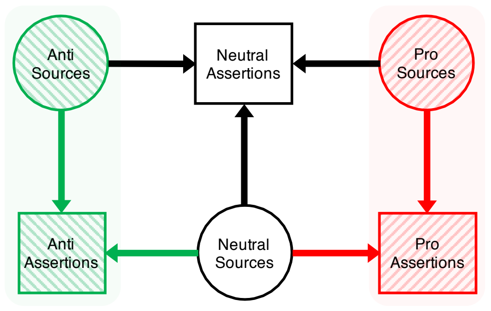


Figure 4.2: Modeling a polarized source-assertion network by identifying different types of edges

specific features.

The *polarity set* is defined as the set of different attitudes, or *polarities*, corresponding to a situation that involves a conflict or debate, which has been termed as the polarity context or *pole* [7]. Consider the situation regarding EU referendum vote [161]. The polarity set can have two polarities {`VoteLeave`, `VoteRemain`}, relating to the campaigns regarding the decision. In the following, we develop models for the common case of two polarities only. The extensions to more polarities, however, are straightforward. In all cases, absence of membership to a particular polarity makes a source or assertion neutral.

To develop the polarity models, we use the terminology {`pro`, `anti`} to denote the two polarities [4,5], regardless of the polarity context. Note that the source-assertion network is a bipartite graph. In the presence of a neutral group obscuring the homophilic tendency of the `pro` and `anti` polarities, the source-assertion network can be modeled as in figure 5.1b. The `anti`, `neutral`, and `pro` networks are shown in left, middle, and right, respectively. Note that there are seven different type of edges in our model, which corresponds to the seven thick arrows emanating from the sources and terminating at the assertions. Because of the partisanship in the network, the `pro` sources do not claim `anti` assertions, and the `anti` sources do not claim `pro` assertions. Therefore the model excludes such edges.

Below, we discuss different models that define behavior of polarized sources, and use them to solve the inverse problem; namely, given the observed source assersion graph, determine the underlying polarities of sources and assertions (or lack thereof). Our models differ in the degree to which behaviors of polarized sources are individualized. There are generally two issues in describing source behavior. First, should individial sources have individualized parameters describing individualized degrees of polarization (that dictate different probabilities of emitting biased versus unbiased claims), or can sources of the same polarity be generally described by the same polarity-specific parameters? Second, should individual sources have individualized parameters describing the blogging probability in geeneral, or could they be described by community-wide parameters as well? Those differences give rise to multiple models, presented below.

4.3 Individualized Bias with Individualized Blogging Rate

Consider a scenario with two polarities, namely `pro` and `anti`. Some of the observations posted in the social media favors or averts the `pro` camp. Some of the observations do the same for the `anti` camp. Also consider the presence of `neutral` assertions which are not favoring any of the polarized camps. There are also `neutral` sources who forward polarized assertions of both polarity or neutral assertions, possibly based on their factual significance. In this chapter, our goal is to separate the polarities.

Consider $\Pr(SC_{ij})$, probability that source i shares an observation j. Then $\Pr(Z_j^l)$ denotes the probability that assertion j is of polarity l. $\Pr(Y_i^k)$ denotes the probability that source i is of polarity k. K and L represent the set of polarities $\{neu, pro, anti\}$ for the sources and the assertions, respectively. Therefore $k \in K$, and $l \in L$. We can then write equation 5.1 by using the total probability theorem for $\Pr(SC_{ij})$.

$$\Pr(SC_{ij}) = \sum_{(k,l)\in K\times L} \Pr(SC_{ij}|Y_i^k, Z_j^l).\Pr(Y_i^k, Z_j^l) \tag{4.1}$$

When the assertion opposes the polarity of a source, the source is not going to share it. Therefore, both $\Pr(SC_{ij}|Y_i^{pro}Z_j^{anti})$ and $\Pr(SC_{ij}|Y_i^{anti}Z_j^{pro})$ reduces to 0. Note that, the neutral group is not opposing to either pro or anti groups. Therefore, neither $\Pr(SC_{ij}|Y_i^{neu}Z_j^l)$, nor $\Pr(SC_{ij}|Y_i^kZ_j^{neu})$ reduce to 0 in general.

In case of social-sensing, the polarity of the source or the assertion is unknown during data collection. The only information known is which source

Table 4.1: Parameters representing $a_i^{kl} = \Pr(SC_{ij}, \theta | Y_i^k, Z_j^l)$

Source Y_i	Assertion Z_j	$SC_{ij} = 1$	$SC_{ij} = 0$
neutral	neutral	a_i^{nn}	$1 - a_i^{nn}$
neutral	pro	a_i^{np}	$1 - a_i^{np}$
neutral	anti	a_i^{na}	$1 - a_i^{na}$
pro	neutral	a_i^{pn}	$1 - a_i^{pn}$
pro	pro	a_i^{pp}	$1 - a_i^{pp}$
pro	anti	0	1
anti	neutral	a_i^{an}	$1 - a_i^{an}$
anti	pro	0	1
anti	anti	a_i^{aa}	$1 - a_i^{aa}$

claimed which assertion. Therefore, the polarities are latent states. Each source can have seven (possibly non-zero) parameters related to its chance of forwarding different type of assertions based on its own polarity and the polarity of the assertion.

Suppose there are m sources and n assertions. The source-assertion network SC is bipartite graph from the sources to the assertions, where $SC_{ij} = 1$ means source i made claim j, and $SC_{ij} = 0$ means source i did not make claim j. This network represents the set of known observatins. Our goal is to jointly estimate the source and the assertion polarities. Mathematically, the problem is to estimate the distribution of $\Pr(Y_i^k)$ for each source, and the distribution of $\Pr(Z_j^l)$ for each assertion.

Next, we cast the problem of computing the distribution of each source and assertion to particular polarities as a maximum likelihood estimation problem from the given observations, $SC = [sc_{ij}]$, which is a binary relation representing whether source i made an assertion j or not. As discussed, we have seven unknown parameters for each source. Let us define the vector θ, which represents the unknown parameters of the problem. Table 4.1 shows the parameters for each source $a_i^{kl} = \Pr(SC_{ij}, \theta | Y_i^k, Z_j^l)$ for each case of source and assertion polarities.

$$\theta = [a_i^{kl} : (k, l) \in K \times L, i \in \{1..m\}] \tag{4.2}$$

The polarities of the source and the assertions are considered as latent states. $Y_i^k = 1$, if source i is of polarity k, and $Y_i^k = 0$ otherwise. $Z_j^l = 1$ if assertion j is of polarity l, and $Z_j = 0$ otherwise. A maximum likelihood estimator is used to estimate the unknown parameters that is maximally consistent with the given observations, i.e. maximizes the probability of observations SC. In another words, we want to find such θ that maximizes $\Pr\{SC|\theta\}$.

Given a likelihood function, an expectation maximization (EM) algorithm can be used to solve the problem. The algorithm starts with some initial guess θ_0 for the parameter vector θ, and then iteratively performs the following two steps until convergence.

- Computes the expected values for the latent states using the given observations and the present guess of the parameter vector θ.
- From the computed expected values, it finds a new parameter vector that maximizes the likelihood function.

4.3.1 Likelihood Function

The likelihood function computes the probability of the known observations to happen for a particular set of parameters. The known observations for our problem is SC, the source-assertion network. Therefore the likelihood function $\mathcal{L}$ is defined by equation 4.3. The expansion on equation 4.4 allows to compute it using the present value of the latent states.

$$\begin{aligned}
\mathcal{L}(\theta; SC) &= \Pr(SC|\theta) && (4.3)\\
&= \sum_{Y,Z} \Pr(SC, Y, Z|\theta) && (4.4)\\
&= \sum_{Y,Z} \prod_{i=1}^{m} \prod_{j=1}^{n} \Pr(SC_{ij}, Y, Z, \theta) = \prod_{i=1}^{m} \prod_{j=1}^{n} \sum_{Y,Z} \Pr(SC_{ij}, Y, Z, \theta) &&\\
&= \prod_{i=1}^{m} \prod_{j=1}^{n} \sum_{(k,l)\in K\times L} \Pr(SC_{ij}, Y_i^k, Z_j^l, \theta) && (4.5)
\end{aligned}$$

Maximizing the likelihood function is equivalent to maximizing the log-likelihood function. Therefore, we derive equation 4.6:

$$\log \mathcal{L}(\theta; SC) = \sum_{i=1}^{m} \sum_{j=1}^{n} \log \sum_{(k,l)} \Pr(SC_{ij}, Y_i^k, Z_j^l, \theta) \qquad (4.6)$$

Expectation of the log-likelihood function is derived in equation 4.7.

$$\mathbb{E}[\log \mathcal{L}(\theta; SC)] = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{(k,l)} \Pr(Y_i^k, Z_j^l | \theta) \log \Pr(SC_{ij} | Y_i^k, Z_j^l, \theta) \tag{4.7}$$

4.3.2 Expectation Step

In the expectation step, the latent states are estimated using the given observations and the present guess of the parameter vector θ. Therefore we want to find:

$$\begin{aligned} &\forall 1 \leq i \leq m, 1 \leq j \leq n, \\ &(k,l) \in K \times L : \Pr(Y_i^k, Z_j^l | SC, \theta) \end{aligned} \tag{4.8}$$

We use Bayes' theorem to determine the expression for $\Pr(Y_i^k, Z_j^l | \theta, SC)$

$$\Pr(Y_i^k, Z_j^l | \theta, SC) = \frac{\Pr(SC, \theta | Y_i^k, Z_j^l). \Pr(Y_i^k, Z_j^l)}{\Pr(SC, \theta)} \tag{4.9}$$

Note that total probability theorem can be used to define $\Pr(SC, \theta) = \sum_{(k,l) \in K \times L} \Pr(SC, \theta | Y_i^k, Z_j^l). \Pr(Y_i^k, Z_j^l)$ Therefore, we write equation 4.10

$$\Pr(Y_i^k, Z_j^l | \theta, SC) = \frac{\Pr(SC, \theta | Y_i^k, Z_j^l). \Pr(Y_i^k, Z_j^l)}{\sum_{(k,l)} \Pr(SC, \theta | Y_i^k, Z_j^l). \Pr(Y_i^k, Z_j^l)} \tag{4.10}$$

Note that the expression $\Pr(Y_i^k, Z_j^l)$ does not involve SC or θ. It is apriori probability of source i to be of polarity k, and assertion j to be of polarity l. The apriori probabilities can be considered equal and hence we can omit them, and write the expression for the expectation step in equation 4.11

$$\Pr(Y_i^k, Z_j^l | \theta, SC) = \frac{\Pr(SC, \theta | Y_i^k, Z_j^l)}{\sum_{(k,l)} \Pr(SC, \theta | Y_i^k, Z_j^l)} \tag{4.11}$$

Given the expressions in equation 4.10 or equation 4.11, the exact expectation step depends on how we formulate $\Pr(SC, \theta | Y_i^k, Z_j^l)$. If a particular source i belongs to polarity group k, and a particular assertion j belongs to polarity group l, it estimates the chance of the given observations SC happening using the present parameter vector θ. In this chapter, we consider three different formulations based on different assumptions.

Here we describe a simple expression for $\Pr(SC, \theta | Y_i^k, Z_j^l)$. As a first assumption, we consider that the given conditions that source i belongs to polarity k and assertion j belongs to polarity l only affects whether source i makes assertion j or not. Therefore, only the presence of the particular (i, j) edge, i.e. SC_{ij} is affected by the premises. Therefore, we can write equation 4.12.

$$\begin{aligned}\Pr(Y_i^k, Z_j^l | \theta, SC) &= \frac{\Pr(SC, \theta | Y_i^k, Z_j^l)}{\sum_{(k,l)} \Pr(SC, \theta | Y_i^k, Z_j^l)} \\ &= \frac{\Pr(SC_{ij}, \theta | Y_i^k, Z_j^l)}{\sum_{(k,l)} \Pr(SC_{ij}, \theta | Y_i^k, Z_j^l)} \\ &= \frac{[a_i^{kl}]^{SC_{ij}}.[1 - a_i^{kl}]^{1-SC_{ij}}}{\sum_{(k,l)} [a_i^{kl}]^{SC_{ij}}.[1 - a_i^{kl}]^{1-SC_{ij}}} \qquad (4.12)\end{aligned}$$

Note that the final expression in equation 4.12 is constructed by consulting Table 4.1. Consider the numerator. $[a_i^{kl}]^{SC_{ij}}.[1 - a_i^{kl}]^{1-SC_{ij}}$. The format of this expression represents a conditional statement. Because SC_{ij} is binary, one of the two product terms will reduce to 1. Therefore, if the observation $SC_{ij} = 1$, the numerator will evaluate to a_i^{kl}, and if the observation $SC_{ij} = 0$, the numerator will evaluate to $1 - a_i^{kl}$.

4.3.3 Maximization Step

The maximization step can be derived by maximizing the expectation of the log-likelihood function. Therefore, we want to maximize equation 4.7. We rewrite the equation with the expression for $\Pr(SC_{ij} | Y_i^k, Z_j^l, \theta)$ for our formulation.

$$\begin{aligned}\mathbb{E}[\log \mathcal{L}(\theta; SC)] &= \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{(k,l)} \Pr(Y_i^k, Z_j^l | \theta). \log([a_i^{kl}]^{SC_{ij}}.[1 - a_i^{kl}]^{1-SC_{ij}}) \\ &= \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{(k,l)} \Pr(Y_i^k, Z_j^l | \theta).[SC_{ij} \log a_i^{kl} + (1 - SC_{ij}) \log(1 - a_i^{kl})] \qquad (4.13)\end{aligned}$$

We differentiate $\mathbb{E}[\log \mathcal{L}(\theta; SC)]$ with respect to each of the parameters, and set those to 0. Solving for a_i^{kl}, equation 4.14 gives the final expression to update the parameter so that it maximizes the likelihood function.

$$\begin{aligned}
\frac{\partial \mathbb{E}}{\partial a_i^{kl}} &= \sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta).\Big[\frac{SC_{ij}}{a_i^{kl}} - \frac{1 - SC_{ij}}{1 - a_i^{kl}}\Big] = 0 \\
\Rightarrow \sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta)\frac{SC_{ij}}{a_i^{kl}} &= \sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta)\frac{1 - SC_{ij}}{1 - a_i^{kl}} \\
\Rightarrow \frac{a_i^{kl}}{1 - a_i^{kl}} &= \frac{\sum_{j=1}^{n} SC_{ij} \Pr(Y_i^k, Z_j^l|\theta)}{\sum_{j=1}^{n} (1 - SC_{ij}) \Pr(Y_i^k, Z_j^l|\theta)} \\
\Rightarrow \qquad a_i^{kl} &= \frac{\sum_{j=1}^{n} SC_{ij} \Pr(Y_i^k, Z_j^l|\theta)}{\sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta)} \qquad (4.14)
\end{aligned}$$

So the final algorithm using our first formulation is using equation 4.12 and equation 4.14 repeatedly. At step t, equation 4.12 is applied for every possible source-assertion pair to estimate the joint distribution $[Y, Z]_t$ of the sources and the assertions to different polarity groups using the current estimate of the parameter vector θ_{t-1}. Equation 4.14 is then applied to every source to update the seven a_i^{kl} parameters, resulting in a new parameter vector θ_t. The iterations continue until convergence.

4.3.4 Enhanced Expectation Step

In this section, we analyze some of our assumptions used in the derived model. To derive equation 4.12, we assumed that the condition that source i is of polarity k, and the assertion j is of polarity l affects only the (i, j) edge in the source-assertion network. It made the derivation of the expectation step easier, but we hypothesize that this is an over-simplified assumption. For example, if a source is `pro`, it increases the chance of all the other assertions it made to be `pro` and increases the chance for all the assertions it did not make to be `anti`. Likewise, if an assertion is `pro`, it increases the chance of all the sources who made this claim to be `pro`, and increases the chance of all the sources who did not make this claim to be `anti`. Therefore, we need to update the expression for the expectation step to include not only the (i, j) edge for a particular source-assertion pair, rather consider all the edges where either i or j is incident. Figure 4.3 explains this mechanism. Here S_3

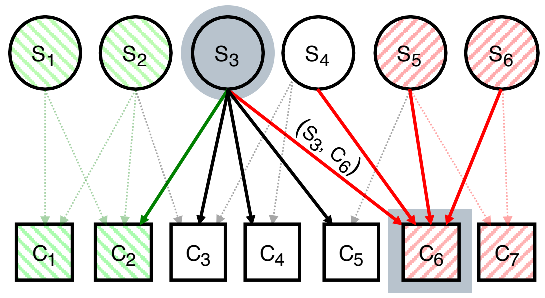


Figure 4.3: Highlighted edges are considered when running expectation step for S_3 and C_6

and C_6 are the selected source and the assertion. Edges that are considered for the (S_3, C_6) pair are highlighted, edges that exist in the network, but are not considered are dimmed in this figure.

Based on this argument, we derive a new expectation step. We consider the source-assertion pairs as independent. Therefore, the joint probability of the observed data SC can be computed as individual probability of presence or absence of each of the source-assertion pairs.

$$\begin{aligned}\Pr(Y_i^k, Z_j^l|\theta, SC) &= \frac{\Pr(SC, \theta|Y_i^k, Z_j^l)}{\sum_{(k,l)} \Pr(SC, \theta|Y_i^k, Z_j^l)} \\ &= \frac{\prod_{i',j'} \Pr(SC_{i',j'}|Y_i^k Z_j^l)}{\sum_{k,l} \prod_{i',j'} \Pr(SC_{i',j'}|Y_i^k Z_j^l)}\end{aligned} \tag{4.15}$$

In equation 4.15, cases where both $i' \neq i$ and $j' \neq j$ can be considered as independent of Yi^k and Z_j^l. This is because of the argument illustrated in figure 4.3. The chance of a different source i' making a different assertion j' is unrelated to the polarity of i and j. Therefore we write,

$$\Pr(Y_i^k, Z_j^l|\theta, SC) = \frac{\prod\limits_{(i'=i)\vee(j'=j)} \Pr(SC_{i',j'}|Y_i^k, Z_j^l)}{\sum\limits_{k,l} \prod\limits_{(i'=i)\vee(j'=j)} \Pr(SC_{i',j'}|Y_i^k, Z_j^l)} = \frac{\mathcal{G}(i,j,k,l)}{\sum\limits_{k,l} \mathcal{G}(i,j,k,l)} \tag{4.16}$$

$$\begin{aligned}&\mathcal{G}(i,j,k,l) = [a_i^{kl}]^{SC_{ij}}.[1 - a_i^{kl}]^{1-SC_{ij}} \\ &.\prod_{j'\neq j} [\sum_{l'\in L} a_i^{kl'}.\Pr(\hat{Z}_{j'}^{l'})]^{SC_{ij'}} [\sum_{l'\in L} (1 - a_i^{kl'}).\Pr(\hat{Z}_{j'}^{l'})]^{1-SC_{ij'}}\end{aligned}$$

$$.\prod_{i' \neq i} [\sum_{k' \in K} a_{i'}^{k'l}. \Pr(\hat{Y}_{i'}^{k'})]^{SC_{i'j}} [\sum_{k' \in K} (1 - a_{i'}^{k'l}). \Pr(\hat{Y}_{i'}^{k'})]^{1-SC_{i'j}} \tag{4.17}$$

Here K and L represent the set of polarities $\{neu, pro, anti\}$ for the sources and the assertions, respectively. Equation 4.16 presents the new expectation step, where $\mathcal{G}$ is defined in equation 4.17. $\Pr(\hat{Z}_{j'}^{l'})$ and $\Pr(\hat{Y}_{i'}^{k'})$ inside equation 4.17 represent current estimate of $\Pr(Z_{j'}^{l'})$ and $\Pr(Y_{i'}^{k'})$, computed by marginalizing $\Pr(Y_i^k, Z_j^l)$ obtained from the previous iteration. The maximization step remains to be the one described in equation 4.14.

4.4 Community-wide Bias with Individualized Blogging Rate

We observe that there is a group behavior for the affinity of the sources to particular assertions, and vice versa. Inspired by figure 5.1b, we hypothesize that instead of seven parameters a_i^{kl} per source, it might be sufficient to have seven parameters a^{kl} per polarized group. a^{kl} represents the chance of an assertion claimed by a source from polarity k to be of polarity l, defined by equation 4.18.

$$a^{kl} = \Pr(Z_j^l | Y_i^k, SC_{ij}) \tag{4.18}$$

The parameters a_i^{kl} for each source can be represented using group parameter a^{kl} and the blogging rate of that source R_i. R_i is blogging rate of a source, and it is independent of the polarity of the source. Some sources are highly active in the social media, and some are relatively less active. Independent of its activity level, a source can be either `pro`, `anti`, or `neutral` source. R_i is defined in equation 4.19. We estimate it by the ratio of number of assertions claimed by source i and total number of assertions. Additionally, we define d^l as apriori probablity of an assertion to belong to particular polarity l, the blogging rate of the source S_i. d^l is an additional input to the algorithm provided as background information, and R_i is inferred from the given data, SC.

$$R_i = \Pr(SC_{ij}) \tag{4.19}$$

$$d^l = \Pr(Z_j^l) \tag{4.20}$$

Blogging rate represents how active a source is. Because blogging rate of a source S_i does not depend on its polarity, we can trivially write $R_i = \Pr(SC_{ij}) = \Pr(SC_{ij}|Y_i^k)$. Therefore, we can derive equation 4.22:

$$\begin{aligned} a^{kl}R_i &= \Pr(Z_j^l|Y_i^k, SC_{ij}) \Pr(SC_{ij}) \\ &= \Pr(Z_j^l|Y_i^k, SC_{ij}) \Pr(SC_{ij}|Y_i^k) \qquad (4.21) \\ &= \Pr(Z_j^l, SC_{ij}|Y_i^k) \\ &= \Pr(Z_j^l|Y_i^k) \Pr(SC_{ij}|Y_i^k, Z_j^l) \qquad (4.22) \end{aligned}$$

Using total probability theorem, we can condition $\Pr(Z_j^l|Y_i^k)$ based on the presence or absence of (i, j) edge in the source-assertion network, i.e. whether $SC_{ij} = 1$ or $SC_{ij} = 0$.

$$\begin{aligned} \Pr(Z_j^l|Y_i^k) &= \Pr(Z_j^l|Y_i^k, SC_{ij}) \Pr(SC_{ij}|Y_i^k) + \Pr(Z_j^l|Y_i^k, \overline{SC_{ij}}) \Pr(\overline{SC_{ij}}|Y_i^k) \\ &= a^{kl}R_i + \Pr(Z_j^l|Y_i^k, \overline{SC_{ij}})(1 - R_i) \qquad (4.23) \end{aligned}$$

Note that, $\Pr(Z_j^l|Y_i^k, \overline{SC_{ij}})$ represents the probability of an assertion j to belong to polarity l, when source i of polarity k did not claim it. Not claiming an assertion does not provide much information. Therefore, we estimate this to be the apriori probability of an assertion to belong to polarity group l, i.e. $d^l = \Pr(Z_j^l)$. Therefore, by substituting equation 4.23 into equation 4.22, we find

$$\Pr(SC_{ij}|Y_i^k, Z_j^l) = \frac{a^{kl}R_i}{a^{kl}R_i + d^l(1 - R_i)} \tag{4.24}$$

In this new formulation, the expectation step stays the same as the previous one as defined in equation 4.16 and equation 4.17. Note that $a_i^{kl} = \Pr(SC_{ij}|Y_i^k, Z_j^l)$. Therefore equation 4.24 is used to find the source parameters a_i^{kl} from the group parameters a^{kl} and per source blogging rate R_i. The source parameters are then used in equation 4.16 during expectation step.

We derive the maximization step by maximizing the expectation of the log-likelihood function. We want to maximize equation 4.7. We rewrite the equation with the expression for $\Pr(SC_{ij}|Y_i^k, Z_j^l, \theta)$ for our formulation.

$$\begin{aligned}
&\mathbb{E}[\log \mathcal{L}(\theta; SC)] = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{(k,l)} \Pr(Y_i^k, Z_j^l|\theta).\\
&\log\Big([\frac{a^{kl}R_i}{a^{kl}R_i + d^l(1-R_i)}]^{SC_{ij}}.[\frac{d^l(1-R_i)}{a^{kl}R_i + d^l(1-R_i)}]^{1-SC_{ij}}\Big)\\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{(k,l)} \Pr(Y_i^k, Z_j^l|\theta).\big[SC_{ij}\log\frac{a^{kl}R_i}{a^{kl}R_i + d^l(1-R_i)}\\
&\quad +(1-SC_{ij})\log\frac{d^l(1-R_i)}{a^{kl}R_i + d^l(1-R_i)}\big]\\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{(k,l)} \Pr(Y_i^k, Z_j^l|\theta).\big[SC_{ij}\log(a^{kl}R_i)\\
&\quad +(1-SC_{ij})\log\{d^l(1-R_i)\} - \log\{a^{kl}R_i + d^l(1-R_i)\}\big] \qquad (4.25)
\end{aligned}$$

We differentiate $\mathbb{E}[\log \mathcal{L}(\theta; SC)]$ with respect to each of the parameters, and set those to 0.

$$\begin{aligned}
&\frac{\partial \mathbb{E}}{\partial a^{kl}} = \sum_{i=1}^{m}\sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta).\big[\frac{SC_{ij}}{a^{kl}} - \frac{R_i}{a^{kl}R_i + d^l(1-R_i)}\big] = 0\\
&\Rightarrow \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{SC_{ij}\Pr(Y_i^k, Z_j^l|\theta)}{a^{kl}} = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{R_i \Pr(Y_i^k, Z_j^l|\theta)}{a^{kl}R_i + d^l(1-R_i)}\\
&\Rightarrow \sum_{i=1}^{m}\sum_{j=1}^{n} SC_{ij}\Pr(Y_i^k, Z_j^l|\theta) = \sum_{i=1}^{m}\sum_{j=1}^{n} \frac{a^{kl}R_i \Pr(Y_i^k, Z_j^l|\theta)}{a^{kl}R_i + d^l(1-R_i)} \qquad (4.26)
\end{aligned}$$

Therefore, we solve the nonlinear equation 4.26 for each of the seven group parameters a^{kl}. This gives the updated parameter vector maximizing the likelihood function. The updated parameters are then used to determine the source parameters using equation 4.24, which are put into equation 4.16 to update the latent states $[Y, Z]$. We continue iterating in this manner until convergence is reached.

Note that d^l can also be considered as a parameter. In that case, we differentiate $\mathbb{E}[\log \mathcal{L}(\theta; SC)]$ with respect to d^l, and set it to 0.

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k\in K} (1-SC_{ij})\Pr(Y_i^k, Z_j^l|\theta) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k\in K} \frac{d^l(1-R_i)\Pr(Y_i^k, Z_j^l|\theta)}{a^{kl}R_i + d^l(1-R_i)} \qquad (4.27)$$

In this mechanism, the maximization step becomes solving the system of equations 4.26 and 4.27 for seven a^{kl} parameters, and three d^l parameters.

4.5 Community-wide Bias with Community-wide Blogging Rate

In this section we develop the formulation when the blogging rate is not individualized. It is specific to different polarities. Therefore, in this model, a^{kl} represents $P(SC_{ij}|Y_i^k, Z_j^l)$, which remains same throughout all (i,j) source-assertion pairs with polarity (k,l). The expectation step is followed from equation 4.16 with $a_i^{kl} = a^{kl}$ for each i. To derive the maximization step, we write equation 4.28. We differentiate and set to 0 with respect to a^{kl}. Solving for a^{kl}, equation 4.29 gives the final expression to update the parameter vector so that it maximizes the likelihood function.

$$\begin{aligned}
&\mathbb{E}[\log \mathcal{L}(\theta; SC)] \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{(k,l)} \Pr(Y_i^k, Z_j^l|\theta). \log([a^{kl}]^{SC_{ij}}.[1-a^{kl}]^{1-SC_{ij}}) \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{(k,l)} \Pr(Y_i^k, Z_j^l|\theta).[SC_{ij}\log a^{kl} + (1-SC_{ij})\log(1-a^{kl})] \qquad (4.28)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathbb{E}}{\partial a^{kl}} &= \sum_{i=1}^{m}\sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta).\Big[\frac{SC_{ij}}{a^{kl}} - \frac{1-SC_{ij}}{1-a^{kl}}\Big] = 0 \\
\Rightarrow \sum_{i=1}^{m}\sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta)\frac{SC_{ij}}{a^{kl}} &= \sum_{i=1}^{m}\sum_{j=1}^{n} \Pr(Y_i^k, Z_j^l|\theta)\frac{1-SC_{ij}}{1-a^{kl}} \\
\Rightarrow \qquad a^{kl} &= \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} SC_{ij}\Pr(Y_i^k, Z_j^l|\theta)}{\sum_{i=1}^{m}\sum_{j=1}^{n}\Pr(Y_i^k, Z_j^l|\theta)} \qquad (4.29)
\end{aligned}$$

This model, therefore, iterates between equation 4.16 and equation 4.29 until convergence.

Table 4.2: Parameters of the simulated network

Parameter	Default
Probability of any source making `neu` claim	0.3
Probability of `pro` source making `pro` claim	0.9
Probability of `anti` source making `anti` claim	0.9
Distribution of `neu`, `pro`, and `anti` assn.	$\{0.34, 0.33, 0.33\}$
Distribution of `neu`, `pro`, and `anti` source	$\{0.34, 0.33, 0.33\}$

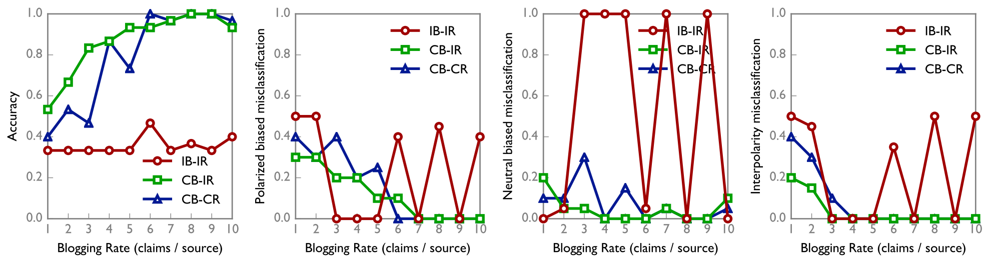


Figure 4.4: Quality of estimation vs. blogging rate

4.6 Simulation Study

In this section, we simulate the three models developed in the earlier sections and analyze their performance. We have written a network generator using `C++` that generates polarized source-assertion networks with known polarities for the nodes. The simulator requires number of sources, number of assertions, and average blogging rate to produce a network. It has several tunable parameters. Table 4.2 shows the parameters. In the presented results, the model from Section 4.3 is represented as `IB-IR` (individual bias with individual blogging rate), the model from section 4.4 as `CB-IR` (community bias with individual blogging rate), and the model from section 4.5 as `CB-CR` (community bias and blogging rate).

4.6.1 Varying Blogging Rate

Figure 4.4 shows results for the three models with respect to blogging rate. The number of assertions was chosen to be 30, and the number of sources was chosen to be 300. Note that such a ratio of sources to assertions comes from the observations on the Twitter collected datasets. The average number of claims per source was varied between 1 to 10. Figure 4.4 shows four

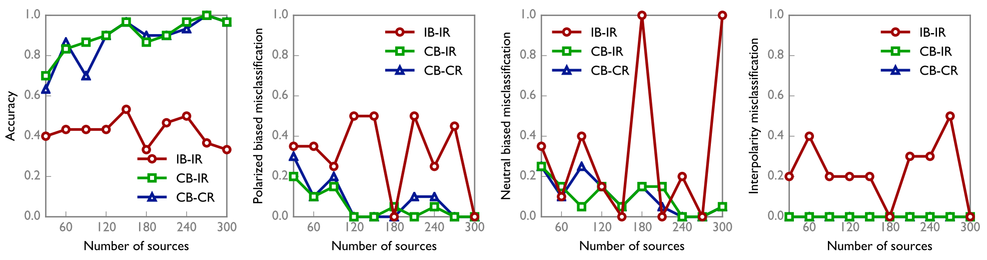


Figure 4.5: Quality of estimation vs. number of sources (average blogging rate = 7)

charts related to the quality of classification. The leftmost chart presents the accuracy of the algorithms, which is the ratio of assertions correctly classified. The next chart presents the cases where the algorithms misclassified a neutral assertion as polarized. The third chart presents the cases where a polarized assertion was estimated to be neutral. The rightmost chart presents the misclassifications within the polarity groups, i.e. `pro` estimated as `anti`, and vice versa.

In all cases, `CB-IR` is found to be superior. The model `IB-IR`, the one with individual bias and blogging rate, does not perform well because it has too many parameters that are estimated. Therefore, the model could converge in many ways. The model `CB-CR` also performs reasonably well, but worse than `CB-IR`. `CB-CR` has too few parameters, therefore, the algorithm is often making compromises. Note that the misclassifications reduce for both `CB-IR` and `CB-CR` as the blogging rate increases.

4.6.2 Varying Number of Sources

In this section, we test the models by varying number of sources from 30 to 300. We present two cases, one with higher blogging rate, and the another with lower blogging rate. Figure 4.5 shows results for the three models when the average number of claims per source was 7. As expected from the previous section, `IB-IR` can converge to many solutions. As the number of sources increase, both `CB-IR` and `CB-CR` starts performing better.

The trend is similar in figure 4.6, which shows results when the average number of claims per source is 3. However, now the algorithms require more sources to perform better. Note that the difference between the two algorithms with community parameters, `CB-IR` and `CB-CR` is larger here. Lower

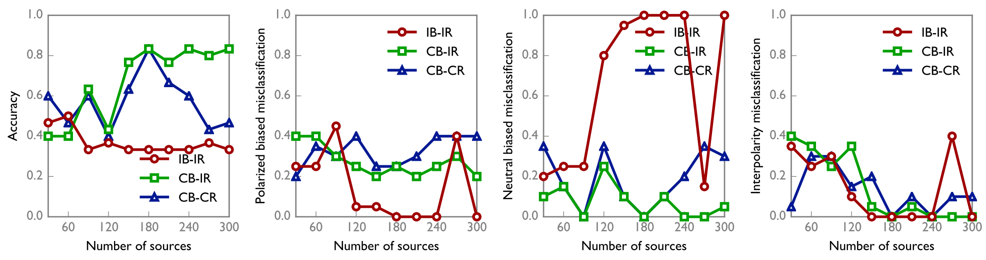


Figure 4.6: Quality of estimation vs. number of sources (average blogging rate = 3)

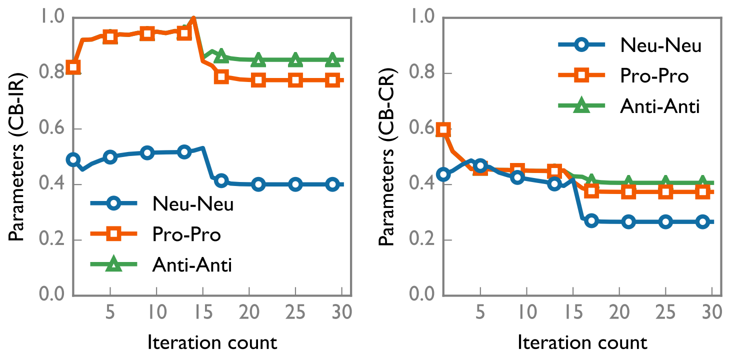


Figure 4.7: Convergence properties of the models with community bias

blogging rate gives the models less information per sources to converge to, therefore, considering individual blogging rate translates to better quality of estimation. On the other hand, `CB-CR` mixes blogging rate in the single a^{kl} parameters, so it has to make compromise for many sources.

4.6.3 Convergence Properties

In this experiment, we consider the convergence properties for the two algorithms with community wide parameters `CB-IR`, and `CB-CR`. Figure 4.7 plots values along the main diagonal of the parameter matrix, i.e. a^{nn}, a^{pp}, and a^{aa} parameters. Note that around 15 to 20 iterations was enough for the algorithms to converge for the simulated networks.

From the simulation experiments, it is clear that `CB-IR` performs better than the other models. Rest of this chapter evaluates and compares this model to other candidates using Twitter as the social network.

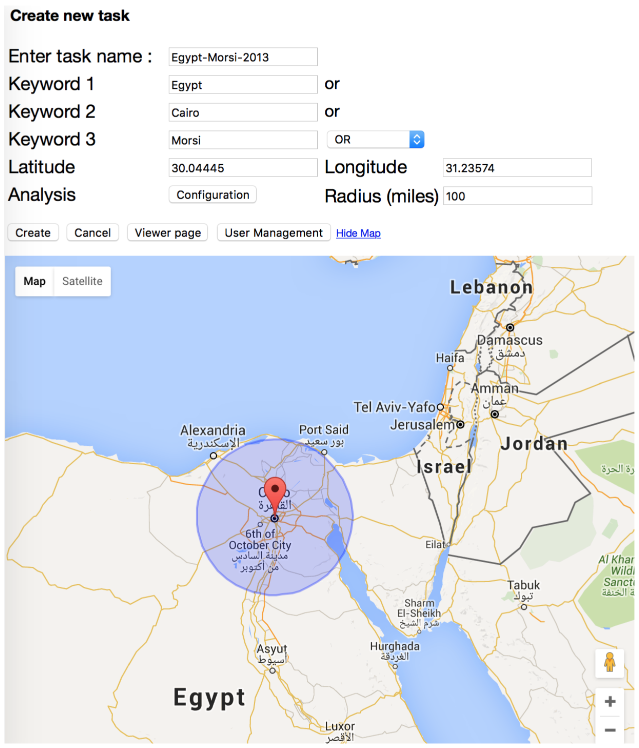
4.7 Evaluation

In this section, we evaluate the polarization models using real-world events tracked from Twitter. Following the simulation study presented in section 4.6, we choose the model with community bias and individualized blogging rate (Section 4.4) to evaluate using the crawled tweets. The model was implemented using Java 1.8. Goal of the evaluation is judge the efficacy of model to track controversial situations. Therefore, the converged model is used to obtain a separation of the polarized tweets, which were manually annotated by human graders beforehand, to indicate whether it is `pro` or `anti`. The polarized tweets received from the model is then evaluated on the goodness of the separation. For this purpose, receiver operating characteristics (ROC) curve is used. ROC plots true positive rate vs. false positive rate. We plot rate of pro vs. rate of anti. For this evaluation, having the area of 1 under the ROC translates to an optimal algorithm, and having the area of 0.5 translates to a random algorithm. We also compare the output obtained from our model with that of some candidate techniques.

Metis-2 and Metis-3 Various variants of community detection has been used in the literature. We use Metis [162] as the readily available tool for graph partitioning. Metis-2 represents when we partition the graph in two parts, and Metis-3 represents when we partition the graph in three parts. Note that, the sequence of the three partitions that result in a higher ROC has been presented.

EM Fact-finder These are also EM based maximum likelihood algorithms [3] to uncover likely facts from social media, also known as veracity analysis. Fact-finders are not associated with polarity of the tweets, but they would perform well if a particular polarity of the opposing groups is more likely to post more facts.

Sentiment Analysis A few eariler works use sentiment analysis based tools to analyze controversial issues in social media [158, 159]. We used Umigon [157], which is a sentiment analysis tool trained for tweets. It is readily available through API.



Create new task

Enter task name :	Egypt-Morsi-2013		
Keyword 1	Egypt	or	
Keyword 2	Cairo	or	
Keyword 3	Morsi	OR	
Latitude	30.04445	Longitude	31.23574
Analysis	Configuration	Radius (miles)	100

Figure 4.8: Crawling `Egypt` dataset using Apollo Social Sensing Toolkit

4.7.1 Data Collection and Preprocessing

Apollo Social Sensing Toolkit [3, 9] was used to collect tweets in real-time. Apollo [9] presents a interface for users to specify interests to start collecting tweets using Twitter search api [163]. We started several crawling tasks during various polarized situations. For example, during 2013 political unrest in Egypt (2013), figure 4.8 illustrates the apollo task At every configurable interval, the set of collected tweets is forwarded to further down pipeline for further analysis, which also includes polarization analysis. The collection of recorded traces was clustered based on text similarity to generate a representative summary [8]. Clustering is required to (a) account for variations in the text asserting the same information, (b) generate a subset of tweets for further processing via various ranking mechanims, and (c) generate cascades regarding how information in particular tweets propagated across Twitter. Because the distribution of the cluster sizes is approximately heavy tailed, considering the largest clusters can sufficiently represent the larger set of tweets [4]. We considered the largest 100 clusters and manually annotated those in relation to the polarity group to obtain the ground truth. The tweets

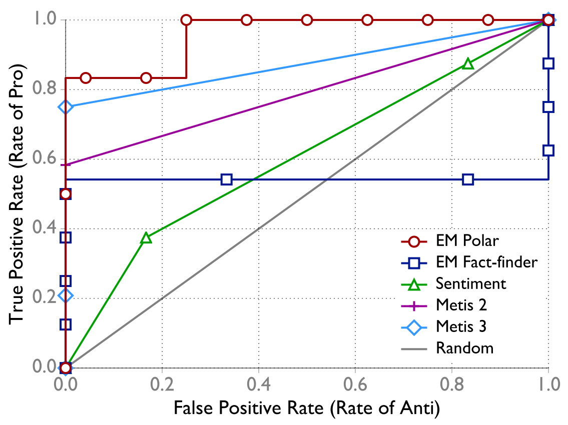


Figure 4.9: Receiver operating characteristics (Trump)

are in JSON [164] format, which also includes various meta-information inclduing the user authoring the tweet (source) and creation time of the tweet. The source information extracted are used to create the source nodes in the source-assertion network. On the other hand, the hierarchy provides assertion information. Different components of the pipeline were interfaced using Python.

4.7.2 Quality of Separation

In this section we present the quality of separation between `pro` and `anti` polarities and compare the results with alternate methods.

Trump

Collected around the U.S. Presidential election 2016, with the single keyword `Donald Trump`. The controversy was regarding the republican candidate, and therefore this dataset has `Pro-Trump` and `Anti-Trump` tweets. Figure 4.9 shows the ROC obtained from various mechanisms. The dataset has a strong separation between `pro` and `anti` groups with the `pro` group talking about their likeness for the candidate, and the `anti` group talking about the controversies or mocking him. 30 largest assertions were chosen. Our formulation, labeled 'EM Polar' had area under ROC 0.96 Note that although there is strong separation, Metis-2 or Metis-3 had area under ROC 0.8 and 0.88, respectively. The reason is that the largest polarized assertions were shared by

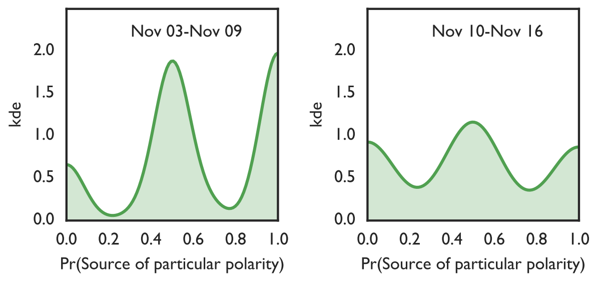


Figure 4.10: Weekly analysis: before and after election

some people alike, creating a neutral group, which confused the partitioning. On the other hand, EM Polar could assign the edges more appropriately in a probabilistic fashion. Because of the presence of campaign related strongly positive and negative words, sentiment analysis has performed reasonably well. The fact-finder has chosen likely truths from both side, which explains the shape of the curve. Figure 4.10 shows an interesting trend resulting from ongoing analysis of the situation, one week before and after U.S. Presidential Election on 2016. It runs our algorithm separately for both week, and plots kernel density estimate for the distribution of sources being in a particular camp, as derived from the converged parameters. Note that one week after the election, the distribution is more uniform, i.e. polarized voice has reduced.

Eurovision

Polarization in this dataset is around the unexpected win of Susana Jamaladinova from Ukraine in Eurovision 2016. The controversy in this dataset is rooted in Ukraine-Russia relations. `Pro-Jamala` corresponds to tweets from her well-wishers spreading the news or congratulating her. `Anti-Jamala` faction asserts the winning song was performed earlier in May 2015 at a separate event, which might have been a reason for disqualification. Some tweets were also related to whether the winning song '1944' was really telling a personal story, or pointing to a political issue related to the deportation of crimean tatars from soviet. Top 100 largest assertions were chosen. Figure 4.11 shows the comparison. It was a `Pro-Jamala` heavy dataset, therefore both our technique 'EM Polar' and Metis-3 performed reasonably. Presence of the positive words related to the win contributed to the performance of sentiment anal-

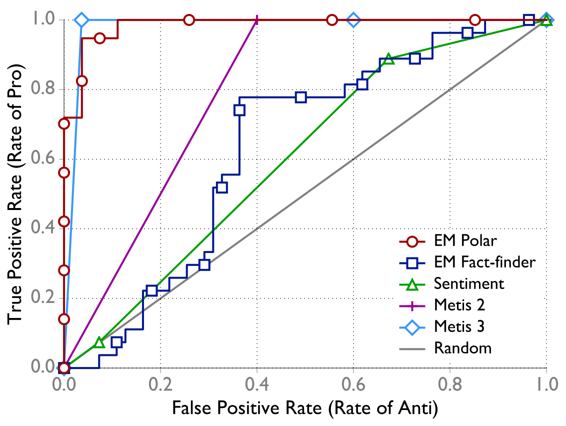


Figure 4.11: Receiver operating characteristics (Eurovision)

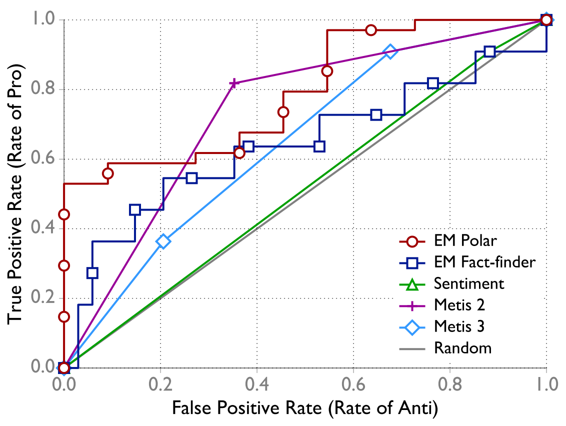


Figure 4.12: Receiver operating characteristics (Egypt)

ysis. One important point to note is that Metis-2 did not perform as well as Metis-3. The reason is the presence of two factions in the anti group, which confused the algorithm.

Egypt

This dataset was collected in July 2013, when the Egyptian army asked the previously elected President Morsi to resign, and appointed a new leader. This led to chaos and confrontations culminating in largely publicized clashes on August 14, where lots of Morsi supporters died. This dataset contains `Pro-Morsi` and `Anti-Morsi` factions as well as large neutral group consisting both neutral sources and assertions. Figure 4.12 presents the ROC. Presence of the neutral has confused the graph partitioning algorithms. Note that

Table 4.3: Occupy Sandy converged parameters

	Neu. Assn.	Pro Assn.	Anti Assn.
Neu. Source	0.79	0.10	0.11
Pro Source	0.69	0.31	0.00
Anti Source	0.71	0.0	0.29

both Metis-2 and Metis-3 have suffered. Therefore, neither a two-way nor a three-way partition was enough to separate the polarized tweets. Sentiment analysis performed poorly, because the sentiment being positive and negative was orthogonal to being `Pro-Morsi` and `Anti-Morsi`.

Occupy Sandy

Occupy Sandy is a disaster relief effort started afterwards Hurricane Sandy in New York area. Tweets related to the restoration activities. The largest 300 assertions were used for polarization analysis. It is not a controversial dataset, and therefore there is no discernible polarization in the dataset which was confirmed by manually reading the tweets. More than 85% sources were classified as neutral by our algorithm. Table 4.3 presents the a^{kl} parameters. Note the large value of first parameters in the second and third columns, which indicates that even though the algorithm could fit a polarization model, most of the fitted polarized sources are connected more strongly to the neutralized network.

Syria

This dataset is collected in the aftermath of Syrian chemican weapons crisis in August 2013. Different camps had different opinions on what happened, blaming different parties for the deaths. These included a hypothesis that Syrian rebels accidentally detonated chemical weapons while transferring them to another location, a hypothesis that the Syrian government ordered those bombs, and a hypothesis that a third (foreign) party carried out the attack to frame Syria. This was a sparsely polarized dataset with multiple polarities, and the results are presented in figure 4.13. Our algorithm, based on different initial conditions of the a^{kl} parameters, came to different

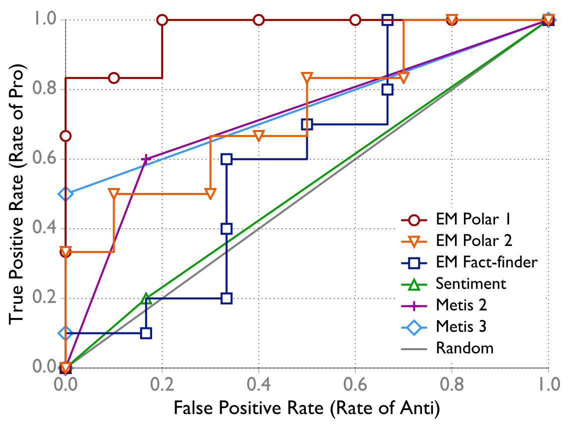


Figure 4.13: Receiver operating characteristics (Syria)

conclusions, which are represented by 'EM Polar 1' and 'EM Polar 2'. One of these are almost perfect with area under ROC around 0.97. Community detection algorithms performed reasonably. And sentiment analysis could not perform well because of foreign language and mostly negative emotions corresponding to both camps.

Results presented in this section confirm that our polarization model based on community bias parameters and individualized speak rates are able to capture polarization present in various controversial situations reasonably well. In most of the cases, our algorithm performed better than graph partitioning or other heuristic based approaches. In the presence of noise or missing information in the source-assertion network, a maximum likelihood model can 'guess' the missing values, and converge to the best solution consistent with the available data.

4.8 Related Works

Polarization in social media has been studied in the context of politics and elections around the globe. Adamic and Glance [165] analyze political blogs related to 2004 US Election. They find difference in the pattern of linking between the liberal and the conservative blogs. Tumasjan *et al.* [166] study the use of Twitter for political deliberation in the context of 2009 election in Germany. They found that the Twitter messages matched with the political campaign and could be used as election predictor. Likewise, Gruzd and

Roy [167] study political polarization on Twitter in the context of 2011 Canadian Federal Election. Polarization also depend on the chosen mechanism of representing relations. According to Conover *et al.* [152], *retweet*-based graphs have strong homophilic separation, but the *mention*-based graphs do not exhibit much. Their study is based on U.S. congressional elections. Garcia *et al.* [168] study polarization in the context of politnetz.ch, a Swiss online political platform.

Lee *et al.* [169] study relation between network heterogeneity and opinion polarization in social media. Barberá [170] argue that access to diversified people through social media can reduce individual exposure to political polarization. Macy *et al.* [171] explore polarizing behavior of dynamic networks. Using dynamics of influence and attraction between computational agents they extend the Hopfield model of attraction. They propose that homophily, xenophobia, etc states of can naturally result in polarization of antagonistic groups.

Many works address polarization as a community detection problem [137, 138]. In general applying such techniques on data directly collected in the real-world social media require case specific tuning. We argue that the preferred technique should be modeling polarization relating to the presence of the non-polarized behavior. Bakshy *et al.* [155] shows polarization and diversity among the facebook users. Barberá [148] explore different case studies showing varying degree of homophily and polarization in Twitter networks. Amin *et al.* [7] propose matrix factorization and ensemble based approach to separate different polarity groups.

Different polarization metrics have also been proposed in various papers. An important finding has been presented by Guerra *et al.* [172]. According to them, *modularity* is not a good measure, as the presence of different communities in the social network does not necessarily correspond to opposing polarized groups. They also find that boundary nodes are likely to be less popular in polarized networks. Morales *et al.* [149] propose methods to measure degree of polarization based on the distribution of opinions. Akoglu [173] uses bipartite opinion networks. Garimella *et al.* [150] attempt to quantify polarization and convtroversy in Twitter using graph partitioning and random walk techniques. Their work focuses on detecting whether there is controvery around a particular topic in Twitter using different statistical measurements. In this chapter, we observe that the dynamics of neutralized sources and content [148, 153, 155, 160] often limits the efficacy of such techniques, and makes it harder to identify the polarized content from the nonpolarized. Therefore, it is necessary to model polarization by exploiting the propagation structure between different polarity types.

It is important to note how presence of polarization imposes additional complexity on the end-user systems. Due to widespread use of social media [144, 145] and availability of mobile devices, various forms of citizen journalism is easily possible. Systems crafted for preparing diversified newsfeed [9, 174, 175] are prone to choosing a side and mislead users if polarization is not taken into account for controversial issues. Jisun *et al.* [154] explore bipartisanship and polarization in the context of Facebook posts. Amin *et al.* [4], Kase *et al.* [5] study social-sensing [3] in polarized networks. Their paper shows polarization is strong enough to alter the results of various analytics algorithms.

Our model is dependent on the presence of source information related to the content. The source-assertion network or some variant of it has been used by veracity analysis, or information ranking algorithms commonly known as fact-finders. The model has been incorporated into several social-sensing architectures [2, 3]. If the social network is not readily available, prior literature addresses how to infer it from the data [132, 133]. Netrapalli and Sanghavi [130], Myers and Leskovec [135], and Rodriguez *et al.* [136] use data driven approaches to estimate a social network using different maximum likelihood formulations. Using source information to perform classification in social networks has been studied by Wang *et al.* [124]. While the maximum-likelihood formulation in those papers focus on reliability or veracity of the sensed observations, our formulation aims to analyze bias and polarization.

Sentiment analysis [157, 176] has been used in earlier works to address polarization. Choi *et al.* [158], Mejova *et al.* [159] study sentiment analysis based techniques to identify controversial issues. It has been pointed out that sentiment analysis is not a preferred technique to model polarization. For example, a tweet that is `pro-government` can have both positive or negative sentiment, and vice-versa. Moreover, because of extensive training required for sentiment analysis tools, they are less applicable to dynamic and internationalized environments like Twitter, Facebook, or Instagram, where people often introduce new terms, hashtags, colloquial language, satire, contextual terms. Therefore, while sentiment analysis or other natural language processing tools can be used as a next-step to refine the output, the first approach to modeling polarization in social media benefits from using structural properties of the network. Sentiment analysis will help in the absence of source information or loosely connected sources.

4.9 Summary

This chapter is an exercise on how to properly model polarization in social sensing. We have started with a simple probabilistic model and improved it with subsequent formulations. Our solution is based on the maximum likelihood estimation of fitting a model consisting pro, anti, and neutral voice and the interaction between these types. We show the strength of our model with real-world polarized datasets collected from Twitter. Our algorithm has been able to correctly follow the level of polarization in ongoing events of international interest. Our research opens avenues for further investigation. Is it possible to fuse information from multiple data sources across different networks? Are the models obtained from different situations comparable in terms of the parameters? Is it possible to derive structural properties of the social network to measure strength or type of on-going contention, and predict future transformations? Are the polarization models affected by the presence of bots and promotional content in the social media? Future research will focus on such questions.

CHAPTER 5

UNVEILING POLARIZATION IN SOCIAL NETWORKS

This chapter presents *unsupervised* algorithms to uncover polarization in social networks (namely, Twitter) and identify polarized groups. The approach is language-agnostic and thus broadly applicable to global and multilingual media. In cases of conflict, dispute, or situations involving multiple parties with contrasting interests, opinions get divided into different camps. Previous manual inspection of tweets has shown that such situations produce distinguishable signatures on Twitter, as people take sides leading to clusters that preferentially propagate information confirming their individual cluster-specific bias. We propose a model for polarized social networks, and show that approaches based on factorizing the matrix of sources and their claims can automate the discovery of polarized clusters with no need for prior training or natural language processing. In turn, identifying such clusters offers insights into prevalent social conflicts and helps automate the generation of less biased descriptions of ongoing events. We evaluate our factorization algorithms and their results on multiple Twitter datasets involving polarization of opinions, demonstrating the efficacy of our approach. Experiments show that our method is almost always correct in identifying the polarized information from real-world Twitter traces, and outperforms the baseline mechanisms by a large margin.

5.1 Overview

This chapter presents algorithms to uncover polarization on social media networks, such as Twitter, and identify opposing sets of biased tweets. We define polarization as a condition in which two opposing views enjoy wide support by different groups in a community. For example, a community might become divided over a political or social issue; this is often manifested as opposing views on how the issue should be resolved. Often, the conflict extends to claims about factual observations, such as whether a person had a gun on them or not at a particular time. These widely held and reported conflicting beliefs obfuscate descriptions of the real progression of events as a result of various injected biases. To uncover a less biased (i.e., more neutral) description of events, it is important to identify polarization and distill neutral observations from the reported mix, which motivates the work in this chapter.

In this chapter, we present a polarization model for information networks, and show that the presence of polarized groups can be detected by considering dependence among posted observations. Using matrix rank as a parameter, we propose a matrix factorization approach to uncover polarization. We explore different degrees of polarization and compare the quality of separation (of tweets of opposing polarity) across different algorithms using real traces collected from Twitter. The work is motivated, in part, by the increased reliance on social networks as news sources. Social network based news dissemination is different from traditional news networks, where raw information goes through curation by expert analysts and journalists before publication. In contrast, in the social media, anybody can post anything. Polarization or bias is inevitable [152]. Hence, tools are needed to clean-up the media before consumption as news.

Our results demonstrate that opposing sets of polarized tweets and sources can be identified automatically (with no content analysis or natural language processing) by the aforementioned matrix factorization approach. Experiments show that the proposed algorithm performs much better than the baseline methods in terms of accuracy in unveiling the polarized sources and groups. The underlying intuition lies in that, in cases of conflict, parties of opposing views tend to disseminate dissimilar sets of claims on social networks. Hence, some of the disseminated tweets can be separated into two subsets propagated by largely non-overlapping sets of sources. We represent the set of tweets as a matrix, where one dimension represents sources and the other represents their tweets (or claims), and where non-zero entries represent who said/forwarded which tweet. Given such a matrix, our algorithm uncovers the underlying latent groups and claims, thereby identifying both the conflicted social groups and their respective views. The language-independent nature of the approach makes it especially advantageous in applications involving multilingual media such as Twitter, since no dependency on a particular lexicon is involved.

It should be noted that the problem addressed in this work is different from detecting communities in the social network. We observe that in practice, the crawled traces of polarized tweets are often intermixed with a large set of neutral sources and observations. Since neutral observations can also be relayed by polarized sources, and since neutral sources may mix and match view of different polarities, the task of separating the polarized clusters becomes a much harder problem. In this setting, algorithms based on community detection do not correctly identify the polarized clusters. In this chapter, solutions are presented that explicily handle the existence of a neutral poplulation of sources and claims that blurs the boundaries of polarized groups.

The problem addressed in this work is also different from the commonly addressed problem of veracity analysis on social media. There can be several types of bias present on the social medium. In the extreme case, one or more of the polarized sources are malicious. People post false information to glorify or defame certain acts or causes. This propaganda may result in an 'online war' on the social platform, and veracity analysis might be used to detect improbable claims. However, it is often that polarization is more benign. Individuals do not post entirely fabricated observations, but rather color true observations depending on their opinions. A more subtle form of polarization occurs when people selectively propagate or suppress observations based on their bias. For example, a person supporting political party X may only forward (true) positive information about X and the negatives about Y. Another person can forward (true) information about the opposite. In this case, veracity analysis does not help identify polarization.

Also, note that the problem is different from sentiment analysis. A statement that mentions, say, a president and features a negative sentiment might not actually be opposing the president. It might be negative on something else. For example, consider these tweets regarding a former Egyptian president (Morsi): "Saudi Arabia accused of giving Egypt $1B to oust Morsi" or "Egypt clashes after army fire kills #Morsi supporters". Both tweets mention "Morsi" (the president) and feature negative sentiments (due to use of such keyword as "accuse", "oust", "clash" and "kill"). However, reading them carefully, it is easy to see that both sympathize with the president depicting him and his supporters as victims.

The contribution is therefore novel in addressing the problem of identifying and separating *polarization* as opposed to, for example, performing veracity analysis, community detection, or sentiment analysis.

The rest of this chapter is organized as follows. Section 5.2 illustrates a motivating example from Twitter that leads to our problem, models polarization in social network, and formulates the problem. Section 5.3 derives a matrix factorization based gradient descent algorithm to estimate the polarities. In section 5.4 we describe the implementation, evaluate our algorithm, and compare it to other baselines. Section 5.5 reviews the literature related to polarization in social networks. The chapter concludes with a discussion in section 5.6.

5.2 Polarization in Social Networks

The end result (of identifying polarization), presented in this chapter, could in principle be accomplished using semantic analysis and natural language processing. The goal of our work, however, is to achieve that end in a *language agnostic* manner. There are two reasons why this is important. First, on a multi-lingual, multi-national medium, such as Twitter, the number of languages used is large. Developing a model for each language specifically to identify polarization is a rather expensive undertaking. Second, it is not always clear that understanding the language helps understand the polarity of a statement. Consider, for example, the following tweet about Jamala, the winner of the Eurovision competition in 2016: “Jamala performs Bizim Qirim at Kiev concert Hall, 18 May, 2015. The same song wins Eurovision one year later”. Is this tweet advertising Jamala (i.e., is “pro”) or is it against her (i.e., is “anti”)? Someone not familiar with the underlying background might consider it pro. In reality, it is not. Eurovision rules dictate that Eurovision songs have to be original. By claiming that the song was performed a year earlier, the source suggests that the entry should have been disqualified. The need to understand situation-specific context on a case-by-case basis poses significant challenges when it comes to building general-purpose schemes for identifying polarity.

Our approach uses a different intuition. Individuals retweeting statements such as the above, on average, understand their context and polarity. Their behavior reflects their understanding. Hence, by monitoring such collective behavior (namely, the overall propagation patterns of tweets), and clustering it by similarity, it is possible to separate "pro" versus "anti" without having to understand the language. In essence, we harness the collective intelligence of the social medium. In the following sections, we introduce the information model for polarized social networks, and formally define the problem.

5.2.1 Information Model for Social Networks

Online social platforms often allow mechanisms to crawl public information. The crawled information at first goes through domain specific cleaning or filtering steps. The content is then clustered using appropriate similarity measurement, which helps to consolidate small variations in the data, and generate a rich information network. A cluster of the very similar observations is considered as a single *assertion*, and the people or the authors who posted those observations are considered as *sources*. The bipartite graph from the *sources* to the *assertions* is called a *source-assertion network*. The method of generating this network from the crawled data has been discussed in detail in different works [3, 8]. In this chapter, we represent the source-assertion network as a binary *source-assertion matrix* A of dimensions $s \times c$, where s is the number of sources, and c is the number of assertions. If source i claims assertion j, then $a_{ij} = 1$, otherwise $a_{ij} = 0$.

In addition to the source-assertion network, a social influence or dependency network can also be derived (or crawled), where an (s, t) edge denotes that source s has a tendency to forward information if it is received from source t. This graph can be weighted when the intensity of influence or dependency is considered into the model, or it can be simplified as an unweighted graph of binary relations. We represent it as a $s \times s$ *social dependency matrix* $T = [t_{ij}]$ of binary values. It can be derived from an explicit social network such as Twitter follower-followee relations. It can also be estimated from retweet behavior of the sources. Netrapalli and Sanghavi [130] model the propagation of information through the social network as cascades of epidemics. Given the tweets along with sources and timestamp information, they solve the inverse problem of finding the latent propagation structure. In this chapter, we estimate the social dependency network using their maximum likelihood estimation mechanism.

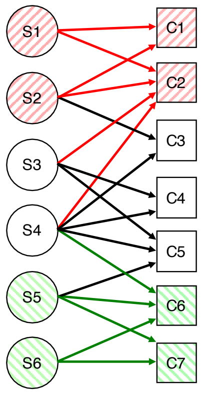
5.2.2 Modeling Polarized Information Networks

In this section we augment the source-assertion network with additional states for polarized scenario. Please note that these models are developed according to the real-world observations reported by multiple independent works [4, 155].

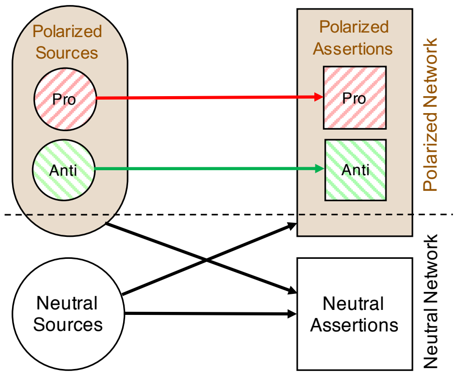
We define a *polarity group* as the set of different senses (*polarities*) relative to a polarity context (*pole*). For example, in a US political parties context, the polarity group can be $K = \{\texttt{democrat}, \texttt{republic}\}$. Please note that although two polarities are common, the polarity group can contain more than two members, if the context is not bipartite. For example, the polarity group of the former example could also contain `libertarian` as a polarity. Given a set of assertions strictly related to the polarity context, each assertion can be classified as one of the polarities from the polarity group. However, due to the nature of data collection, often there are assertions that do not belong to any of the polarities, which can be termed as `neutral`, or *nonpolarized assertions*. For example, every tweet that contains the keyword `Morsi` is not necessary pro-Morsi or anti-Morsi.

Obtaining the ground truth about the polarity of an assertion requires human effort (that we want to automate). It requires a human grader to understand the content of the assertion and its context. Then the grader assigns a polarity from the polarity group, or classifies it as a neutral or nonpolarized assertion. A source is polarized if its odds of making non-neutral claims of a particular polarity is above a threshold τ, otherwise a source is neutral.

Suppose the polarity group is $K = \{\texttt{pro}, \texttt{anti}\}$. The bipartite source-assertion network takes the form shown in Figure 5.1a. Circles $S1$ to $S6$ represent six sources, and squares $C1$ to $C7$ represent seven assertions. Empty circles (squares) represent neutral sources (assertions). Filled circles (squares) represent polarized sources (assertions). Arrows represent claims of different polarities. The relationship between the polarized and the neutral components can be represented as Figure 5.1b. Here sources (assertions) with particular polarities are consolidated together as a single circle (square) rep-



(a)



(b)

Figure 5.1: (a) Model of a polarized source-assertion network, (b) Relation between the polarized and the neutral network.

resenting that polarity. The rest of the vertices are consolidated as `neutral` sources and `neutral` assertions. The polarized vertices are consolidated as the polarized network.

5.2.3 Problem Formulation

Consider a scenario with two polarities, namely `pro` and `anti`. Some of the observations posted in the social media favors or averts the `pro` camp. Some of the observations do the same for the `anti` camp. In this chapter, our goal is to separate the polarities. To develop the formulation, we consider the simplified case with opposing polarities only, without the presence of the neutral network. Later we show how the solution to the simplified formulation is adapted to solve the general case with a huge neutral network obscuring the polarized network.

The observation that there are polarized factions that do not share posts contradicting their polarities, allows us to separate them. Consider $\Pr(S_iC_j)$, probability that source i shares an observation j. $\Pr(C_j^q)$ denotes the probability that assertion j is of polarity q. $\Pr(S_i^q)$ denotes the probability that source i is of polarity q. We can then write equation 5.1.

$$\begin{aligned}
&\Pr(S_iC_j)\\
&= \Pr(S_iC_j|C_j^{pro}).\Pr(C_j^{pro}) + \Pr(S_iC_j|C_j^{anti}).\Pr(C_j^{anti})\\
&= \Pr(S_iC_j|S_i^{pro}C_j^{pro}).\Pr(S_i^{pro}).\Pr(C_j^{pro})
\end{aligned}$$

$$
\begin{aligned}
&+ \Pr(S_iC_j|S_i^{anti}C_j^{pro}). \Pr(S_i^{anti}). \Pr(C_j^{pro}) \\
&+ \Pr(S_iC_j|S_i^{pro}C_j^{anti}). \Pr(S_i^{pro}). \Pr(C_j^{anti}) \\
&+ \Pr(S_iC_j|S_i^{anti}C_j^{anti}). \Pr(S_i^{anti}). \Pr(C_j^{anti}) \qquad (5.1)
\end{aligned}
$$

When the assertion opposes the polarity of a source, the source is not going to share it. Therefore, both $\Pr(S_iC_j|S_i^{pro}C_j^{anti})$ and $\Pr(S_iC_j|S_i^{anti}C_j^{pro})$ reduces to 0.

$$
\begin{aligned}
\Pr(S_iC_j) = &\Pr(S_iC_j|S_i^{pro}C_j^{pro}). \Pr(S_i^{pro}). \Pr(C_j^{pro}) \\
&+ \Pr(S_iC_j|S_i^{anti}C_j^{anti}). \Pr(S_i^{anti}). \Pr(C_j^{anti}) \qquad (5.2)
\end{aligned}
$$

Now we consider the terms $\Pr(S_iC_j|S_i^{pro}C_j^{pro})$ and $\Pr(S_iC_j|S_i^{anti}C_j^{anti})$ in equation 5.2. In an ideal situation, these values are 1, making a source share each and every observation whenever the polarity matches. In practice, this does not happen. $\Pr(S_iC_j|S_i^qC_j^q)$ depends on various social and human factors, but we can simplify this probability as a combination of two *independent* components, (i) activity level of the source i denoted by $act(S_i)$, and (ii) circulation level of the assertion j denoted by $cir(C_j)$. Taking δ as a scaling constant, we can write $\Pr(S_iC_j|S_i^qC_j^q) = \delta.act(S_i).cir(C_j)$.

$$
\begin{aligned}
\Pr(S_iC_j) = &\delta.act(S_i).cir(C_j). \Pr(S_i^{pro}). \Pr(C_j^{pro}) \\
&+ \delta.act(S_i).cir(C_j). \Pr(S_i^{anti}). \Pr(C_j^{anti}) \qquad (5.3)
\end{aligned}
$$

Consider the general case with k polarities, $q \in \{1..k\}$. $U = [u_{iq}]$ is an $s \times k$ matrix, and $V = [v_{jq}]$ is a $c \times k$ matrix. Activity levels of the sources and their probabilities to belong to particular polarized camps are represented in U. Circulation levels of the assertions and their probabilities to favor particular camps are represented in V. Therefore, $u_{iq} = \delta_1.act(S_i). \Pr(S_i^q)$, and $v_{jq} = \delta_2.cir(C_j). \Pr(C_j^q)$. If $\hat{A} = [\hat{a}_{ij}]$ represents the probability of a source to share a particular assertion, we can rewrite equation 5.3 as $\hat{a}_{ij} = \sum_{q=1}^{k} u_{iq}v_{jq}$, or $\hat{A} = UV^T$.

Given $A = [a_{ij}]$ as the actual observations on whether source i shared assertion j in the social network, $T = [t_{ij}]$ as the social dependency matrix on whether source i is likely to forward information received from source j, and a polarity group K, our goal is to estimate U and V component matrices that allow us to separate the polarized components.

5.2.4 Solution Approach

Given k as the rank, we can factorize A to estimate U and V components. Please note that, $A = UV^T = URR^{-1}V^T = (UR)(VR^{-T})^T$, where R is a $k \times k$ multiplier matrix. Therefore, factorizing A without any constraint will result in UR and VR^{-T} as component matrices. In the following section, we add appropriate constraints to limit the arbitrariness of R. In the simplified case, when there is only the polarized network, rank of A is exact. Sources and assertions of different polarities can be uniquely separated using the estimated factor matrices $\hat{U}$ and $\hat{V}$. However, in the presence of a large neutral network, the number of polarized camps k does not correctly represent the rank of A. In this case, different separations are possible that can approximate the observation matrix A. We estimate multiple instances of $(\hat{U}, \hat{V})$ using different initializations. For each instance, observations are partitioned into different polarities. Instances are generally related to each other in terms of similarity between corresponding partitions. Anomalous instances that are highly different than the rest are discarded. Rest of the instances are aggregated to estimate the final partitions.

5.3 A Matrix Factorization Approach to Uncover Polarization

In this section, we derive a gradient-descent algorithm to jointly estimate the polarization of the sources and assertions. Suppose A is the $s \times c$ source-assertion matrix. Polarization of the sources and the assertions can be estimated from A by factorizing it in the form of matrices $\hat{U}$ and $\hat{V}$, defined eariler.

If $k = rank(A)$, A can be factorized exactly in the form $A = UV^T$, where $U = [u_{ij}]$ is an $s \times k$ matrix that represent the polarization of the sources, and $V = [v_{ij}]$ is a $c \times k$ matrix representing the polarization of the assertions. Please note that A is an incomplete matrix because when source i does not claim assertion j, it can be that source i did not have opportunity to observe j, or i ignored assertion j after observing it. Therefore a sample of the missing edges in the source-assertion network are represented as $a_{ij} = 0$, and the rest are considered as missing. Because A is incomplete, we do not know the exact rank of A. However, as visible from Figure 5.1a, the sources and assertions of different polarized camps are independent when they are sharing information related to the particular polarized scenario. Hence, we take the number of polarized groups $|K|$ as $rank(A)$, and *approximately* factorize A as $A \approx UV^T$.

Note that this condition is defined only for the entries of A that are observed. Therefore, let us define the set $O = \{(i, j) : a_{ij} \text{ is observed}\}$ to be all the indices in matrix A that are observed. Given a particular U and V, the estimate of an entry of A is given by $\hat{a}_{ij} = \sum_{q=1}^{k} u_{iq}v_{jq}$. Therefore, the estimation error is $e_{ij} = a_{ij} - \hat{a}_{ij}$. In order to approximately factorize the matrix A, we need to minimize the objective function, which is equal to the sum-of-squared errors $J = \sum_{(i,j)\in O} e_{ij}^2 = \sum_{(i,j)\in O}(a_{ij} - \sum_{q=1}^{k} u_{iq}v_{jq})^2$. This form of the objective function, however, can result in infinitely many solutions, each of which minimizes J. Therefore, we impose the following constraints. The first constraint corresponds to overfitting of the objective function. The second constraint corresponds to the impact of the social dependency matrix.

Regularization

If U and V is a particular solution, then multiplying U by an arbitrary $k \times k$ real matrix R, and multiplying V by R^{-T} would also minimize J, provided R is inversible. This is because $UV^T = UIV^T = URR^{-1}V^T$, where I is a $k \times k$ identity matrix. Depending on the chosen initial values or the missing entries, the objective function can overfit the model, or oscillate between multiple solutions. Therefore, we impose a regularization constraint on J. We choose to use L_2-regularization $\lambda(||U||_F^2 + ||V||_F^2)$ so that arbitrarily large values in R would be prevented. The multiplier $\lambda > 0$ represents the value of the regularization parameter.

Social dependency-based polarization consistency

We observed that polarization in the crawled data is obscured by a large non-polarized or neutral network. Presence of such sources and assertions result in multiple separations between the different polarity groups likely. The objective function would result in multiple candidate solutions. Therefore, we add an additional constraint. Users that depend on one another according to the social dependency matrix T are more likely to exhibit polarization consistency. So the columns in U corresponding to sources who depend on each other contains similar entries. If $\overline{u_x}$ is the row in U corresponding to source x, the additive component $\gamma t_{ij}||\overline{u_i} - \overline{u_j}||^2$ would add a penalty whenever source i depends on source j, but their corresponding columns vary. Here $\gamma > 0$ is a parameter that regulates the importance of the social consistency component. This parameter can be chosen later in the tuning phase. Please note that adding this constraint will increase the error in the factorization but it will favor solutions that have higher consistency with the social dependency network.

Therefore, by adding these terms, our objective function becomes $J = \sum_{(i,j)\in O}(a_{ij} - \sum_{q=1}^{k} u_{iq}v_{jq})^2 + \sum_{i,j}\gamma t_{ij}||\overline{u_i} - \overline{u_j}||^2 + \lambda(||U||_F^2 + ||V||_F^2)$, which needs to be minimized.

5.3.1 Solving the Optimization Problem

We minimize J with respect to the parameters in U and V using gradient descent method. We rewrite the objective function, and compute the partial derivative of J with respect to each parameter in U and V:

$$J = \sum_{(i,j)\in O} e_{ij}^2 + \sum_{i,m}\gamma t_{im}\sum_{q=1}^{k}(u_{iq} - u_{mq})^2 + \lambda(||U||_F^2 + ||V||_F^2) \tag{5.4}$$

$$\frac{\partial J}{\partial u_{iq}} = 2\sum_{j:(i,j)\in O} e_{ij}(-v_{jq}) + 2\sum_{m}(\gamma t_{im} + \gamma t_{mi})(u_{iq} - u_{mq}) + 2\lambda u_{iq} \tag{5.5}$$

$$\frac{\partial J}{\partial v_{jq}} = 2\sum_{i:(i,j)\in O} e_{ij}(-u_{iq}) + 2\lambda v_{jq} \tag{5.6}$$

Note that we can ignore the constant factor of 2 throughout the RHS of the aforementioned equation for the purposes of gradient descent. We compute all partial derivatives with respect to the different parameters in u_{iq} and v_{jq} to create gradient matrix ∇U of dimensions $s \times k$, and ∇V of dimensions $c \times k$. The gradient-descent method updates $U \Leftarrow U - \alpha\nabla U$, and $V \Leftarrow V - \alpha\nabla V$, where α is the step-size. The parameter γ, λ can be selected using cross-validation. Figure 5.2 enumerates this mechanism.

We impose the additional constraint that the entries of the matrix U and V are non-negative, although the optimization objective function remains

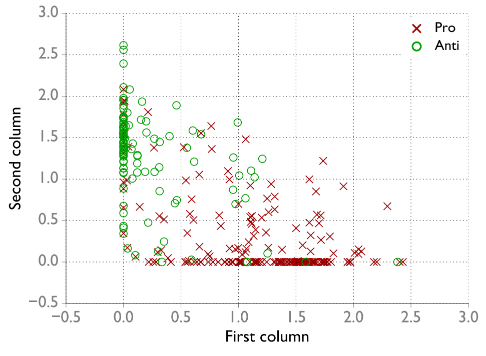
```
1: procedure FACTORIZE($A, T, k$)
2:     Randomly initialize $U, V$
3:     repeat
4:         for each $(i, q)$ do
5:             $u_{iq}^{+} \leftarrow u_{iq} - \alpha \frac{\partial J}{\partial u_{iq}}$        ▷ Equation 5.5
6:         end for
7:         for each $(j, q)$ do
8:             $v_{jq}^{+} \leftarrow v_{jq} - \alpha \frac{\partial J}{\partial v_{jq}}$        ▷ Equation 5.6
9:         end for
10:        for each $(i, q)$ do
11:            $u_{iq} \leftarrow u_{iq}^{+}$
12:        end for
13:        for each $(j, q)$ do
14:            $v_{jq} \leftarrow v_{jq}^{+}$
15:        end for
16:    until convergence reached on $U, V$
17:    return $(U, V)$
18: end procedure
```

Figure 5.2: Gradient descent algorithm for factorization

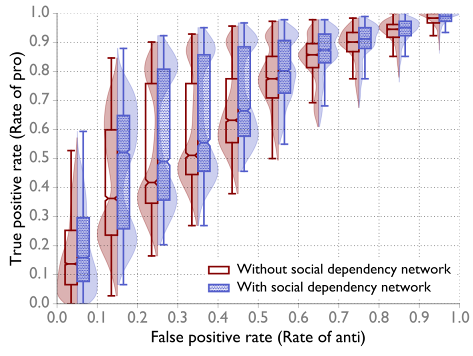
the same. It provides a sum-of-parts decomposition to the source-assertion matrix as dictated by the problem formulation. To achieve this, during initialization, the entries of matrices U and V are set to non-negative values in $(0, 1)$. During an update, if any entry in U or V becomes negative, then it is set to 0.

5.3.2 Separating Polarities using $\hat{U}$ and $\hat{V}$

Activity levels of the sources and their probabilities to belong to particular polarized camps are represented in U. Circulation levels of the assertions and their probabilities to favor particular camps are represented in V. Rows of U and V can be considered as points in a k-dimensional euclidean space. In the simplified case, where the source-assertion matrix consists of only the polarized network with $K = \{\texttt{pro}, \texttt{anti}\}$, the extreme points of U or V are $(1, 0)$ or $(0, 1)$. These points represent the sources making all the `pro` assertions, or making all the `anti` assertions, respectively. All the other points would fall on either x-axis or y-axis. However, in the general case, the neutral network is present, hence it is possible to have points that fall within the right triangle defined by vertices at $(0, 0)$, $(1, 0)$, and $(0, 1)$.



(a)



(b)

Figure 5.3: (a) Assertions from the estimated factor matrix $\hat{V}$ and their polarities, (b) Although the social dependency network improves performance, there is still variance in the separation due to the presence of the neutral network.

Through factorization we have estimated $\hat{U} = UR$, and $\hat{V} = VR^{-T}$. This multiplier R causes the estimated values in $\hat{U}$ or $\hat{V}$ to have been applied a linear transformation. A linear transformation in general can be decomposed to several rotations and scales. Due to the constraints we have added to J, effect of R is small. Figure 5.3a shows an output where the rows of $\hat{V}$ are plotted on the $2D$ plane for a particular experiment. We observe that the multiplier R has been mostly restricted to a diagonal matrix corresponding to scale transformation.

Figure 5.3a also plots the ground truth of the assertions as obtained via manual annotaion. To separate the different polarity groups, we note that linear transformations preserve parallel lines. Therefore, the midpoint of a transformed line corresponds to the transformation of the midpoint of the original line. We can separate the polarities by finding the pair of assertions (a, b) from $\hat{V}$ with maximum euclidean distance, i.e. $\arg\max_{a,b} ||\overline{\hat{v}_a} - \overline{\hat{v}_b}||^2$, and assigning the other assertions to either the polarity of a or b, using a nearest neighbor rule. However, we observe that R has been mostly restricted to scaling. Therefore, to obtain a separation of the polarities, assertion j can be assigned to the group corresponding to $\arg\max_q\{\hat{v}_{jq}\}$. For Figure 5.3a, this corresponds to using sign of $\hat{v}_{j,1} - \hat{v}_{j,2}$ as the separator. The sources can also be separated in a similar manner using $\hat{U}$.

5.3.3 Ensemble of Factorization Experiments

We note that in the presence of a large neutral network, different runs of factorization results in different separations. Figure 5.3b illustrates the receiver operating characteristics (ROC) for the egypt scenario. ROC curve plots true positive rate vs. false positive rate, and is used to assess the quality of classification. The optimal algorithm has an area of 1 under the ROC, which happens when the output includes all the true positives before any of the false positives.

Figure 5.3b plots the distribution of true positive rate for different false positive rates, and shows that although the factorization algorithm is able to achieve good performance, there is significant variance in the separation obtained from the results. We also compare the result of when the social dependency network is used as a constraint vs. when it is not. We observe that although use of social dependency network improves the quality of the results, there is still variance in the separation. We, therefore, use an ensemble of factorization experiments to estimate the most likely assignments of the assertions to the respective polarities.

It is not possible to directly compare $\hat{V}_m$ with $\hat{V}_n$, when m and n different experiments, because of the transformation difference caused by R_m and R_n. We, therefore, separate the assertions to different polarity groups for each experiment. Experiments are aligned to each other using a mechanism based on Jaccard distance [177]. We explain it for two polarities, i.e. $k = 2$.

Figure 5.4 and figure 5.5 shows the algorithm, with the procedure EstimatePolarities at line 9 of figure 5.5 being the starting point. Suppose the separation generated by factorization experiment m is B_m^1 and B_m^2, and the separation generated by factorization experiment n is B_n^1 and B_n^2. It is possible that (B_m^1, B_m^2) aligns with (B_n^1, B_n^2), or with (B_n^2, B_n^1). Figure 5.6a illustrates the two cases considering the polarities as `pro` and `anti`. We compute a 2×2 matrix of the Jaccard distances between the separations created by the experiments. Jaccard distance between two sets X, Y is defined as $1 - \frac{|X \cap Y|}{|X \cup Y|}$. It is used to assess how similar or dissimilar they are. In order for the two experiments to match, either the main diagonal will exhibit more similarity than the anti-diagonal (or vice versa). If the maximum in the matching diagonal is below a threshold τ_{edge}, given their difference is within τ_{diag}, the experiments are considered to match and a weighted edge is added

```
procedure PartitionDistance($B_m, B_n$)
    ▷ JaccardDist($X, Y$) $= 1 - \frac{|X \cap Y|}{|X \cup Y|}$
    $d_{11} \leftarrow$ JaccardDist($B_m^1, B_n^1$)
    $d_{12} \leftarrow$ JaccardDist($B_m^1, B_n^2$)
    $d_{21} \leftarrow$ JaccardDist($B_m^2, B_n^1$)
    $d_{22} \leftarrow$ JaccardDist($B_m^2, B_n^2$)
    $dist \leftarrow 1.0$
    if $d_{11} < d_{12}$ and $d_{22} < d_{21}$ and $|d_{11} - d_{22}| < \tau_{diag}$ then
        ▷ $(B_m^1, B_m^2)$ aligns with $(B_n^1, B_n^2)$
        $dist \leftarrow max(d_{11}, d_{22})$
    else if $d_{12} < d_{11}$ and $d_{21} < d_{22}$ and $|d_{12} - d_{21}| < \tau_{diag}$ then
        ▷ $(B_m^1, B_m^2)$ aligns with $(B_n^2, B_n^1)$
        $dist \leftarrow -max(d_{12}, d_{21})$
    end if
    return $dist$
end procedure

procedure GenerateExpGraph($B, size$)
    $G \leftarrow \varnothing$
    for $m \in [1, size - 1]$ do
        for $n \in [m + 1, size]$ do
            $dist \leftarrow$PartitionDistance($B_m, B_n$)
            if $|dist| < \tau_{edge}$ then
                ▷ Insert weighted undirected edge $(m, n, dist)$ to $G$
                $G \leftarrow G \cup (m, n, dist)$
            end if
        end for
    end for
    return $G$
end procedure
```

Figure 5.4: Algorithm to form an ensemble of factorization experiments

to G, the graph of experiments. The weight is considered positive if the experiments matched along the main diagonal, and negative if the experiments matched along the anti-diagonal. Figure 5.6b shows an experiment graph (without the weights) obtained from 20 experiments on the egypt polarized scenario. Experiments that highly differ from the others remain isolated in the experiment graph, or form small islands. We find the experiment with the largest degree in G, and agreegate all the adjacent experiments.

There can be several procedures to aggregate the experiments. We keep a vector of frequencies (x_j, y_j) for each assertion. x_j and y_j counts the number of times assertion j has been assigned to polarity x or y. Normalizing these

```
1: procedure MergeExp($freq, B^X, B^Y$)
2:     for $v \in B^X$ do                                          ▷ Assertions in $B^X$
3:         $freq[v] \leftarrow freq[v] + (1, 0)$
4:     end for
5:     for $v \in B^Y$ do                                          ▷ Assertions in $B^Y$
6:         $freq[v] \leftarrow freq[v] + (0, 1)$
7:     end for
8: end procedure

9: procedure EstimatePolarities($A, T, size$)                  ▷ Run $size$ experiments
10:    ▷ $A$ source-assertion matrix, $T$ source dependency matrix, $k = 2$
11:    for $l \in [1, size]$ do
12:        $(\hat{U}_l, \hat{V}_l) \leftarrow$ Factorize($A, T, 2$)                 ▷ Figure 5.2
13:        $(B_l^1, B_l^2) \leftarrow$ SeparateAssertions($\hat{V}_l$)             ▷ Section 5.3.2
14:    end for
15:    $G \leftarrow$ GenerateExpGraph($B, size$)                    ▷ Graph of experiments
16:    $node \leftarrow$ Vertex with maximum degree in $G$
17:    $freq \leftarrow \varnothing$                      ▷ Mapping assertions to frequency of polarities
18:    MergeExp($freq, B_{node}^1, B_{node}^2$)
19:    for each edge $(node, exp, dist) \in G$ do
20:        if $dist \geq 0$ then
21:            MergeExp($freq, B_{exp}^1, B_{exp}^2$)
22:        else
23:            MergeExp($freq, B_{exp}^2, B_{exp}^1$)
24:        end if
25:    end for
26:    $prob \leftarrow \varnothing$                      ▷ Mapping assertions to distribution of polarities
27:    for each assertion $v$ in $freq$ do
28:        $prob[v] \leftarrow (\frac{freq[v]_1}{degree[node]}, \frac{freq[v]_2}{degree[node]})$
29:    end for
30:    return $prob$
31: end procedure
```

Figure 5.5: Algorithm to estimate polarities from the ensemble

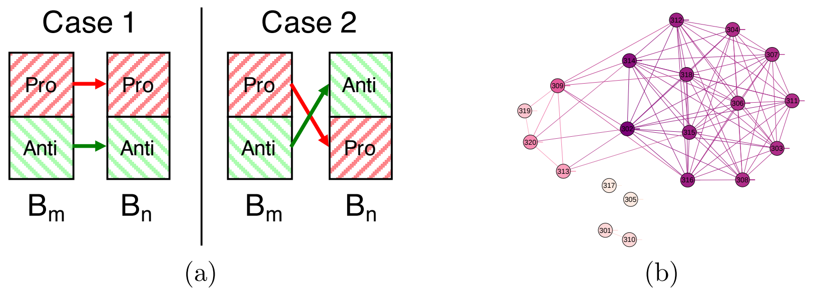


Figure 5.6: (a) Aligning two experiments, (b) Graph of 20 experiments

frequencies and sorting them by the difference of the vector components $(x_j - y_j)$ gives us a spectrum of assertions, from the most likely to belong to one polarized camp to the most likely to belong to the other camp.

5.4 Evaluation

We evaluate our algorithm in the context of polarized scenario in Twitter. Tweets were crawled in real time with tools using Twitter search API. Three sets of traces were collected that contains polarization around (i) former Egyptian president Morsi, (ii) Eurovision song contest 2016 winner Jamala, (iii) US Presidential election candidate Donald Trump. The entire collection of recorded traces was clustered based on text similarity to generate a representative summary [8, 177, 178]. We implemented the factorization program using Java. Sparse matrix data structures were used to efficiently store large matrices. Different components of the pipeline were interfaced using Python. Factorization was performed followed by the ensemble of multiple experiments to separate the tweets between two polarities. We used $k = 2$, $\alpha = 0.001$, $\gamma = 0.1$, $\lambda = 0.5$, ensemble $size = 20$, $\tau_{diag} = 0.15$, $\tau_{edge} = 0.7$. We compare the quality of separation obtained by our algorithm with the following related techniques:

Sentiment Analysis

Sentiment analysis [156, 176] uses language models to understand the sense of content written in natural language, and classifies them as having *positive*, *negative*, or *neutral* sentiment. To annotate the assertions we used Umigon [157] and Sentiment140 [179], two freely available specialized tools to perform sentiment analysis on tweets.

Community Detection

Polarized sources are unlikely to share tweets contradicting their own polarity. Therefore detecting communities in the social network is a candidate mechanism for separating the polarities. We partition the graph of sources and assertions into $k = 2$ communities with the objective of minimizing the *edge-cut* (number of edges that cross partitions). We used Metis [162] to obtain that. In addition to detecting communities, we have added another baseline where the assertions in each community are ranked by their degrees. We refer this mechasim by *MetisVoting*.

Veracity Analysis

Algorithms to perform veracity analysis [2, 3, 44, 124] utilize the source-assertion network to uncover likely facts from the set of tweets. They can be considered related techniques if one of the polarities have more affinity towards factual information. We used the EM-Social [3] algorithm to jointly asses the credibility of the sources and the assertions.

5.4.1 Egypt

Mass street protests against the then president Mohamed Morsi was followed by a coalition led by the army chief, on July 3, 2013. The president was deposed and arrested by the army along with other leaders of his political party. This incident resulted in protests and clashes between the supporters and the opponents of the removed president. Tweets related to the deposed president were collected. For the purpose of evaluation, the largest 1000 clusters containing English tweets were read and manually annotated on whether they were pro-Morsi, anti-Morsi, or neutral in sense. There were 199 pro-Morsi, 109 anti-Morsi, and 692 nonpolarized assertions.

Figure 5.7 compares the receiver operating characteristics (ROC) achieved from our algorithm to other baselines. To obtain the ROC, the set of assertions were sorted in the order of highest polarity in one class to the highest polarity in the other class. When consuming the assertions in that sequence, finding a pro-Morsi assertion was considered as an occurence of true positive, and finding an anti-Morsi assertion was considered as an occurence of false positive. The area under ROC curve measures how well an algorithm performs both in terms of finding the correct answers, and omitting the wrong answers.

Factorization algorithm performs really well. Area under the ROC curve is approximately 0.93. Both Umigon and Sentiment140 performed just as good as a random technique, because (i) a large number of assertions were

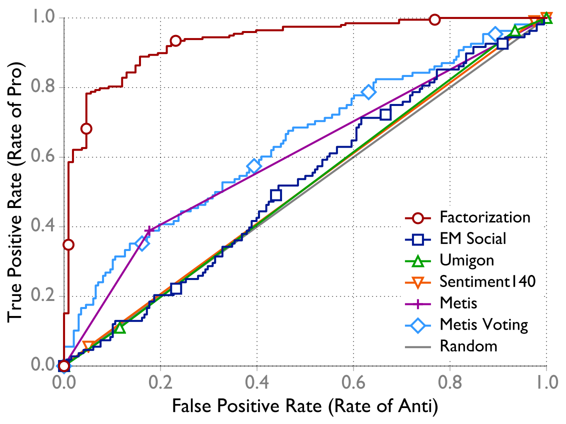


Figure 5.7: Egypt: Factorization performs best with area under ROC 0.93, EM-Social 0.53, Umigon 0.51, Sentiment140 0.51, Metis 0.61, MetisVoting 0.64

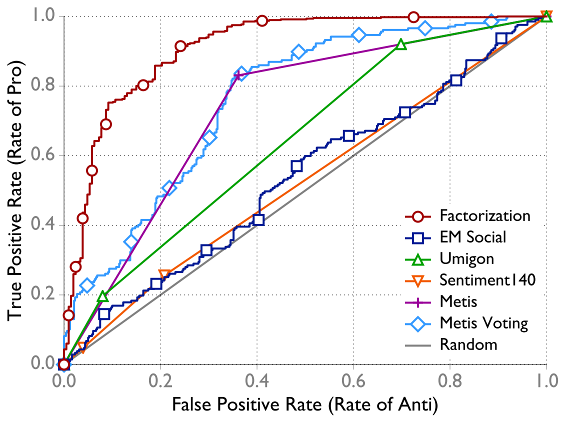


Figure 5.8: Eurovision: Factorization performs best with area under ROC 0.91, EM-Social 0.54, Umigon 0.64, Sentiment140 0.52, Metis 0.73, MetisVoting 0.76

classified as neutral, and (ii) as described earlier in the chapter, sentiment analysis is not the correct technique to uncover polarization. An assertion having positive sentiment can be a positive statement favoring either camp. Hence, sentiment is orthogonal to polarity. EM-Social is also unable to differentiate between the polarities. It illustrates that there was almost no correlation between the veracity of a tweet and any particular polarity. Metis and MetisVoting techniques performed better than the other baselines because of their graph partitioning nature. However, the source-assertion network had around 80% nonpolarized sources and 70% nonpolarized assertions. Therefore a community detection analysis was unable to perform well.

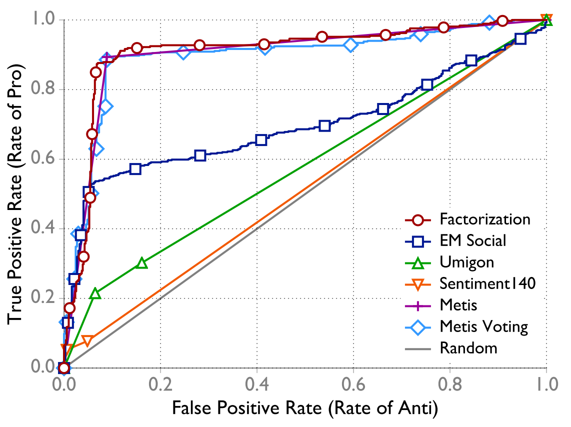


Figure 5.9: Trump: Factorization 0.92, EM-Social 0.70, Umigon 0.58, Sentiment140 0.52, Metis 0.90, MetisVoting 0.90

Table 5.1 shows the top 10 tweets from each polarity from the separation achieved using our algorithm. Note that the tweets on the left column sympathize with the deposed president or his supporters. On the other hand, the tweets on the right column is vocal against the deposed president and his political party, and reporting negative news about them.

5.4.2 Eurovision

Susana Jamaladinova (Jamala) from Ukraine was the winner of Eurovision 2016, an annual European song competition. It was unexpected to many as the expected winner was Russia or Australia according to pre-competition polls. The winning song, 1944, according to the artist, was telling a personal story related to her family in the aftermath of the deportation of the Crimean Tatars by the Soviet Union. However, it was also alleged to have political connotations against Russian interference with Crimea in 2014. Tweets related to *Jamala* were collected for five days after her win. The largest 1000 assertions were manually annotated. There were 600 pro-Jamala, 239 anti-Jamala, and 161 neutral assertions.

Figure 5.8 compares quality of factorization with other baselines. Our algorithm performs best in this scenario. Metis performs reasonably better than the earlier case because of relatively better community separation. Because there were many tweets with positive sentiment that were correlated to pro-Jamala, Umigon also performed better than it did in the other cases.

Table 5.1: Top 10 tweets from the separated polarities (Egypt)

	Pro-Morsi	**Anti-Morsi**
1	Sudden Improvements in Egypt Suggest a Campaign to Undermine Morsi http://t.co/0yCjbKGESr	Prayers for the Christian community in Egypt, facing violent backlash for opposing the Muslim Brotherhood. https://t.co/O5X7BwUjCI
2	Saudi Arabia accused of giving Egypt $1B to oust Morsi http://t.co/d4ZQNntCH	Egypt's Coptic Christians, under attack for supporting overthrow of Muslim Brotherhood, need continued prayers: http://t.co/dW0gdcielb
3	Before Morsi's Ouster, Egypt's top generals met regularly with opposition leaders http://t.co/LbdHKJF508 via @WSJ	Islamic extremists reportedly attacking Egypt's Christian community over Morsi ouster — Fox News http://t.co/VMMN2m49Sw
4	#Egypt: #Morsi supporters denied rights amid reports of arrests and beatings — Amnesty International http://t.co/koVRHlmdWk	In Egypt, the death toll in the clashes between police and pro-Morsi supporters in Cairo has risen to 34.
5	Crowds March in Egypt to Protest Morsi Detention http://t.co/Hp9566xyfB	Amnesty International — Egypt: Evidence points to torture carried out by Morsi supporters http://t.co/8hgAHrNoWd
6	Egypt's Morsi 'To Stand Trial Over Deaths' http://t.co/dDtTFoc0Qt	This... Is... Rab3aaaaaaa! #Morsi http://t.co/f6PJwpQqeE
7	Huge turnout of Morsi supporters here in Mohandiseen #Egypt http://t.co/TzWsV2CP8D	BBC News - Egypt's cabinet orders police to end pro-Morsi sit-ins http://t.co/v2MQV9wgoh
8	Egyptian Leaders Freeze Assets of Morsi Backers, via @nytimes http://t.co/qFndiQbR0u	Egypt's Morsi to be tried for inciting violence http://t.co/rFJ9e5n06w
9	Egypt's ousting of Mohamed Morsi was a coup, says John McCain http://t.co/O8BaBX1vu1	#Egyptians close #Ramsis square, one of the main active points in #Cairo to support #Morsi #AntiCoup #CNN #Egypt http://t.co/rKXSE2y8Ih
10	Good luck today EGYPT!Peacefully Fight for what you know is right! We are thinking of you! http://t.co/ciTqadOjfs	You must be either stupid or stupid if you don't see a direct relation between Morsi's presidency and terrorism in Sinai.

Table 5.2 shows the top 10 tweets from each polarity from the separation achieved using our algorithm. Note that the tweets on the left column are congratulating the winner (pro-Jamala), sharing winning related news, or talking against Russia. On the other hand, the tweets on the right column are against Jamala, and pointing out reasons for the deportation of Crimean

Table 5.2: Top 10 tweets from the separated polarities (Eurovision)

	Pro-Jamala	**Anti-Jamala**
1	Incredible performance by #Jamala, giving Crimean Tatars, suffering persecution & abuse, reason to celebrate https://t.co/XWOZADrywH	For #Jamala1944: Crimea Tatar volunteers in the Nazi army parade before senior German officers, 1942 #Eurovision https://t.co/itgsyKvzO2
2	Breaking: #Russia launches harassment campaign against #Jamala's @Twitter a/c. All known Kremlin trolls. @BBC_ua @AP https://t.co/btk9QyUpkH	Repeat after me: NATO loves Jamala and there was absolutely nothing political about her win (via @marcelsardo) https://t.co/0bTbzVIR35
3	President awarded @jamala title of the Peoples Artist of Ukraine https://t.co/2df8J9zHP5	#BOOM Jamala released Eurovision song commercially on 19.06.2015 in Kiev club Atlas. @EBU_HQ https://t.co/LGbgb77RzH https://t.co/9Se1rwEkIg
4	Jamalas father: We do not talk with Russian journalists https://t.co/EEewVpZ9D2 https://t.co/51rcf8Je3K	#Oops Poroshenko accidently confirms on TV that Jamala's Eurovision song 1944 is the same song "Crimea is ours" from May 2015. @EBU_HQ
5	Congratulations to Ukraine on winning #Eurovision 2016! @JAMALA wrote and composed her song '1944' by herself. https://t.co/vZjYHvtoC	Second left grandfather Jamala!. Ordinary fascist, that "the tyrant Stalin sent him to Kyrgyzstan"! @antonio_bordin https://t.co/b9jsXHhiP1
6	After deportation requiem, #Jamala explains how restrictions in/on #Crimea prevented parents fm joining her @ #ESC https://t.co/tMKlX1qNA9	NATO here also confirms political neutrality of Jamala's Eurovision song: https://t.co/QoZ4Toqnsg @Russianspringru https://t.co/uyt8e4qNAY
7	Russian coverage of Jamalas victory descends to the level of old Soviet anecdote https://t.co/HoNwJKdtCO via @EuromaidanPress	Crimea invites Ukraines Jamala to sing at opening of memorial to deportation victims https://t.co/gIANBIAuFL
8	Photo gallery: #Eurovision winner #Jamala arrives in #Kyiv https://t.co/MBKhuQ7W03 https://t.co/1eEHKAgsjB	Video appears where Poroshenko confirms that the old title of the Jamala's song is "Crimea is ours" @EBU_HQ https://t.co/e7kEFfD6i9
9	#Jamala sends everyone a postcard from home, #Ukraine https://t.co/l1eN6KPVfK	And scene. @Jamala admits that the music to the Eurovision song 1944 was written before September 2015 @EBU_HQ https://t.co/e5lswwcRhn
10	Another thank you from #Jamala #Eurovision #CrimeaIsUkraine https://t.co/W8zOlDRnXn	I must say I feel a little sorry for @jamala, from the start simply a tool in the Wests "#CrimeanTatars" campaign https://t.co/ZduJgP8X7J

Tatars, or arguing that the winning song should be disqualified. Some of the tweets are also sarcastic.

5.4.3 Trump

Donald Trump is the Republican Party nominee for President of the United States in the 2016 election. There have been much debate and controversies around the candidate. Tweets were collected using a single keyword `Donald Trump`, during April 2016. Collected tweets show support by the pro-Trump polarity and the negative opinions or mockery posted by the anti-Trump polarity. For the purpose of generating the ROC curves, the largest 1000 assertions were manually annotated. There were 372 pro-Trump, 522 anti-Trump, and 106 neutral assertions.

Figure 5.9 compares quality of factorization with other baselines. In this particular scenario, performance of our algorithm is around 2% better than community detection. This is because the corresponding source-assertion network had strong community separation, with only 10% nonpolarized assertions being lightly connected. EM-Social also performs reasonably to find the separations because of the same reason. Table 5.3 shows the top 10 tweets from each polarity from the separation achieved using our algorithm. Note that the tweets on the left column are strongly pro-Trump in nature and describing support for him or praising him. On the other hand, tweets on the right column are sharing the negative information about the candidate, and pointing out the controversies.

5.5 Related Work

Presence of polarization in social networks has been studied in various contexts. Conover et al. [152] study *retweet*-based social networks and *mention*-based social networks in political contexts related to U.S. congressional elections. Guerra et al. [172] study polarization metrics for social networks. They argue that *modularity* is not directly applicable as a measure of polarity because even without polarization modular communities are present. Uncovering polarization in social networks is important in various contexts. Bakshy et al. [155] study polarization in the context of Facebook. Amin et

Table 5.3: Top 10 tweets from the separated polarities (Trump)

	Pro-Trump	**Anti-Trump**
1	Retweet if you are 100 PERCENT voting for Donald Trump	Donald Trump said women should be punished for seeking an abortion. That's not a distractionit's a disgrace. https://t.co/sbJ3opebyB
2	@realDonaldTrump Fugedaboudit!!! The woman in New York love Donald Trump!!! https://t.co/7yzgMHVzL4	At this point, Donald Trump has insulted the vast majority of Americans. The good news is, there's something we can all do about it: Vote.
3	Thank you, @NYPost! #Trump2016 https://t.co/KzGweIxaEo	Read and sign this letter that people all over are signing to Donald Trump: https://t.co/S56QbW5K5C
4	"The police are the most mistreated people in this country," Donald Trump #BlueLivesMatter #Trump2016 https://t.co/WfJvWUkMaB	We've earned more votes than any other candidate – Republican or Democrat. https://t.co/tRJNMj86AJ https://t.co/XJIt2bGevs
5	Nobody beats me on National Security. https://t.co/sCrj4Ha1I5	Study: Hillary Clinton, not Donald Trump, gets the most negative media coverage https://t.co/CyONOdFTU0
6	Donald Trump #DonaldTrump #NewYork #NY #NYPrimary #RhodeIsland #Pennsylvania #NewYork4Trump #Delaware #CT #Maryland https://t.co/iYfo13Jjut	Donald Trump says wages are too high. (Yeah, you read that right.) https://t.co/up8ZI1WULC
7	Latinos For Donald Trump 2016 "Go Out & #VoteTrump" #LatinosForTrump #Hispanics4Trump #Trump2016 @realDonaldTrump https://t.co/eSNy170CXu	Happy to hear @realDonaldTrump accepted my challenge to debate one-on-one: https://t.co/mikc6fXZei
8	1987: Donald J. Trump Celebrated As Model Citizen in #NYC. Remember TV without HD? #NYPrimary #MAGA #Trump2016 https://t.co/5ROvjhJyAK	Ive released 9 years of tax returns. RT if you agree its time for Donald Trump to release his! https://t.co/08whtFVC0r
9	The Post endorses Donald Trump https://t.co/bGIxG1DnZO https://t.co/1lC8E4Xi89	Donald Trump says wages are too high. Really? Hard-working Americans don't think so. https://t.co/5oEK9UhGI1 https://t.co/1z0tuCedJa
10	I'm a Veteran. I was born in Mexico, but I am here Legally! I am not racist! I support Donald Trump #Trump2016 https://t.co/pD076BcWU5	It's not just Trump: Every Republican presidential candidate has attacked women's health and rights. https://t.co/3TQdSvYTSs

al. [4], Kase et al. [5] study crowd-sensing and fact-finders in the context of war and conflict situations. In this chapter, we solve the orthogonal problem of separating the polarity classes.

Polarization in social network can be viewed as a community detection or graph partitioning problem [137, 138]. We do not directly apply such techniques because of the presence of neutral sources and assertions. Moreover, the requirement of fusing muliple signals to converge to an expected solution required an optimization framework. Sentiment analysis [157, 176] can also be viewed as a related technique to uncover polarization. However, in our case, sentiment analysis is not directly applicable because the positive and negative classes in sentiment analysis can be orthogonal to the polarization group in question. Moreover, sentiment analysis is a *supervised* technique, while our technique is *unsupervised*. Sentiment analysis can require training and language model to map the sense of the text to sentiments. Even after training, such techniques can miss the assertions that are composed of sentiment-neutral wording, but semantically biased toward a certain side. On the other hand, our method looks at the source information and exploits the network structure to uncover polarity. As it does not consider text information, assertions that are not well connected in the network can be misclassified.

Finding a social-influence network, or source-dependency network has been studied in prior literature [130,132,135]. In this chapter, we use the maximum likelihood approach proposed by Netrapalli and Sanghavi [130] to generate the social dependency matrix used as an input to our algorithm. In addition, bagging [180] and boosting [181] are two main solutions in ensemble learning [182]. In this chapter, we follow this idea by filtering out bad separations by identifying Jaccard distance among the candidates and bagging filtered candidates.

5.6 Summary

In this chapter, we have presented a matrix factorization and ensemble based gradient descent algorithm to uncover polarization in social networks. We have evaluated our algorithm in the context of ongoing disputes, conflicts, or controversies as polarized situations. Experiments show that it can separate the tweets of different polarities by looking just at the source-assertion network and the social dependency network, and can be more than 90% accurate. Our algorithm performs much better than supervised techniques like sentiment analysis. Moreover, it also performs around 20% − 30% better than the community detection approaches, when the separation between the sources or the assertions of different polarities is obscured because of the presence of a large neutral network. If a particular source or assertion is not well connected to the network, the method can misclassify. Correctly estimating such cases with the help of additional information, deriving confidence bounds for the detected polarity, and jointly estimating polarity of the tweet with its veracity will be addressed in future works.

CHAPTER 6

SOCIALTROVE: A SUMMARIZATION SERVICE FOR SOCIAL SENSING

This chapter describes a general-purpose self-summarizing storage service, called SocialTrove. The objective is to obtain a *representative sampling* of large data streams at a configurable granularity, in real-time, which can be used to build an information network for subsequent consumption by the algorithms presented in earlier chapters. SocialTrove summarizes data streams from human sources, or sensors in their possession, by hierarchically clustering received information in accordance with an application-specific distance metric. It then serves a sampling of produced clusters at a configurable granularity in response to application queries. While SocialTrove is a general service, we illustrate its functionality and evaluate it in the specific context of workloads collected from Twitter. Results show that SocialTrove supports a high query throughput, while maintaining a low access latency to the produced real-time application-specific data summaries. As a specific application case-study, we implement a fact-finding service on top of SocialTrove.

6.1 Overview

This chapter describes the design, implementation, and evaluation of SocialTrove; a self-summarizing storage service for social sensing applications. The service offers an API that allows applications to access their data at different degrees of summarization in a configurable manner. SocialTrove is motivated by the advent of an age of data overload, brought about by the increasing availability of smart devices with instant data collection and sharing capabilities, as well as by the growth of social network broadcast, such as microblog upload on Twitter. Early autonomic computing envisioned machines with self-* properties that independently meet application needs. The rise of social networks in the present decade, together with the proliferation of smart devices and other digital data sources, suggests that an increasing application need in the foreseeable future will be one of summarizing large volumes of redundant data for subsequent processing. This motivates development of a *general-purpose summarization service*.

In this chapter, we focus on *social sensing* applications. We refer by social sensing to those applications, where humans share information on themselves or their environment, either directly (e.g., by blogging) or using sensing devices in their possession (e.g., sensing on a smart phone). The application features a back-end, where collected data is stored, which is the focus of our work. Social sensing applications encompass participatory sensing [183–186], opportunistic sensing [23, 187, 188], and use of humans as sensors [2–4, 95, 125, 189]. For example, smartphone users on a participatory sensing campaign might run a geotagging application that allows them to upload GPS locations of items of interest via the phone. The application might also allow them to describe these items using text tags, or to supply images. For another example, Internet-connected vehicles may upload speed information periodically from on-board navigation systems, allowing the back-end servers to compute city traffic speed of different streets. In recent work, the authors explored the use of social networks, such as Twitter, as sensor networks, observing that many tweets can be viewed as bits of information about the state of the physical world. For such sensor networks, an application might construct physical state estimates from "human sensor" observations [2–4, 108]. A common characteristic of social sensing systems exemplified above is that they generate large amounts of redundant data. The underlying data objects may be different, depending on the application. The simplest way to summarize data is to reduce redundancy by offering a *sampling* of the original data set, where the selected samples are minimally redundant. We call such a sampling policy, *representative sampling*. A challenge, therefore, is to develop a representative sampling service agnostic to the data type.

SocialTrove is an exercise in building a *general-purpose* representative sampling service that reduces redundancy in large data sets. The service allows application designers to specify an *application-specific* distance metric that describes a measure of similarity relevant to this application among data items. Based on that application-specific measure, the service hierarchically clusters incoming data streams in real time, and allows applications to obtain representative samples at arbitrary levels of granularity by returning cluster heads (and member counts) at appropriate levels of the cluster hierarchy.

An important design consideration in developing our service is scalability. When data are large, if the observations are stored in a cluster-agnostic manner, retrieving a representative summary would require scanning the entire set of observations, thereby communicating with many machines and decreasing throughput. Instead, SocialTrove stores content in a *similarity-aware* fashion, according to the application-specific similarity metric. We implement SocialTrove and evaluate its performance in the context of summarizing Twitter data. We demonstrate that it outperforms the alternate mechanisms in terms of both (summary) query latency, and maximum query throughput. To demonstrate an application that uses Twitter data summaries, we built a fact-finding service [3] that uses the produced summaries to determine which observations are more credible in the presence of noise, errors, and conflicts. We observe that the fact-finder implementation on top of SocialTrove required significantly fewer lines of code than a standalone service.

The rest of this chapter is organized as follows. Section 6.2 describes the main interface exported by SocialTrove as a self-summarizing storage service. In Section 6.3 we present the distributed architecture of the SocialTrove runtime. Section 6.5 presents microbenchmarks and a performance evaluation. We review the related work in Section 6.6. The chapter concludes with a discussion in Section 6.7.

6.2 A Self-Summarizing Storage Model

Our goal in this chapter is to build a (data storage) service that allows an application to retrieve summaries of their data at arbitrary levels of granularity based on an application-specific redundancy metric. We call such a service, *self-summarizing* storage. The main purpose of summarization is to reduce data redundancy by selecting data samples that are minimally redundant. Towards that end, SocialTrove employs a hierarchical clustering scheme and returns data samples constitituting cluster-heads at a configurable granularity (together with the sizes of corresponding clusters). Finally, we aim to design the service that is agnostic to the data type, so that it may be reused in different application contexts. Hence, we allow applications to define their own application-specific distance metric between data objects, and cluster objects in the corresponding feature space. The SocialTrove API is carefully designed not to make assumptions regarding the feature space in which application objects live, and yet perform clustering, store clusters, and serve summary queries in an efficient manner at different levels of granularity.

In accordance with the above design requirements, the fundamental abstraction and main "citizen" of SocialTrove is the abstract *data object*. It is an opaque data type that SocialTrove itself does not interpret. Instead, it stores object records that are tuples of (`ObjectSource`, `ObjectHandle`, `FeatureVector`), where `ObjectSource` specifies the ID of the input source (e.g., sensor ID, camera ID, or social network user ID) from which the object was obtained, `ObjectHandle` is a handle to the abstract data type, and `FeatureVector` is a placeholder for the object's application-specific feature vector (not computed by SocialTrove).

Further, the service offers two interfaces; (i) a *customization interface* that allows applications to define their application-specific features and distance metrics for objects, and (ii) a *summary query interface*, that allows applications to retrieve data summaries at different degrees of granularity. We begin the chapter by describing those interfaces first to give the reader, respectively, an understanding of (i) the way we attain independence of the service from the application-specific data type, and (ii) the functionality we offer to the application.

6.2.1 The Customization Interface

To customize SocialTrove to the summarization needs of a particular application, two application-specific callback functions must be written by the application developer. These functions will be called by SocialTrove. Namely:

- `Vectorize(u)`: SocialTrove requires applications to implement a callback function, called `Vectorize()`. SocialTrove passes an object handle, `u`, to this function. The function returns a corresponding feature vector, `FeatureVector`. Note that, SocialTrove never interprets the incoming objects themselves or assumes their format. Rather, only

`Vectorize()` is aware of what an object means. Similarly, SocialTrove does not interpret the output feature vector. It is stored as an opaque data type in the object's record.

- `Distance(u.FeatureVector, v.FeatureVector)`:
  SocialTrove requires applications to implement a callback function, called `Distance()`, that computes the distance between two objects, `u` and `v`, based on their feature vectors. As mentioned above, SocialTrove itself never interprets the feature vectors, as they are application specific. Instead, it treats the feature vectors generated by the `Vectorize` function as an opaque data type. A handle to the data type is stored in the object's record. The `Distance()` function operates on these vectors and returns a scalar distance value. We require that the scalar distance value obey the triangle inequality. In other words, we require that $distance(u, v) + distance(v, w) \geq distance(u, w)$.

The above interface is flexible and supports the needs of very different applications. For example:

- *Scalar measurements*: In applications involving scalar sensor values, `Vectorize()` trivially returns the sensor measurements. `Distance()` returns the difference between two measurements.

- *Vector measurements*: In applications where objects such as, environmental measurements, are associated with metadata, such as time and location, `Vectorize()` might focus on metadata elements of objects, viewed as a feature vector. `Distance()` might then return a weighted Cartesian distance between feature vectors, where weights reflect the relative impact of differences in the corresponding dimension on the likelihood of similarity between objects. For example, say, we know that a particular variable does not change much over time, but has large spatial variations. Hence, the weight of the location dimension is set larger and the weight of the time dimension is set smaller. This allows computing a scalar similarity measure between any two objects and estimating measurements at one time and location using a nearby object in the feature space (albeit from a different time and location).

- *Pictures*: In applications involving visual objects, `Vectorize()` might apply a library of image processing tools to extract relevant image features. `Distance()` may compute visual similarity between images based on these features.

- *Text and tags:* In applications where objects constitute small amounts of text (such as tweets or tags associated with images), `Vectorize()` might split the text entry on whitespaces into different tokens (words). `Distance` may be applied on pairs of vectors (token lists) by counting the proportion of similar tokens. The Tanimoto distance and the Angular distance are suitable distance metrics in this space [177, 190].

The point of the above discussion is to demonsrate versatility. Many application domains (e.g., vision and speech) already have well-defined distance metrics between objects. The definition of vector spaces and distance metrics is thus out of scope for SocialTrove. In our case study, we demonstrate a distance defined on short text (tweets), showing how it leads to meaninful summaries of human observations.

6.2.2 The Summary Query Interface

Using the above two application-specific callback functions, SocialTrove has all it needs to perform hierarchical clustering in real time, as will be described later in this chapter. With clusters at different levels of granularity constructed, SocialTrove exports an interface to retrieve data summaries at different degrees of granularity. A summary in our service is given by a list of cluster-heads. For each cluster-head, the service allows one to optionally retrieve a member count (i.e., count of objects in the same cluster) or a member list (list of object record handles for objects in the same cluster). Remember that an object record is a tuple, (`ObjectSource`, `ObjectHandle`, `FeatureVector`), specifying the source ID, feature vector and object handle. Hence, given a list of record handles, the application can retrieve the corresponding objects, sources, or features, depending on how much data they need.

For example, an application interested in the degree of data corroboration only, might retrieve a summary that consists of cluster heads and member counts only. An application that also needs to know which sources reported the observations in the cluster (e.g., in case some are trusted more than

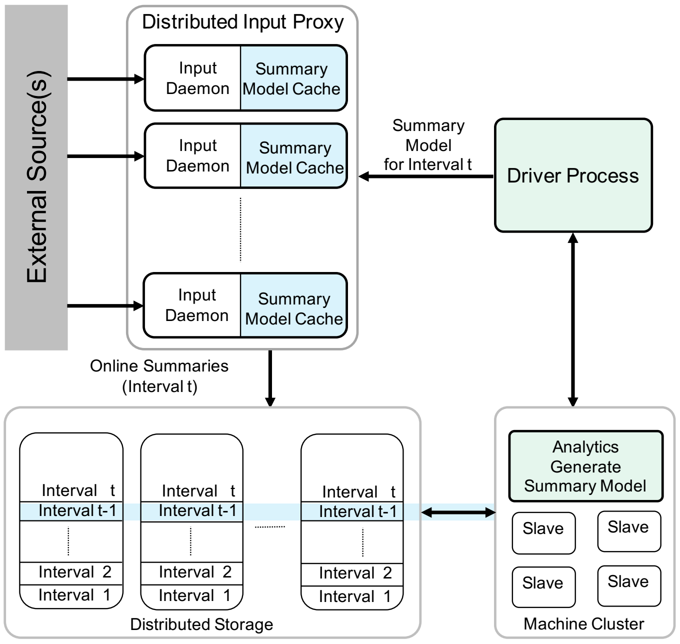


Figure 6.1: SocialTrove system design

others), can retrieve the member (handle) list and inspect the sources. An application interested in statistics over clusters may also inspect the feature vectors. The SocialTrove runtime is described in the following sections.

6.3 SocialTrove Runtime

SocialTrove is designed for large-scale social sensing services where collected data is too big for a single machine. Hence, we design and implement SocialTrove on a machine cluster. In this section, we describe the design of the runtime environment that makes it scalable. The design is based on two observations:

- Latency and throughput are improved by limiting global state updates to only once per a configurable interval, called the *batching interval*. Hence, incoming data are buffered until enough of it is present, then a batch process makes an update to existing clusters, once per batching interval. Batching amortizes run-time overhead across a larger body of input data. The batching interval (e.g., 5 minutes) is thus a configurable parameter that offers a trade-off between data freshness and update overhead.

- Availability is improved by noting that social sensing content is likely to exhibit temporal locality. Hence, state does not change significantly across batching periods, making further optimizations possible.

6.3.1 System Components

Figure 6.1 shows the components of SocialTrove and their interactions, described below.

Data Input Proxy

We envision SocialTrove to sit on top of a data collection service. This service will interact with the various sources and will supply a stream of real-time data to SocialTrove for summarization. In the current implementation, the input is supplied as a set of tuples (`ObjectSource`, `ObjectHandle`) in JSON [164] format. In our particular application example, we replace the data collection service with Twitter and write a simple interface that uses Twitter API to stream tweets. In this instantiation, `ObjectSource` is a Twitter user ID, and `ObjectHandle` is a handle to a tweet object (including text and metadata).

The input proxy is composed of several data input daemons that receive streaming objects and must resolve where to store each. This resolution is done by consulting a *Cluster Model*, also known as *Summary Model*, which keeps track of the existing clusters for the present batching interval and the mapping from these clusters to individual storage machines.

Client Query Proxy

Similarly to the input daemon nodes, are the client query proxies. (The proxies are not shown in Figure 6.1 to keep it simple.) They function like input daemons and cache the cluster model as well. Instead of clustering collected data from external sources, the query proxies receive queries from SocialTrove clients, and fetch the matching data summaries from the storage nodes using their locally cached cluster model.

Cluster Model (also known as Summary Model)

The Cluster Model is a data structure that contains the set of cluster centroids, along with their hierarchical relationships. Computing an accurate cluster model requires knowledge of all the data objects, including those that would be arriving in future. Because the data objects arrive as a stream, having an accurate model is often not possible. Maintaining a streaming cluster model that updates the existing clusters as the data objects arrive would be close to accurate [191]. In this scenario, the input daemons would require exclusive locks to update the model at every insertion, and all the proxies would need to synchronize the updates to maintain consistency. Such a write-heavy scheme would greatly reduce both throughput and response time of the system, and would not be scalable as a service.

To solve this problem, SocialTrove maintains a system wide batching interval of Δ minutes. A new cluster model is computed using the recently collected data objects, and advertised at the beginning of every interval. The input daemons and the client query proxies cache the cluster model (or portions of it) in their main memory that remains consistent until the interval ends. In later sections, we discuss different solutions to organize and update the cluster model.

Storage

The storage nodes store actual data objects in a clustered form. The objects are received from the input daemon nodes that cluster incoming data objects using the cluster model. The clusters stored in the storage nodes are partitioned and indexed according to the interval they were received. For a particular interval, the union of the respective partitions over all the storage nodes constitutes the 'sensed universe' for that interval.

Model Update Routine

The model update routine is run every batching interval of Δ minutes. During interval t, it considers the data objects received in interval $t - m$ to $t - 1$ (the previous m intervals), and computes the cluster model that the data input and the client query proxies will use during interval $t + 1$ (the next interval). As an option, output of this routine can be fed back to the storage nodes so that the data objects received during interval $t-1$ could be readjusted.

6.3.2 Cluster Model Management

The Cluster Model is a key part of SocialTrove. It maintains a set of centroids as the cluster heads of the existing clusters. For an incoming data object, the input daemons traverse the cluster model to find the centroid of the cluster this object belongs to. Similarly, to serve the applications running on top of SocialTrove, the client query proxies traverse the cluster model to find matching vectors. Depending on how the distributed cluster model is organized and maintained, there can be different trade-offs and flexibilities the system can offer.

SocialTrove is scalable by virtue of efficient realization of these insert and lookup queries. If there are k centroids and n incoming data objects in an interval, the naive and most versatile implementation requires a query object to be compared with all the centroids to find the nearest match, resulting in a $\mathrm{O}(nkd)$ algorithm when the whole cluster model fits in the cache and the comparisons take $\mathrm{O}(d)$ time. The comparison time can be considered a constant. We also observe that the cluster sizes in socially sensed data objects approximately follow a long tail distribution, and k is roughly of the same order as n. Hence, the naive algorithm requires $\mathrm{O}(n^2)$ time when the entire cluster model fits in the cache. This naive solution would not be scalable.

If object distances, however, follow the triangle inequality, some distances can be inferred from others and hence the above extensive comparison is an overkill. Given a metric distance space, we thus build a nearest neighbor data structure (tree) during clustering. The insert and search operations on the clustered data objects can then be performed efficiently using the tree. Disjoint partitions of the tree are mapped to different storage nodes, so that an input daemon can quickly decide which storage node to forward the incoming data object, and a query proxy can quickly decide which storage node(s) to forward the user query to.

If the distance function satisfies all the properties of a metric, (i) non-

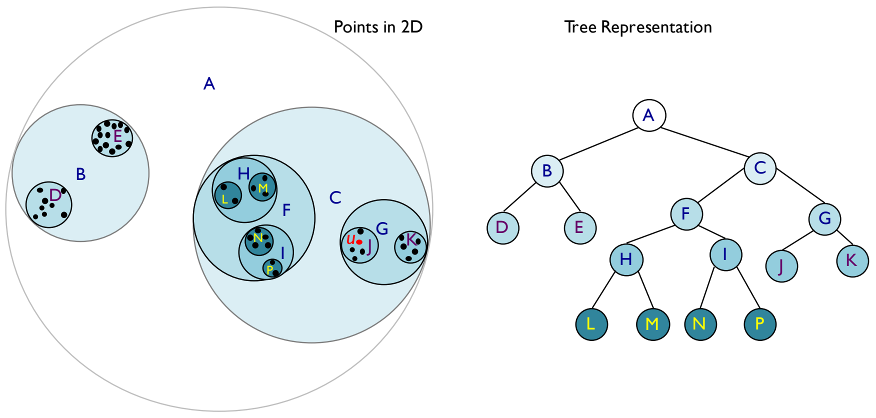


Figure 6.2: Mapping a set of points in two dimensions to a tree

negativity, (ii) small self-distance, (iii) isolation, (iv) symmetry, and (v) triangle inequality) [192], it enables us to use rich nearest neighbor data structures like M-tree [193] or Ball-tree [194] to perform k-`means` [195] clustering efficiently. It is trivial to satisfy the first four properties. The triangle inequality may not be satisfied by all distance measures. However, if any of the last three conditions fail; provided the other four are satisfied, it is possible to find a function through transformation, which is a metric function [192].

The euclidean distance function follows triangle inequality and is a metric function. In fact, all normed vector spaces are metric spaces, if we define $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$. Some distance measures like KL-Divergence or Mahalanobis Distance do not follow triangle inequality, but instead follow another property called Bregman Divergence. There are Nearest Neighbor data structures inspired by Ball-tree; for example Bregman Ball-tree [196] that can be used in this case for efficient clustering. These, however, are currently not implemented on SocialTrove.

In SocialTrove, the cluster model is represented as a binary tree of centroids. The tree is constructed using a divide and conquer paradigm. At every stage, the current set of vectors is partitioned into two sets, using a 2-`means`[1] clustering algorithm. The centroids of the two sets are considered as the two children of the centroid of the original set. This process continues until we arrive at a set of vectors with diameter less than a threshold, which is considered as a single indivisible cluster. The data objects are separable in this way, provided the distance function satisfies the triangle inequality (and

[1]k-`means` clustering with $k = 2$

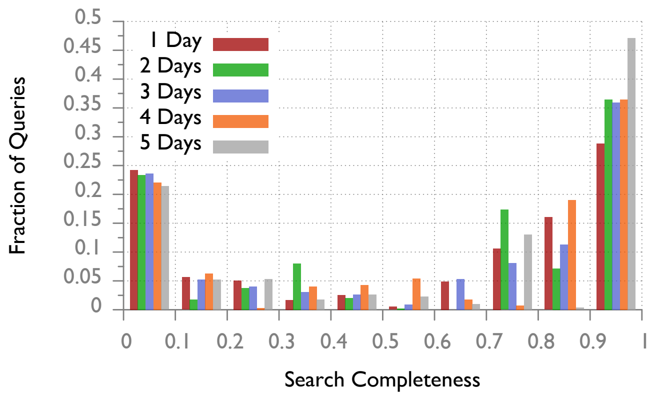


Figure 6.3: Distribution of search completeness

all the properties of a metric). The tree is generated by the model update module, and synchronized every interval to the input daemon and the proxy nodes that cache it. As an illustrative example, Figure 6.2 shows a set of points in two dimensional space and maps those to a corresponding binary tree that divides the space using the euclidean distance among the points as a distance metric.

For each collected object, the input daemons find the closest centroid from the tree, and assign it to the corresponding cluster. If there are k centroids, this operation can be performed quickly, in $\mathrm{O}(\log k)$ time. However, for the clustering performed this way to be correct and the lookup operations to succeed, the nodes of the tree requires perfect centroids for all objects that would be collected during the current interval, which is not possible.

We assume that objects collected in the present interval are correlated with those collected in past intervals. Hence, we estimate the cluster tree for interval t during interval $t-1$ by clustering the objects collected in last m intervals (i.e., intervals $t-m-1$ to $t-2$).

To check the validity of our assumption, experiments were performed using Twitter data as the input by clustering past tweets to build the cluster tree, and inserting new tweets using it. The objective was to check how complete the lookup operations would be, if a scheme for quickly clustering recent tweets based on a past model is used. The hashtags present in the current set of tweets were then used as search queries. We used a very large Δ, of 1 day, as a very extreme case. Figure 6.3 shows the distribution of search completeness for the newest tweets for different values of m from 1 day to 5 days.

The plot confirms that tweets from present and past intervals are correlated. The plot also reveals one potential limitation of this method; false negatives. Over 20% of user queries could not find any match at all, and 40% could only find at most 50% of the desired results. This problem is visible with high dimensional data like text or tweets, where some dimensions were not known when the summary model was generated. As the cluster tree is computed and circulated to the input daemon nodes in synchronous intervals, it fails to look up using query terms that are unique to the present interval.

As a solution, we add an asynchronous component to the cluster model using Bloom filters [197]. There are Bloom filters corresponding to every node of the cluster tree. Dimensions (keywords, in case of tweets) unique to the present interval are locally inserted to the Bloom filters corresponding to the tree nodes visited by an incoming data object. Crawlers use a gossip protocol [198] to propagate their local updates to the Bloom filters. These updates are not expensive because only relative changes are sent over the network, which are easily merged using bitwise ORing. Lookups are performed using the Bloom filter. A Bloom filter has a 100% recall rate; hence it solved the aforementioned problem of false negatives when searching with the query terms unique to the present interval.

Insertion

New object insertions use the cluster model to find the correct cluster for incoming objects. Here, we illustrate using Figure 6.2 how insertions are performed. Suppose an incoming object u (the red point in Figure 6.2) arrives. To assign the nearest cluster to this point, it is at first compared with centroids B and C. Distance from centroid C is found to be lower. Thus, the object is pushed down that branch of the tree. Centroid C has two children, namely F and G. Again, object u is compared to both. The distance from G is found lower. Thus, the object is pushed down that branch. The two children of G (namely J and K) are compared to u next. The incoming object is closer to J, which is a single cluster. Hence, u is assigned to cluster J.

The pseudo-code for insertion using the cluster tree is shown in Figure 6.4. $node_{sync}$ corresponds to the synchronized component of $node$ that is updated

```
 1: procedure INSERT($u$)                                          ▷ Data object $u$
 2:     $node_{sync} \leftarrow root(CM_{sync})$                   ▷ Global cluster model
 3:     $node_{async} \leftarrow root(CM_{async})$                 ▷ Local cluster model
 4:     while $node_{sync}$ is not leaf do
 5:         $l_{sync} \leftarrow left(node_{sync})$                ▷ Left child
 6:         $l_{async} \leftarrow left(node_{async})$
 7:         $r_{sync} \leftarrow right(node_{sync})$               ▷ Right child
 8:         $r_{async} \leftarrow right(node_{async})$
 9:         if $dist(u, l_{sync}) < dist(u, r_{sync})$ then
10:             $node_{sync} \leftarrow l_{sync}$
11:             $node_{async} \leftarrow l_{async}$
12:         else
13:             $node_{sync} \leftarrow r_{sync}$
14:             $node_{async} \leftarrow r_{async}$
15:         end if
16:         for all $token \in u$ do
17:             Set $node_{async}[token]$                          ▷ Update Bloom filter
18:         end for
19:     end while
20:     Append $u$ to the cluster $node_{sync}$                    ▷ Invoke RPC
21: end procedure
```

Figure 6.4: Algorithm to insert an object

every interval, and $node_{async}$ corresponds to the asynchronous components that are maintained through Bloom filters. Lines 4–19 push the incoming object u down the tree. Lines 9–15 compare the new point with the two children of the presently considered node of the tree and decide which branch to take next. As the incoming object traverses down the tree, the local Bloom filters of the corresponding nodes are updated (which would be later propagated to the data input and the client query proxies). In Line 20, the cluster that u belongs to has been decided and the corresponding cluster summary is pushed to the in-memory distributed cache at this point.

Lookup

Lookups use the asynchronous component of the cluster model to find the correct cluster summaries related to an incoming query. Please note that, for an insertion, the incoming data object is assigned to only one cluster, which is nearest from it. The incoming data items are expected to follow the trend of the existing clusters, so that the summary model can be used to find the

```
 1: procedure LOOKUP($w$, $d_q$)                                ▷ Query object $w$
 2:     $node_{async} \leftarrow root(CM_{async})$              ▷ Local cluster model
 3:     $result \leftarrow \varnothing$                         ▷ Set of matching objects
 4:     if $dist(w, node_{async}) \leq d_q$ then
 5:         EXPLOREBRANCH($w, d_q, node_{async}, result$)
 6:     end if
 7:     return $result$
 8: end procedure

 9: procedure EXPLOREBRANCH($w, d_q, node, result$)
10:     if $node$ is not leaf then
11:         $l \leftarrow left(node)$                           ▷ Left child
12:         if $dist(w, l) \leq d_q$ then
13:             EXPLOREBRANCH($w, d_q, l, result$)
14:         end if
15:         $r \leftarrow right(node)$                          ▷ Right child
16:         if $dist(w, r) \leq d_q$ then
17:             EXPLOREBRANCH($w, d_q, r, result$)
18:         end if
19:     else
20:         Append cluster $node$ to $result$
21:     end if
22: end procedure
```

Figure 6.5: Algorithm to lookup cluster summaries

nearest cluster. However, for a lookup, the queries can be any point in space. The response is a set of cluster summaries within a mentioned distance from the query.

Figure 6.5 presents the pseudo code. Lines 4–5 decide if the query object w is within a specified distance d_q of the root node. If it is not, it is decided that the query does not match any of the existing summaries in the model. If the distance is within d_q, Lines 10–21 traverse the tree, taking the branches for which the distance of the centroid is less than the specified threshold d_q, and pruning when it is not.

Model Update

Model update is an offline job that runs once per batching interval. It considers the objects collected in the previous m intervals, and constructs the cluster model by repeatedly performing 2-`means` clustering. Because the distance function satisfies triangle inequality, divisions performed at each stage

```
 1: procedure GenerateModel($S, d_c$)                         ▷ Set of objects $S$
 2:     $root \leftarrow mean(S)$                             ▷ Calculate centroid of $S$
 3:     if $diameter(root) > d_c$ then
 4:         TwoMeansModel($root, d_c$)                        ▷ Non-blocking
 5:     end if
 6: end procedure

 7: procedure TwoMeansModel($node, d_c$)
 8:     ▷ $node$ must be divisible in atleast two clusters.
 9:     ▷ TwoMeans uses 2-means clustering to
10:     ▷ partition $node$ into two clusters $l$ and $r$.
11:     $(l, r) \leftarrow$ TwoMeans($node$)                  ▷ MapReduce job
12:     $left(node) \leftarrow l$                             ▷ Assign $l$ as left child
13:     $right(node) \leftarrow r$                            ▷ Assign $r$ as right child
14:
15:     ▷ Calls to TwoMeansModel are independent,
16:     ▷ asynchronous, and can be scheduled in parallel.
17:     if $diameter(l) > d_c$ then
18:         TwoMeansModel($l, d_c$)                           ▷ Non-blocking
19:     end if
20:     if $diameter(r) > d_c$ then
21:         TwoMeansModel($r, d_c$)                           ▷ Non-blocking
22:     end if
23: end procedure
```

Figure 6.6: Algorithm to generate summary model

are independent, and are scheduled in parallel for further division.

Figure 6.6 presents the pseudo code. d_c is a threshold parameter the algorithm uses to decide if the current set of objects are distant enough to be partitioned into two clusters. Line 2 initializes the root node of the tree. Line 11 calls the TwoMeans procedure to perform a 2-means clustering. In reference to Figure 6.2, if C is the current set of points, F and G are calculated in line 11. For a large set of data objects, this is an expensive operation, and we use a MapReduce framework to parallelize the workload [199]. Lines 12–13 updates the tree with the newly calculated centroids. At this point, the problem has been divided into two independent subproblems. Line 18 and 21 schedule new invocations of TwoMeansModel in parallel, and the process continues until the diameter of the current set is less than d_c.

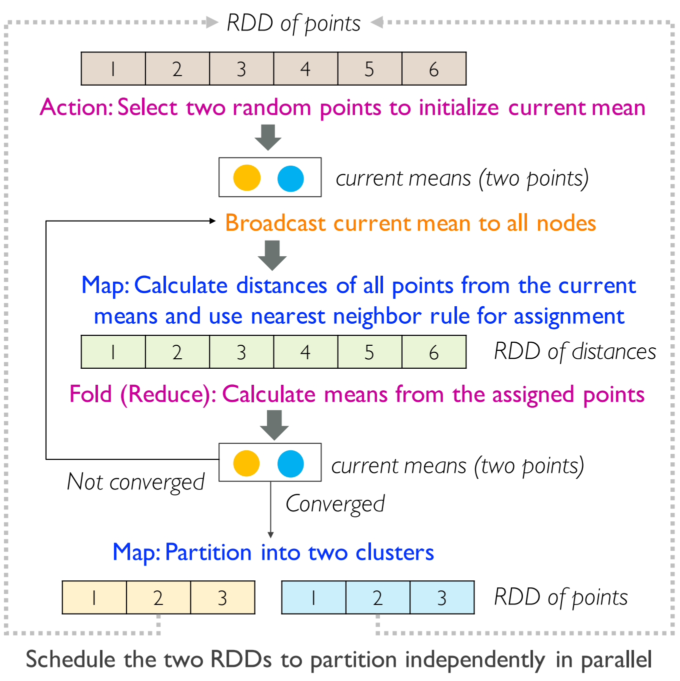
6.3.3 Implementation

SocialTrove runs in UIUC Green Data Center [12]. We use Python to implement a data collection service to provide input data. Apache Thrift [200] is used as a Serialization and RPC framework. Memcached [201] is used as a distributed in-memory cache layer for the input data and the client query proxies. Apache Hadoop [202] and Spark [203] are used for offline analytics.

The input data objects are sent by input data daemons to be stored in a Hadoop Distributed File System (HDFS) [202]. There are 23 machines with one 6-core 2.0 GHz processor (Intel Xeon E5-2620), 16 GB memory, and 1 TB of storage. The model update routine has been implemented using Java, which runs on Apache Spark [204], in a subset of the available machines. Spark has been configured to run in Standalone mode (i.e., without Yarn [202] or Mesos [205]). Each of the Spark slaves runs one worker process using 12 GB memory. Due to higher memory requirements of the Spark tasks, 30% of the memory is reserved for caching the RDDs (Resilient Distributed Datasets) [206] instead of the default allocation of 60%. The remaining 9 GB is available for the Java heap.

A machine with two 10-core 3.0 GHz processors (Intel Xeon E5-2690 v2) and 128 GB memory works as the driver machine. The driver machine commands the worker machines to build RDD (Reslilient Distributed Datasets) using the data that has been collected over the past interval. It uses the GenerateModel algorithm (Figure 6.6), which generates a summary model by repeated use of `2-means` clustering as a subroutine to bisect the distributed dataset.

Figure 6.7 shows the flow of distributed computation in generating the summary model. At each step, two objects are randomly selected from the present dataset that act as initial centroids. These two centroids are broadcast to all workers. After this broadcast, the driver machine initiates a map phase (known as RDD Transformation in Spark) so that the workers calculate the distance of each object in its collection from the two broadcast centroids, and assigns it to the centroid with the smaller distance. After that, the driver issues a reduce phase (RDD Action) that calculates two new centroids from the previous assignments. The new centroids are broadcast to the workers again, and the process continues until it converges and results in two clusters. The parent RDD is then partitioned into two child RDDs cor-



RDD of points
1 2 3 4 5 6
Action: Select two random points to initialize current mean
current means (two points)
Broadcast current mean to all nodes
Map: Calculate distances of all points from the current means and use nearest neighbor rule for assignment
1 2 3 4 5 6
RDD of distances
Fold (Reduce): Calculate means from the assigned points
current means (two points)
Not converged
Converged
Map: Partition into two clusters
1 2 3
1 2 3
RDD of points
Schedule the two RDDs to partition independently in parallel

Figure 6.7: Flow of map, fold (reduce), and broadcast operations in Spark to recursively partition an RDD of points

responding to the newly formed clusters, each of which are scheduled to run TwoMeansModel (Figure 6.6), in parallel. Due to the overhead of small jobs, once an RDD becomes small enough, we start bisecting it in a single thread, instead of spawning new Spark jobs. Once clustering is complete, the data input proxies and the client query proxies update their cluster models accordingly.

6.4 An Application Case Study

In this section, we present an example of using SocialTrove. Our cases study describes a simplified implementation of a fact-finding service, reported in recent literature [3], on top of SocialTrove. The fact-finder views humans as sensors, and Twitter as a sensor network. It performs maximum-likelihood estimation to determine the likelihood of correctness of different reported observations and offers the users a list of facts that are most likely to be true (in a maximum-likelihood sense). Hence, the term fact-finder. The exact algorithm used for estimating credibility of observations has been published in previous literature. In this chapter, we reimplement the service as an example of how it could use SocialTrove.

6.4.1 Data Collection

Our data objects are tweets posted on Twitter. We developed a thin interface where crawlers periodically query Twitter using the 'Search API' [163] and generate the JSON object files input to SocialTrove. In our implementation of data collection, the Twitter query is simply be a set of keywords and a geographic radius. The output of which is a sample of latest tweets that match the query. Twitter's API also supports crawling tweets of a particular user, or crawling a timeline. The queries are subject to rate limits. In our implementation, we set up a Web-based user interface for the data collection service that can be used to create Twitter queries. Resulting JSON files are randomly assigned to SocialTrove input daemons.

6.4.2 Mapping Tweets to Vectors

The input daemons pre-process the tweets to generate feature vectors. In our implementation, each word in the tweet is a dimension of the feature vector, and is associated with a weight. Standard techniques for processing text documents suggest to (i) remove stopwords (about 750 in English language), (ii) remove high frequency and low frequency terms, (iii) use stemming, and (iv) apply TF-IDF scaling to associate a weight with each word. In practice, for our Twitter input, we found that removing the embedded URLs, symbols, and the words under a certain length resulted in acceptable fact-finding performance.

After preprocessing, a tweet is represented as a high dimensional vector in our vector space model. The order of words inside the tweet and multiplicity of occurrences are ignored. The weight of each dimension thus becomes either 0.0 or 1.0. These very sparse vectors are conveniently represented using a dictionary data structure.

6.4.3 Distance Function

Euclidean metrics $\|\mathbf{u} - \mathbf{v}\|$ fail to provide a good separation of very high-dimensional data like text. Instead, Jaccard distance [177] is a good measurement of similarity between high-dimensional sets. We use Tanimoto distance [177], which can be considered as a vector expansion of Jaccard distance. Tanimoto distance obeys the triangle inequality when the weights of the components are all non-negative [190], which is true for our vector representations of the tweets. If $\mathbf{u}$ and $\mathbf{v}$ are two vectors, their Tanimoto distance is computed by Equation (6.1), below.

$$d(\mathbf{u}, \mathbf{v}) = 1 - \frac{\mathbf{u.v}}{\mathbf{u^2} + \mathbf{v^2} - \mathbf{u.v}} \tag{6.1}$$

For three tweets t_1, t_2, and t_3, an example of applying the distance function is shown below. It is assumed that words with less than four characters are ignored and no stemming is applied.

$$\begin{aligned}
t_1 &= \text{Today is warm} \\
t_2 &= \text{I am feeling warm} \\
t_3 &= \text{Today is very warm} \\
\mathbf{t_1} &= \{\texttt{today} : 1.0, \texttt{warm} : 1.0\} \\
\mathbf{t_2} &= \{\texttt{feeling} : 1.0, \texttt{warm} : 1.0\} \\
\mathbf{t_3} &= \{\texttt{today} : 1.0, \texttt{very} : 1.0, \texttt{warm} : 1.0\} \\
d(\mathbf{t_1}, \mathbf{t_2}) &= 1 - \frac{1}{2 + 2 - 1} = 0.6666 \\
d(\mathbf{t_2}, \mathbf{t_3}) &= 1 - \frac{1}{2 + 3 - 1} = 0.75 \\
d(\mathbf{t_1}, \mathbf{t_3}) &= 1 - \frac{2}{2 + 3 - 2} = 0.3333
\end{aligned}$$

If t_1 and t_3 are considered to belong to the same cluster, then the centroid for that cluster becomes the mean of the two vectors, i.e. $\{\texttt{today} : 1.0, \texttt{very} : 0.5, \texttt{warm} : 1.0\}$

6.4.4 Ranking

The application queries SocialTrove using keywords. SocialTrove returns summaries of objects that approximately match the set of keywords according to the above distance metric. As stated earlier, the summary is composed of cluster-heads.

In the simplest implementation, the application requests cluster heads and member counts. More corroborated clusters have a large count. The application can therefore display the received cluster head tweets as facts, sorted by their degree of corroboration (i.e., member count of the corresponding cluster).

A more involved implementation of the fact-finding application is to also retrieve the list of sources per cluster in the received summary (see the section on service API). The fact-finder then constructs a distributed graph of `(source, object)` pairs, where the objects are tweets and the sources are user IDs (i.e., constructs a source-tweet graph). It does so by connecting each source to all clusters where the source is listed, and connecting each cluster (head) to all sources who contributed a member tweets of the cluster. The resulting graph is analyzed using fact-finding algorithms from recent literature [3] to jointly estimate the credibility of sources and tweet clusters in a maximum-likelihood fashion using the source-tweet graph.

6.5 Evaluation

We evaluate SocialTrove in the context of summarizing tweets. Each tweet is represented as a high dimensional vector of tokens in our vector space model. We use Tanimoto distance [177], which obeys triangle inequality [190], and can be considered as a vector expansion of the Jaccard distance [177]; a good measurement of similarity between high-dimensional sets. Our objective is to answer the following questions:

- For summarization to be a service, is it necessary to precalculate a summary model? Instead, can we generate the summaries only for the related tweets on demand as the queries arrive?

- Where and how much do we gain by organizing the summary as a hierarchy? Instead, can we build a reverse index from keywords to list of tweets (or tweet summaries)? Such techniques, used by the web search engines, are able to support a high request throughput.

- Does SocialTrove scale well?

Live tweets crawled from Twitter via its search API are subject to rate limits. Hence, we merge tweets collected during several events in the physical world, and play those back to SocialTrove. The events include Crimean Crisis (February 2014), Sochi Winter Olympics (February 2014), Syria Chemical Attack (August 2013), Boston Marathon Bombings (April 2013), Hurricane Sandy (October 2012), Hurricane Irene (August 2011), England Riots (August 2011), Fukushima Nuclear Disaster (March 2011), Egyptian Revolution (January 2011), etc. This combined set contains 4 142 586 tweets.

To the best of our knowledge, SocialTrove is the first system to offer summarization of social streams as a cloud-backend service. We did not find any corresponding system in the state of the art. Hence, we compare the performance by replacing SocialTrove components and algorithms with the following options:

- **Baseline** In this scheme we do not build periodic summary models. The input daemon randomly picks a storage machine for an incoming tweet even without deserialization (JSON parsing). To perform a lookup, the client proxies broadcast the query to all machines. The machines collect matching tweets, cluster those, and return a representative sample.

- **Indexing** This scheme does not precalculate a summary model. However, it maintains a keyword-to-storage map in memory. An incoming tweet is scanned to find keywords. For every keyword occuring in the tweet, we consult the map, and assign the corresponding storage machines. To perform a lookup, we cluster the matching tweets collected from the storage machines, and present a representative summary.

- **Summary Baseline** This scheme precalculates a summary model. It opts for a flat organization of the cluster summaries. At every interval, the model is pushed to the data input and the client query proxies, just like SocialTrove. To assign an incoming tweet to an existing cluster, this scheme computes the distance to all the existing clusters and finds

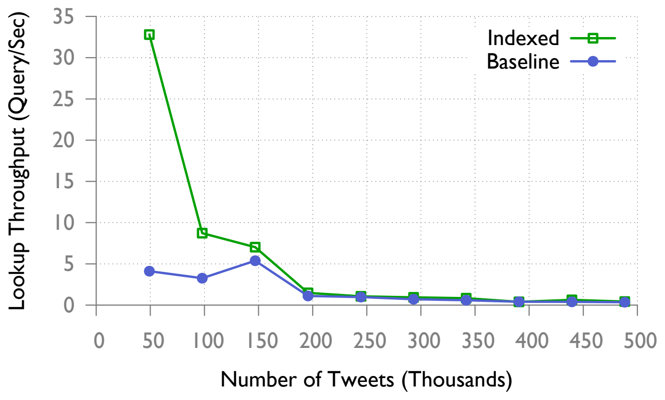


Figure 6.8: Without a summary model, lookup throughput is very low

the nearest one. To perform a lookup, this scheme again needs a linear search through the list of summaries.

- **Summary Indexing** This scheme is derived from techniques used by the web search engines. It computes the summary model, and organizes those by reverse indexing from keywords to the list of summaries. To insert a new tweet, a data input proxy extracts the keywords from it, searches only the reverse indexes corresponding to those keywords, and finds the nearest summary to assign the tweet. To perform a lookup, this scheme needs to scan the list of reverse indexes corresponding to the given keywords, and find the matching summaries.

6.5.1 Query Throughput

In our application, a query is a set of keywords. In response to a query from the application, the Client Query Proxy prepares a representative summary of the tweets that contain the given keywords. Please note the difference between returning all the results and returning a *representative summary*. The former is the application of known data structures and storage systems that can return all the matching objects. However, SocialTrove is a summarization service to deal with information overload, and as such, returns a representative sample (i.e., cluster-heads). The queries can also include an optional distance parameter, which specifies the minimum diversity among the returned samples.

Figure 6.8 shows that the throughput is very low for the Baseline and the

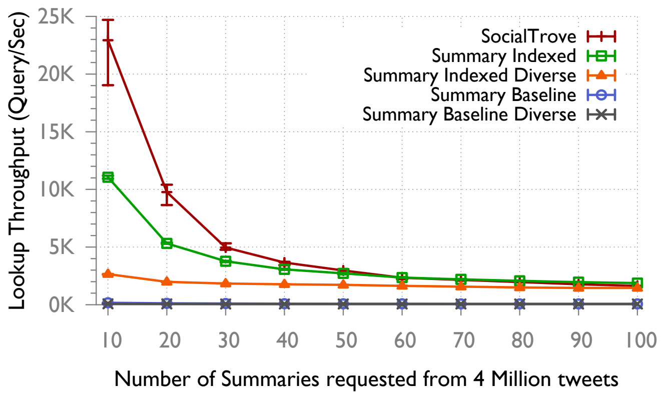


Figure 6.9: SocialTrove offers high throughput for small to medium sized requests

Indexed methods that do not pre-calculate a summary model. These methods calculate a summary on demand, in response to a query. The baseline option suffers the most in lookup throughput as it is putting load on every worker machine for every query. The indexed meachanism would have a high throughput if the queries would ask for all matching data objects instead of a representative summary. It performs better than the baseline because of the underlying indexing that provides it the set of candidate tweets without searching. However, the throughput quickly falls off towards zero as the number of tweets in the universe increases.

SocialTrove client query proxies cache the summary model in their memory once it is generated. For every query request, it traverses the tree according to the Algorithm in Figure 6.5 and finds the corresponding leaves. To answer queries, it consults the distributed in-memory cache (Memcached) to fetch a sample tweet from each of the clusters. Figure 6.9 compares SocialTrove with the other methods that prepare a summary model in advance, in a universe of 4 million tweets. SocialTrove can sustain a much higher throughput compared to the other methods, because of the hierarchical organization of the summary model. It can also incorporate the distance parameter (diversity) without any overhead because the tree had already calculated and cached the necessary distance information. The Summary Indexing method offers roughly 50% of SocialTrove throughput when the number of objects requested is small (around 20). On the other hand, the Summary Indexing method suffers when the diversity parameter is included. Baseline Indexing has the lowest throughput with and without the diversity parameter because

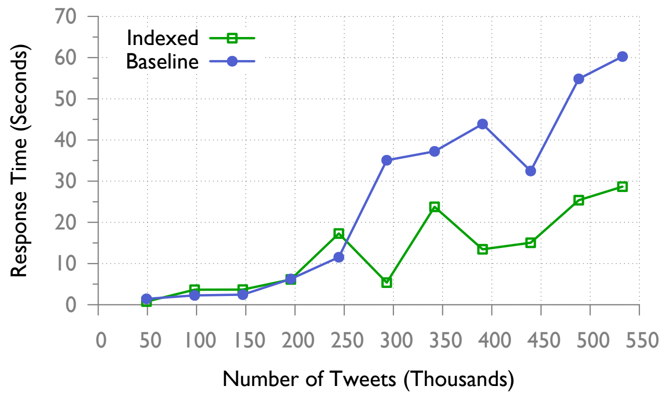


Figure 6.10: Response time for a request is often high without a summary model

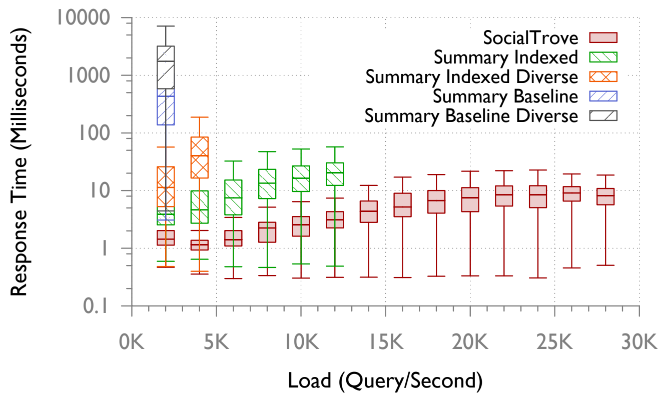


Figure 6.11: SocialTrove has lower response time compared to the other methods

of the lack of organization in the cluster summaries.

The evaluation presented here shows that when serving small requests like updating a web-page with the cluster summaries, or showing a set of tweets on a cellphone screen, SocialTrove allows high throughput. This is particularly a useful aspect of SocialTrove, because the user-facing applications often need a 'concise' amount of useful information.

6.5.2 Query Response Time

In this section we measure and compare the query response time of SocialTrove and the alternate mechanisms. Figure 6.10 shows that the Baseline and Indexed methods that do not precompute a cluster summary do not scale. These mechanisms are acceptable as a service only when the number of data objects that pass through the system at every interval is very low. Twitter receives around 500 million tweets per day [11] (or 20 million per hour), so clearly precomputing a summary model is necessary.

Figure 6.11 compares the response time between the methods that precompute the summary model in advance. We measure the response time at various levels of load (number of requests per second) in a universe of 4 million tweets, and observe that SocialTrove responds in 10 ms under heavy load, and nearly in 1 ms under light load. The summary indexing method is acceptable only when the system is lightly loaded (around 5K queries per second). If the diversity parameter is added, the summary indexing method suffers even more due to the additional distance calculations to ensure diversity.

We conclude that SocialTrove performs best, because it (1) precalculates the cluster summaries, (2) organizes the summaries as a tree, which prunes many options and reduces the search space, and (3) makes it possible to cache the summary model in main memory. If the summary model was not cached, traversing the tree would require at least one RTT (round trip time) in the network, reducing both throughput and response time. On the other hand, caching the summary model has been possible by allowing updates to the model only in synchronous intervals. This is how SocialTrove avoids a write-heavy data structure and cache consistency issues.

6.5.3 Cluster Model

We now present the time it takes SocialTrove to generate a summary model using Spark. Figure 6.12a shows the time in minutes, for different number of tweets as input, using 8 worker machines. k-`means` (in our case, $k = 2$) clustering algorithms sometimes converge to local minima, which in our case translates to unbalanced partitioning at some stages, requiring more time to finish. This is the reason for the variability in the summary generation time. Figure 6.12b shows the effect of parallelizing the clustering workload by comparing the median model generation time for 4 million tweets with different number of worker machines. Note that the data presented in figure 6.12a refer to an earlier code that used Spark Reduce operations, Java Serializa-

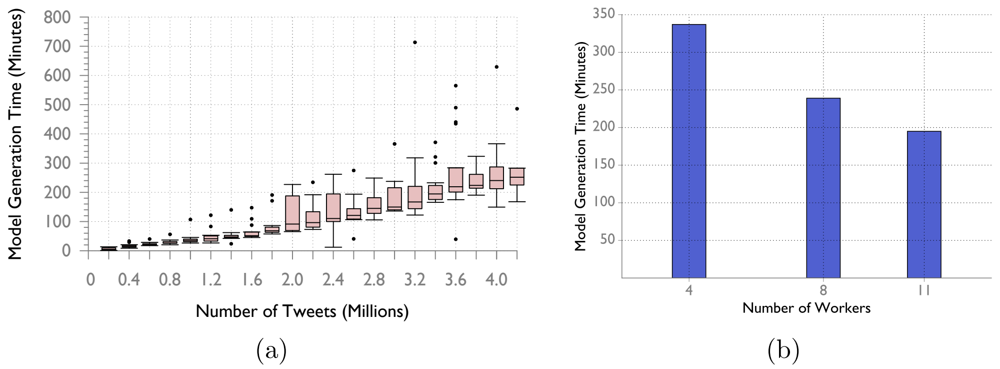


Figure 6.12: Time to generate summary using unoptimized code with (a) 8 worker machines, (b) different number of machines

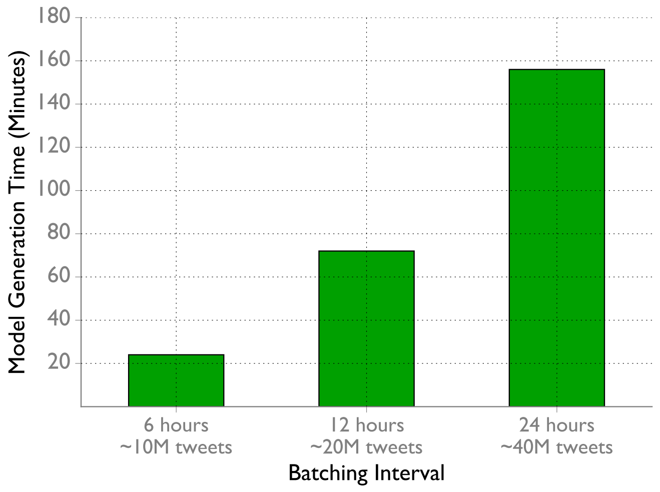


Figure 6.13: Time to generate summary from Decahose (optimized code)

tion, `java.util` library, and used string representation for the keywords. We mention this code as *unoptimized code*.

We later optimized SocialTrove to use (i) Spark Fold instead of Reduce, (ii) `it.unimi.dsi.fastutil` library, (iii) Kryo Serialization, (iv) Hash representation of the tweet keywords, which resulted in 10x improvement in memory usage and job completion time. We mention this code as *optimized code*. Figure 6.13 shows runtime for summary tree generation on a single machine, using 6 hours (10 million), 12 hours (20 million), and 24 hours (40 million tweets) of Decahose stream. Decahose provides a 10% random sample of all the tweets in a particular interval. Figure 6.14 and figure 6.15 compare the performance of scaling out and scaling up, in the context of SocialTrove.

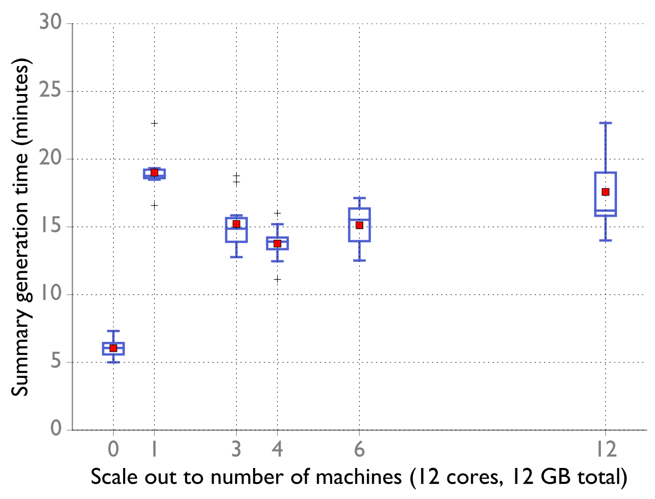


Figure 6.14: Effect of scaling out (optimized code), 4 million tweets

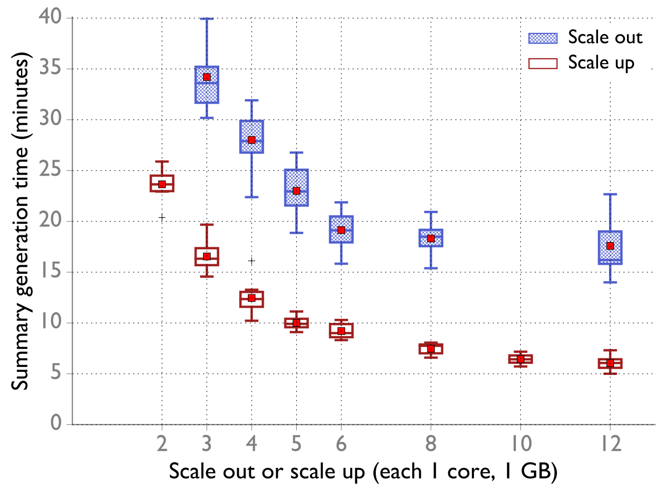


Figure 6.15: Comparing scaling out and scaling up (optimized code), 4 million tweets

In figure 6.14, we always allocate 12 cores, and 12 GB memory for SocialTrove using different number of machines. A dataset of 10 million tweets were used. In the left-most point, everything including HDFS name-node, data-node, Spark master, Spark workers, Spark driver is running in the same machine and therefore it has the fastest run time. The later points illustrate the run-time when the resource allowance is uniformly distributed across different number of machines. Figure 6.15 shows the different in runtime when we increase the resource allowance from 2 cores and 2 GB gradually up to 12 core and 12 GB. The trend is similar for both scaling out to more machines,

Table 6.1: Top summaries

Index	Top level tweet
1	Thousands at Moscow rally against Russian intervention in #Ukraine: http://t.co/6U0AIOOgQv http://t.co/kobbd7KzXY
2	Man in Ukraine plays the piano to help calm down a riot. http://t.co/fdNAc0cfJ2
3	For Crimea, Google Shows Different Borders Based on Your Location: Russia's Minister of Communications and Mass Media http://t.co/vIHGYlibOC
4	Militants in eastern #Ukraine were equipped with Russian weapons and the same uniforms as those worn by Russian forces that invaded Crimea.
5	50,000 #Ukraine supporters march in Moscow to protest Russia's intervention in #Crimea. http://t.co/qMJjYgPNxI
6	Some russian tanks on ukrainian border already painted with 'peacekeeping' slogans. How much longer until the 'humanitarian intervention'?
7	I've been speaking to @BarackObama about the situation in Ukraine. We are united in condemnation of Russia's actions. http://t.co/7Rk2k8iOIK
8	Ukrainian Defense Ministry says its lone submarine has been taken by Russians. http://t.co/lj1XP4q1BX http://t.co/mDDhQ2lqAO
9	Ukraine prepares armed response as city seized by pro-Russia forces http://t.co/ahVX7lKftT
10	Ukraine crisis: Nato warns Russia against further intervention - BBC News http://t.co/GtdmRMxAPI

and scaling up to more resources.

Table 6.1 shows sample output of the summary tweets ranked by a fact-finder application on top of SocialTrove. Our application queried SocialTrove for the set of summaries related to the keywords {Crimea, Ukraine, Russia} and ranked them according to the algorithm in [3]. Note how the tweets offer a quick insight into the highlights of the current event.

Figure 6.2 shows sample output of the summary tweets from a diversity ranking application. Our application queried the cluster model for a diverse set of summaries related to the keywords {Occupysandy}. Note how the tweets offer a quick insight into the highlights of the event. Next, the application queried for tweets related to $S4$. The output is shown in Figure 6.3. Note how the tweets are more related to the selected tweet.

Table 6.2: Querying cluster model with `occupysandy`

Index	Top level tweet
S1	RT @morningmoneyben: As a sometime critic of the Occupy movement, have to say they are out BIG TIME helping w/ Sandy relief, huge credit to them #occupysandy
S2	RT @JimGaffigan: Before the Red Cross and FEMA came to help @OccupySandy was there. Thanks! http://t.co/nlyOS7qt
S3	RT @VeganLunchTruck: Serving FREE hot #Vegan food, fresh donuts Friday 12:00-6:00ish 192 beach 96th street rockaway beach @occupysandy #sandyrelief
S4	RT @TheAtlantic: How @OccupySandy is using Amazon's wedding registry to collect donations for storm victims http://t.co/PjUwo8te
S5	RT @OccupyWallStNYC: "Capable of summoning an army with the posting of a tweet" @NYTimes http://t.co/kSskJe54 #OccupySandy
S6	Urgent: need A Lot of thermals+ponchos for #Rockaways for Weds storm. Deliver to 5406 4th Ave or 520 Clinton Ave in BK. @OccupySandy #Sandy
S7	RT @whoisMGMT: Hurricane Sandy devastated the coastal areas(cont. - http://t.co/aLtkP2fZ) @OccupySandy @wavesforwater @RockawayHelp
S8	RT @ofthespirit: you know things are changing when you get official email from the city of new york telling u to volunteer through @Occupysandy
S9	RT @OneLoveOccupy: On the ground with #occupysandy – more effective than the Red Cross? http://t.co/kVTBaX10 via @slate
S10	RT @OccupySandy: Drug store offering free meds RT @Jamester85: @OccupySandy it's awesome everyone is doing their part to help out..

6.6 Related Work

SocialTrove is motivated by the needs of data-intensive applications that handle sensor or social media data. We consider social sensing applications where redundant data are collected from people or sensors in their possession. For exampe, CabSense [207] is a crowd-sourced service that collects information on taxi cab fleets. Mediascope [208] describes a media retrieval service to query and retrieve photos taken by people directly from their mobile devices. Another recent service uses Twitter as a sensor network and models humans as noisy sensors to report and summarize ongoing events [3].

To reduce the inherent redundancy in data reported by such services, a clustering algorithm is needed. A very common one is k-`means` clustering [195]; an iterative method that repeats between selecting k means as centroids, assigning the rest of the points to the means based on similarity, and recalculating the means. Our work uses a special form of the k-`means`

Table 6.3: Querying for tweets similar to $S4$ (Amazon wedding registry)

Index	Related tweets
S4	RT @TheAtlantic: How @OccupySandy is using Amazon's wedding registry to collect donations for storm victims http://t.co/PjUwo8te
D1	RT @annawiener: A rare moment of unbridled enthusiasm about Amazon: @occupysandy is using its wedding registry to collect donations (!)
D2	RT @GregChase: Creative: @occupysandy using wedding registry on Amazon to coordinate donations for #SandyRelief http://t.co/c8QrXKya
D3	Brilliant RT @OccupyWallStNYC RT @NYCSandyNeeds Genius. RT @TheAtlantic: How @OccupySandy is using Amazon's wedding ... http://t.co/VG8TzmeQ
D4	@occupysandy are you using Amazon wedding registry to coordinate donation requests? Are deliveries coming? New registry how often?
D5	Great: RT @TheAtlantic How @OccupySandy uses Amazon's wedding registry to collect donations for storm victims http://t.co/5xL5sqqs
D6	RT @rachaelmaddux: Shop @OccupySandy "wedding registry," have supplies shipped straight to hurricane victims: http://t.co/QQe6ibBT
D7	@OccupySandy has set up a "wedding registry" on Amazon for anyone who wants to donate supplies. http://t.co/7VmVLDWZ @EcoWatch
D8	RT @gregpak: (h/t to @RNonesuch_OH for the scoop on the @occupysandy wedding registry: http://t.co/Dg6viHS7)
D9	RT @askdebra: If you don't know about the @occupysandy amazon gift registry, it's an innovative crowdfunding idea: http://t.co/11zLTyfi
D10	@OccupySandy I'm on the wedding registry team. @sandy registry temp down, but Amazon registry is on fire!!! msg with ???

algorithm, where $k = 2$, repeatedly bisecting a data set to form a hierarchy. The k-`means` algorithm is sensitive to its initialization. Different efforts have addressed this problem. For example, k-`means++` [209] avoids the issue and can be applied to SocialTrove in a straightforward way. The Buckshot Clustering algorithm [210] combines Hierarchical Agglomerative Clustering (HAC) and k-`means`. It selects $O(n)$ points randomly and runs a group average on this sample, which takes $O(n)$ time. Using the result of HAC as initial seed for k-`means` can avoid the bad initialization problem.

To apply k-`means` on streaming data, Ailon et al. [211] run online facility location algorithm on a stream of size n, to arrive at a partial solution with $O(k \log(n))$ clusters. The partial solution is followed by a ball k-`means` step to reduce the number of clusters to k. Shindler et al. [191] simplify the algorithm, which results in a better approximation guarantee. DS-`means` [212] describes a distributed algorithm to cluster data streams in a p2p environment. This system mainly uses the distributed k-`means` algorithm described by Bandyopadhyay et. al. [213], along with local instances of X-`means` [214] and gossip propagation to converge to the actual number of clusters in the system. We do not directly incorporate streaming algorithms in SocialTrove due to the need for model updates, required upon insertions. Instead, we use a batching interval to update the summary model, and exploit the "slow-changing" nature of social sensing observations in between updates.

To attain scalable implementations of data processing services, one common execution model is MapReduce [215]. MapReduce, however, is not efficient for a large class of vertex parallel iterative algorithms that have a substatial data shuffling phase. Another limitation is that the results of each round are stored on disk to be read again in the next step. Spark [203], in contrast, is an in-memory cluster computing framework that uses Resilient Distributed Datasets (RDD) to record the lineage of operations on the datasets instead of storing the data. Once a fault occurs, the lineage can be traversed to recover from the fault. Stark [216] improves in-memory computing on dynamic dataset collections. Trinity [217] is another in-memory distributed platform for iterative computation that partitions the dataset over the main memory of individual machines. Other systems include Storm [218], a distributed and fault-tolerant framework for processing streams in real-time, and Spark-Streaming [203], which uses RDDs for streaming workloads. SocialTrove uses Spark to generate the summary model because the main building block of that algorithm is 2-`means` clustering, which is a data parallel iterative algorithm. Typically many rounds of iterations on many subsets of the data are necessary, along with back and forth communications with the driver machine. The in-memory computation reduces the inter-round overhead and latencies.

Memcached [201] is an in-memory key-value store, often utilized to mask latencies from external data sources by caching results [219]. SocialTrove uses Memcached [201] as the distributed in-memory cache for the data input and the client query proxies, which improves throughput and response time of the queries. Druid [220] is a distributed column-oriented real-time OLAP system that uses a combination of real-time nodes and historical nodes to answer both real-time queries and historical aggregate queries. Compared to Druid, SocialTrove is not limited to structured time-series data. Moreover, Druid emphasizes fast ingestion for real-time queries, whereas SocialTrove provides a flexible summarization service by allowing the users to define a summarization criteria. Duong et al. [221] consider social network topology as a sharding technique to reduce query costs on large social network databases. ApproxHadoop [222] introduces approximation mechanisms into the MapReduce paradigm to reduce runtime. Their approach utilizes statistical sampling theory to aggregate data, where SocialTrove utilizes application defined distance measurements and clustering algorithms to generate summary.

6.7 Summary

In this chapter, we described SocialTrove; an information summarization service for social-sensing. The design of the service is motivated by the advant of an age of information overload, where much data is generated in real-time, and where redundancy is common. SocialTrove delivers data summaries at arbitrary levels of granularity by reducing redundancy through clustering. Evaluation shows that SocialTrove is scalable in serving data summaries because it caches a cluster summary model in memory for a predefined interval, which allows it to provide high throughput, low-latency lookups for real-time social sensing data, without incurring signficiant insertion overheads. It outperforms traditional indexing methods, which incur a heavier latency and suffer from lower throughput. A limitation of the current evaluation is that it tests SocialTrove only in the context of Twitter data summarization. Future work of the authors will focus on exploring the benefits and performance of SocialTrove in summarizing other types of large streaming data.

CHAPTER 7

APOLLO SOCIAL SENSING TOOLKIT

The earlier chapters have presented algorithms to summarize the observable states of the physical world using observations shared in the social network, considering redundancy, influence, bias, and polarization. This chapter integrates the algorithms, and presents a pipeline to build a real-time news feed application using Apollo Social Sensing Toolkit. Apollo Social Sensing Toolkit is a platform to create, execute, and customize social-sensing tasks based on Twitter. The toolkit has several data collection, processing, and presentation modules implemented, and the modules can be interconnected to create a customized pipeline that accomplishes a social sensing task. Application programmers can use existing pipelines, use the existing modules to create a new pipeline, or implement entirely new modules. A social-sensing task generally starts with data collection. The user can specify interests, keywords, and analysis algorithms. A task configuration file is created, and the corresponding pipeline starts. Depending on the configuration, the data is collected through real-time crawlers, or input from underlying storage. Batches of data stream operates through the modules to accomplish a task.

7.1 Architecture

Apollo Social Sensing Toolkit uses a distributed architecture. Each module is executed as a separate process, typically written using Python, Java, or C++. Separate modules perform separate operations, and the modules communicate through RPC mechanism. The pipelines are executed on Apollo Runtime, which is a supervisor process to schedule data crawling or processing tasks, manage existing resources over a cluster, and handle crashes or faults. Because the modules execute as separate processes, crashes are local to particular tasks. Every 5 minutes the collected data are written to persis-

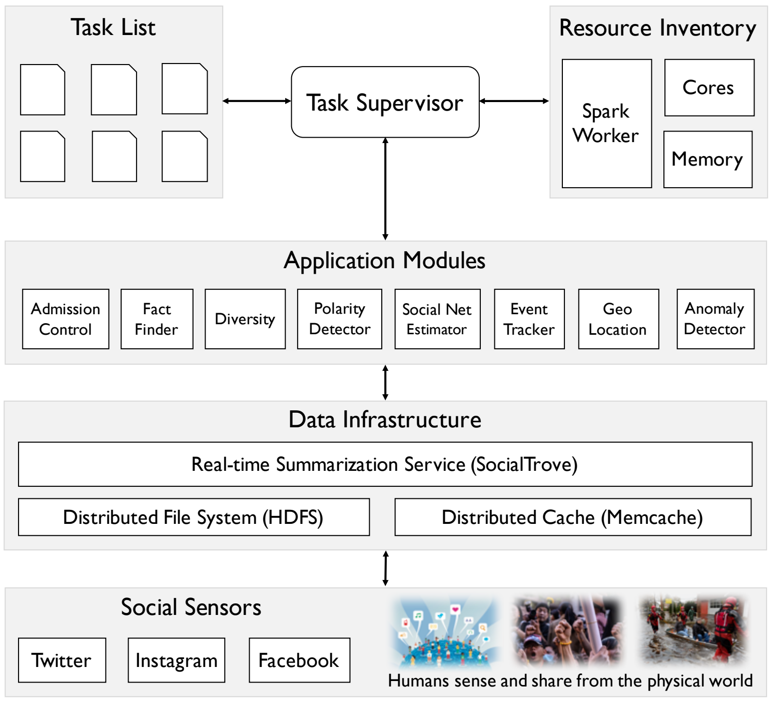


Figure 7.1: Architecture of Apollo Social Sensing Toolkit

tent storage. At every configurable interval (e.g. 30 minutes, 60 minutes, or 24 hours), the accumulated batch is operated through the pipeline.

Figure 7.1 illustrates the system level architecture of Apollo Social Sensing Toolkit.

7.1.1 Social Sensors

Social sensing starts with the 'Social Sensors'. Events of significance (for example, sports, concert, riot, protest, war, earthquake, flood, hurricane, campaign, procession, etc) transpire in the physical world. Humans can observe and share these events through the human social network (for example, when two friends meet) or through the online social media. Twitter, Instagram, Facebook etc. are popular online social networks where people share about these events. Apollo can crawl information through Twitter or Instagram provided API, by making peridic queries using keywords or geographic locations. Apollo can also stream tweets from Twitter Firehose (100% Twitter stream) or Decahose (10% Twitter stream). At this level the sensed data are raw tweets or pictures.

7.1.2 Data Infrastructure

A common characteristic of social sensing systems is that they generate large amounts of redundant data. Summarization services continually cluster the incoming raw objects (for example, tweets or pictures) in a hierarchical fashion. They offer API to obtain a representative summary of a given query (e.g. keywords), at a configurable granularity. Apollo uses SocialTrove [8] as the summarization service. The service allows the task configurations to specify an application-specific distance metric that describes a measure of similarity relevant to this application among data items. Based on that application-specific measure, the service hierarchically clusters incoming data streams in real time, and allows applications to obtain representative samples at arbitrary levels of granularity by returning cluster heads (and member counts) at appropriate levels of the cluster hierarchy. SocialTrove uses Spark to parallelize the summarization workload throughout the cluster, and a distributed file system (for example, HDFS) allows fault-tolerant access to the crawled data throughout the cluster during the summarization phases. At the Data Infrastructure level, Apollo also contains a distributed cache to serve the popular items with low response time.

7.1.3 Application Modules

At the application level, Apollo contains a library of analytics modules. Figure 7.1 shows some example modules. *Admission Control* implements source-selection algorithms to filter objects based on sources [2]. *Fact Finder* implements fact-finder algorithms like Voting, Bayesian, EM-CRB, or EM-Social [3,125]. *Diversity* samples a diversified set of representable summaries [8,175]. In case of polarized scenario, *Polarity Detector* separates the set of inputs into different classes [4]. A pipeline configuration file describes the interconnection between the modules to perform a social sensing task. The application developer can use a defined pipeline, or write a new pipeline to create a new type of social sensing application, or write new modules.

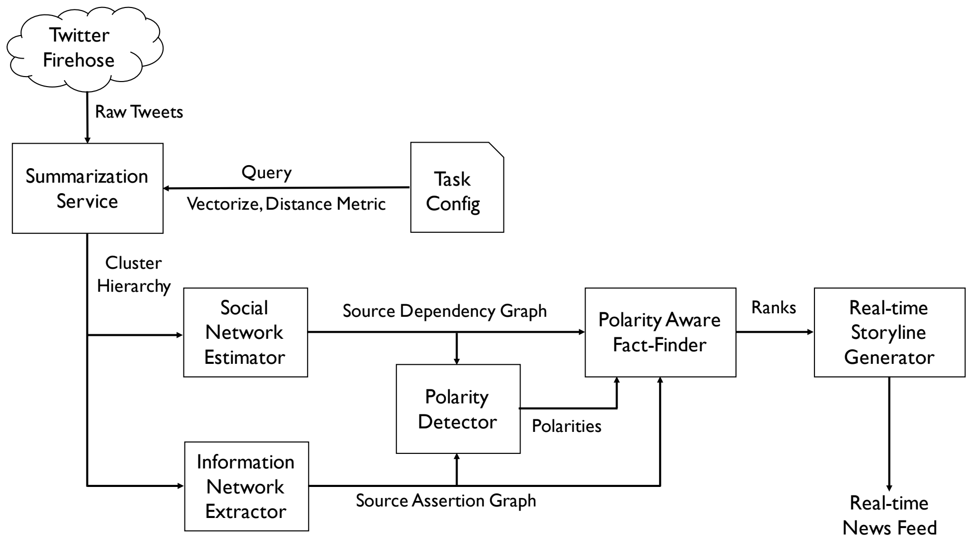


Figure 7.2: Workflow of a real-time news feed generation pipeline

7.1.4 Apollo Runtime

Apollo Runtime consists of a supervisor process to coordinate the social sensing tasks. The Task Supervisor keeps track of the running tasks, and the available resource (memory, cores) over the machine cluster. Based on the available resources, it schedules spark jobs over the cluster, or starts standalone processes for the modules that do not require parallelism. Task Supervisor handles crash recovery. Optionally, it can also prioritize the tasks.

7.2 Real-time News Feed Pipeline

In this section, we build a real-time news feed application using the algorithms presented in the earlier chapters. Based on the input keywords, this application shows a diversified collection of newsworthy tweets that are more likely to be facts from the events happening in the physical world. The workflow of this application is shown in Figure 7.2. Tweets are collected by Apollo crawlers. The task configuration supplies appropriate parsing mechanism (vectorize) and distance functions to SocialTrove, which periodically forms the hierarchical cluster summaries. Based on the keywords specified in the task configuration, the matching objects are collected and passed to a series of application modules described below.

Social Network Estimator estimates a social dependency graph from the cluster summaries using the algorithm from [130]. This graph accounts for uncertain provenance of the sources, who may have tweeted based on their own observations or observations they heard from others. The social network estimator module models the sources and the timestamps in the clusters of tweets as cascades of epidemic propagation, and estimates the latent social dependency network by using a iterative greedy strategy.

Information Network Extractor forms a Source-Assertion graph from the cluster summaries. This is a bipartite graph that relates assertions to the sources. As assertion is formed from the clusters of tweets, by considering the clusters as binary observations.

Polarity Detector In many cases, the events in the physical world are polarized and a community might become divided over an issue, manifesting opposing views. Often, the conflict extends to claims about factual observations. To uncover a less biased (i.e., more neutral) description of events, the polarity detector module uses a matrix factorization approach to separate the set of assertions into groups of different polarities. The polarity detector module is based on the factorization and ensemble based algorithm presented in Chapter 5. Note that only two polarities with $k = 2$, corresponding to `Pro` and `Anti` are used. `Neutral` assertions may get binned with assertions of either polarity. For a news feed service, separating the polarities result in a more accurate reconstruction of the events [4].

Polarity Aware Fact-Finder Using the information network, social dependency network, and the detected polarities, this module estimates the polarity aware credibilities of the sources and the assertions. It uses a maximum likelihood approach to infer the ground truths. Polarity aware fact-finder implements the algorithms described in Chapter 3. The method uses EM-Social algorithm from earlier work [3] as a subroutine. EM-Social is expensive to run when the source-assertion graph is large, and apollo can parallelize the workload in a machine cluster.

Using the ranks generated from the polarity aware fact-finder module, the 'Real-time Storyline Generator' module then prepares the news feed to serve

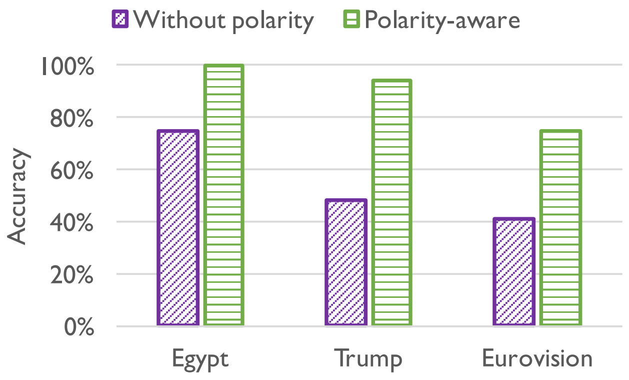


Figure 7.3: Accuracy of the claims unique to particular schemes

to the users. This is a presentation module that keeps track of what information is already visible to the user on the browser window, and shows only new information.

7.3 Evaluation

In this section, we evaluate the quality of distillation and runtime of Apollo. Specifically, we measure the accuracy of results provided by the polarity-aware fact-finder application, and the corresponding runtime. Figure 7.3 considers the claims from the polarity-aware algorithm to EM-Social running without polarity information. For the purpose of comparing the quality, claims that are believed by both algorithms are not included in the evaluation. The plot presents the accuracy of the claims that are exclusive to each algorithm. Three datasets Egypt, Trump, and Eurovision are considered. The datasets have been described in detail in Chapter 4 and Chapter 5. In each cases, the polarity aware algorithm has much better performance. Figure 7.4 shows the runtime of the different components of a polarity-aware fact distillation pipeline. SocialTrove was running on a dataset of 10M tweets with an input size of 50GB. Regardless of the output size, it took around 19 minute for SocialTrove to generate an summarization hierarchy using 16 cores. After the summary tree has been generated, a diverse set of 100 (medium-sized output) or 1000 assertions (large-sized output) are selected for further analysis. The later stages are much faster for the medium-sized output. Note that the timing for the factorization step is dependent on how

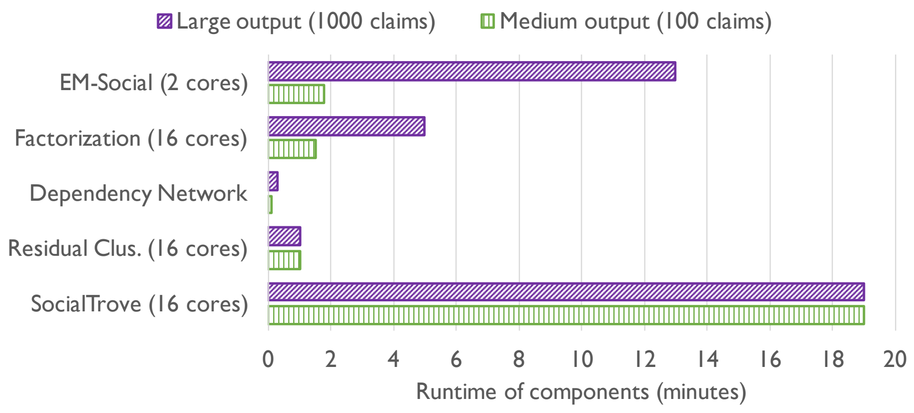


Figure 7.4: Runtime for medium (100 claims) and large (1000 claims) output

polarized the data is, so an average has been plotted. EM-Social runs using 2 cores, each core corresponding to a particular polarity. Note that, it is possible to execute EM-Social in a shared-memory multi-threaded fashion for faster completion.

7.4 Summary

In this section, we have described the architecture of Apollo Social Sensing Toolkit. Apollo uses SocialTrove described in Chapter 6 to generate a summarization hierarchy every interval. From the summarization hierarchy, we build an information network using the most diverse assertions, most popular assertions, or assertions nearest to an event described by keywords. Using the algorithms presented in this dissertation, we have implemented a polarity-aware fact distillation pipeline, that considers the information propagation pattern of the sources and produces distilled facts to present to an analyst.

CHAPTER 8
CONCLUSIONS

The explosive growth in social network content suggests that social sensing might be the future of sensing. In this dissertation, we have presented that exploiting propagation and corroboration properties of the human sources result in a better estimation of the observable states of the physcial world. We have incorporated our algorithms in "Apollo Social Sensing Toolkit", an infrastructure for implementing summarization services on the cloud. We have evaluated our research in the specific context of tweets and workloads collected from Twitter. However, the proposed systems have been carefully designed to separate the content-specific components from the content-indepdent components. In most of the cases, our algorithms work with graph data structures or feature vectors obtained from the input data objects. Adapting our social sensing systems to a different type of data requires writing some content-specific parsers only. To design an appropriate architecture for social sensing, we have observed that scalability is an important concern as the common characteristic of social media is that they generate large amounts of redundant data. Therefore, the back-end needs to take advantage of a machine cluster. SocialTrove is used as the data infrastructure for our proposed architecture for Apollo Social Sensing Toolkit. SocialTrove delivers data summaries at arbitrary levels of granularity by reducing redundancy through clustering. We also observe that social sensing data streams have various types of noise including social influence and polarization. We have proposed matrix factorization and ensemble methods to detect polarization in social networks. We have also proposed polarity aware fact-finder. The summarization services and the fact-finder algorithms have enabled us to build a crowd-sensed news service using Apollo Social Sensing Toolkit. Future research will focus on optimally updating the summary model in SocialTrove from interval to interval, performing credibility estimation in real-time, and scheduling algorithms to jointly improve coverage of different tasks.

REFERENCES

[1] J. Surowiecki, *The Wisdom of Crowds.* Anchor, 2005.

[2] M. Uddin, M. Amin, H. Le, T. Abdelzaher, B. Szymanski, and T. Nguyen, "On diversifying source selection in social sensing," in *9th International Conference on Networked Sensing Systems (INSS)*, 2012.

[3] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. Aggarwal, R. Ganti, X. Wang, P. Mohapatra, B. Szymanski, and H. Le, "Humans as sensors: An estimation theoretic perspective," in *ACM/IEEE Conf. on Information Processing in Sensor Networks*, 2014.

[4] M. T. A. Amin, T. Abdelzaher, D. Wang, and B. Szymanski, "Crowd-sensing with polarized sources," in *Proc. 2014 IEEE Intl. Conference on Distributed Computing in Sensor Systems*, 2014, pp. 67–74.

[5] S. E. Kase, E. K. Bowman, M. T. Amin, and T. Abdelzaher, "Exploiting social media for army operations: Syrian crisis use case," in *Proc. SPIE Defense, Security, and Sensing*, 2014.

[6] M. T. A. Amin, T. Abdelzaher, and L. Kaplan, "On evaluating polarization models in social networks," 2017, submitted.

[7] M. T. A. Amin, C. Aggarwal, S. Yao, T. Abdelzaher, and L. Kaplan, "Unveiling polarization in social networks: A matrix factorization approach," in *Proc. IEEE International Conference on Computer Communications (INFOCOM 2017)*, May 2017.

[8] M. T. A. Amin, S. Li, M. R. Rahman, P. T. Seetharamu, S. Wang, T. Abdelzaher, I. Gupta, M. Srivatsa, R. Ganti, R. Ahmed, and H. Le, "SocialTrove: A self-summarizing storage service for social sensing," in *International Conference on Autonomic Computing (ICAC'15).* IEEE, July 2015, pp. 41–50.

[9] "Apollo-Toward Fact-finding for Social Sensing," Feb 2017. [Online]. Available: http://apollofactfinder.net/

[10] M. Y. S. Uddin, M. T. A. Amin, T. Abdelzaher, A. Iyengar, and R. Govindan, "Photonet+: Outlier-resilient coverage maximization in visual sensing applications," in *ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*, 2012.

[11] Oct 2014. [Online]. Available: https://blog.twitter.com/2013/new-tweets-per-second-record-and-how/

[12] [Online]. Available: http://greendatacenters.web.engr.illinois.edu/

[13] D. Wang, T. Abdelzaher, and L. Kaplan, *Social sensing: building reliable systems on unreliable data.* Morgan Kaufmann, 2015.

[14] K. Sha, A. Striegel, and M. Song, "Advances in computer communications and networks," 2016.

[15] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, ser. IPSN '12, 2012, pp. 233–244.

[16] H. Le, D. Wang, H. Ahmadi, M. Y. S. Uddin, Y. H. Ko, T. Abdelzaher, O. Fatemieh, J. Pasternack, D. Roth, J. Han, H. Wang, L. Kaplan, B. Szymanski, S. Adali, C. Aggarwal, and R. Ganti, "Apollo: A data distillation service for social sensing," University of Illinois Urbana-Champaign, Tech. Rep., 2012.

[17] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric sensing," *Philosophical Transactions of the Royal Society, A*, vol. 370, no. 1958, pp. 176–197, 2012.

[18] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "CarTel: a distributed mobile sensor computing system," in *Proc of SenSys*, 2006.

[19] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "The bikenet mobile sensing system for cyclist experience mapping," in *Prof of Sensys*, 2007.

[20] M. Y. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, and T. Huang, "Photonet: a similarity-aware picture delivery service for situation awareness," in *Proc. of RTSS*, 2011.

[21] J.-H. Huang, S. Amjad, and S. Mishra, "Cenwits: a sensor-based loosely coupled search and rescue system using witnesses," in *Proc. of SenSys*, 2005, pp. 180–191.

[22] Sense Networks, "Cab sense," http://www.cabsense.com/.

[23] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell, “Urban sensing systems: Opportunistic or participatory?” in *9th workshop on Mobile computing systems and applications*, 2008.

[24] D. Cuff, M. Hansen, and J. Kang, “Urban sensing: out of the woods,” *Communications of the ACM*, vol. 51, no. 3, pp. 24–33, 2008.

[25] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, “Image browsing, processing, and clustering for participatory sensing: Lessons from a dietsense prototype,” in *Proc. of EmNets*, 2007.

[26] S. Nath, “Ace: Exploiting correlation for energy-efficient and continuous context sensing,” in *Proceedings of the tenth international conference on Mobile systems, applications, and services (MobiSys’12)*, 2012.

[27] I. Boutsis and V. Kalogeraki, “Privacy preservation for participatory sensing data,” in *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on.* IEEE, 2013, pp. 103–113.

[28] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, “Sociablesense: exploring the trade-offs of adaptive sampling and computation offloading for social sensing,” in *Proceedings of the 17th annual international conference on Mobile computing and networking.* ACM, 2011, pp. 73–84.

[29] S. Brin and L. Page, “The anatomy of a large-scale hypertextual Web search engine,” *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[30] J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[31] R. Lempel and S. Moran, “Salsa: the stochastic approach for link-structure analysis,” *ACM Transactions on Information Systems (TOIS)*, vol. 19, no. 2, pp. 131–160, 2001.

[32] D. Achlioptas, A. Fiat, A. R. Karlin, and F. McSherry, “Web search via hub synthesis,” in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on.* IEEE, 2001, pp. 500–509.

[33] L. Berti-Equille, A. D. Sarma, X. Dong, A. Marian, and D. Srivastava, “Sailing the information ocean with awareness of currents: Discovery and application of source dependence,” in *CIDR*, 2009.

[34] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti, “Probabilistic models to reconcile complex data from inaccurate data sources,” in *CAiSE*, 2010, pp. 83–97.

[35] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796–808, June 2008.

[36] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, "Corroborating information from disagreeing views," in *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 2010, pp. 131–140.

[37] X. L. Dong, L. Berti-Equille, and D. Srivastava, "Integrating conflicting data: The role of source dependence," *PVLDB*, vol. 2, no. 1, pp. 550–561, 2009.

[38] X. Dong, L. Berti-Equille, Y. Hu, and D. Srivastava, "Global detection of complex copying relationships between sources," *PVLDB*, vol. 3, no. 1, pp. 1358–1369, 2010.

[39] J. Pasternack and D. Roth, "Knowing what to believe (when you already know something)," in *International Conference on Computational Linguistics (COLING)*, 2010.

[40] J. Pasternack and D. Roth, "Generalized fact-finding (poster paper)," in *World Wide Web Conference (WWW'11)*, 2011.

[41] M. Gupta, Y. Sun, and J. Han, "Trust analysis with clustering," in *WWW*, ser. WWW '11. ACM, 2011.

[42] D. Wang, T. Abdelzaher, L. Kaplan, and C. C. Aggarwal, "On quantifying the accuracy of maximum likelihood estimation of participant reliability in social sensing," in *8th International Workshop on Data Management for Sensor Networks (DMSN)*, 2011.

[43] S. Wang, L. Su, S. Li, S. Hu, T. Amin, H. Wang, S. Yao, L. Kaplan, and T. Abdelzaher, "Scalable social sensing of interdependent phenomena," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks (IPSN)*, 2015, pp. 202–213.

[44] S. Yao, M. T. Amin, L. Su, S. Hu, S. Li, S. Wang, Y. Zhao, T. Abdelzaher, L. Kaplan, C. Aggarwal, and A. Yener, "Recursive ground truth estimator for social data streams," in *Proc. IPSN*, 2016.

[45] G.-J. Qi, C. C. Aggarwal, J. Han, and T. Huang, "Mining collective intelligence in diverse groups," in *Proc. International Conference on World Wide Web*, 2013, pp. 1041–1052.

[46] V. V. Vydiswaran, C. Zhai, and D. Roth, "Content-driven trust propagation framework," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 974–982.

[47] J. Lehmann, D. Gerber, M. Morsey, and A.-C. Ngonga Ngomo, "Defacto - deep fact validation," in *Proc. 11th International Semantic Web Conference*, 2012, pp. 312–327.

[48] D. Yu, H. Huang, T. Cassidy, H. Ji, C. Wang, S. Zhi, J. Han, C. R. Voss, and M. Magdon-Ismail, "The wisdom of minority: Unsupervised slot filling validation based on multi-dimensional truth-finding." in *Proc. Int. Conf. on Computational Linguistics (COLING'14)*, 2014.

[49] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, "A confidence-aware approach for truth discovery on long-tail data," *Proc. VLDB Endow.*, 2014.

[50] Y. Cao, W. Fan, and W. Yu, "Determining the relative accuracy of attributes," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 565–576.

[51] M. Sensoy, A. Fokoue, J. Z. Pan, T. J. Norman, Y. Tang, N. Oren, and K. Sycara, "Reasoning about uncertain information and conflict resolution through trust revision," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, 2013, pp. 837–844.

[52] A. Pal, V. Rastogi, A. Machanavajjhala, and P. Bohannon, "Information integration over time in unreliable and uncertain environments," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 789–798.

[53] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han, "On the discovery of evolving truth," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[54] S. Zhi, B. Zhao, W. Tong, J. Gao, D. Yu, H. Ji, and J. Han, "Modeling truth existence in truth discovery," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1543–1552.

[55] M. Wu and A. Marian, "A framework for corroborating answers from multiple web sources," *Information Systems*, vol. 36, no. 2, pp. 431–449, 2011.

[56] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava, "Truth finding on the deep web: is the problem solved?" in *Proceedings of the 39th international conference on Very Large Data Bases*, 2013.

[57] A. Marian and M. Wu, "Corroborating information from web sources," *Data Engineering*, p. 11, 2011.

[58] X. L. Dong, L. Berti-Equille, and D. Srivastava, "Truth discovery and copying detection in a dynamic world," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 562–573, Aug. 2009.

[59] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti, "Probabilistic models to reconcile complex data from inaccurate data sources," in *International Conference on Advanced Information Systems Engineering.* Springer, 2010, pp. 83–97.

[60] X. Liu, X. L. Dong, B. C. Ooi, and D. Srivastava, "Online data fusion," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 932–943, 2011.

[61] A. D. Sarma, X. L. Dong, and A. Halevy, "Data integration with dependent sources," in *Proceedings of the 14th International Conference on Extending Database Technology*, 2011, pp. 401–412.

[62] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han, "A bayesian approach to discovering truth from conflicting sources for data integration," *Proc. VLDB Endow.*, vol. 5, no. 6, pp. 550–561, Feb. 2012.

[63] B. Zhao and J. Han, "A probabilistic model for estimating real-valued truth from conflicting sources," *Proc. of QDB*, 2012.

[64] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 1187–1198.

[65] J. Pasternack and D. Roth, "Latent credibility analysis," in *Proc. International Conference on World Wide Web*, 2013, pp. 1009–1020.

[66] X. L. Dong and D. Srivastava, "Compact explanation of data fusion decisions," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013.

[67] R. Pochampally, A. Das Sarma, X. L. Dong, A. Meliou, and D. Srivastava, "Fusing data with correlations," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 2014, pp. 433–444.

[68] X. Li, X. L. Dong, K. B. Lyons, W. Meng, and D. Srivastava, "Scaling up copy detection," in *2015 IEEE 31st International Conference on Data Engineering*, April 2015, pp. 89–100.

[69] J. P. Callan, Z. Lu, and W. B. Croft, "Searching distributed collections with inference networks," in *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '95, New York, NY, USA, 1995, pp. 21–28.

[70] L. Gravano, H. Garcia-Molina, and A. Tomasic, “Gloss: text-source discovery over the internet,” *ACM Transactions on Information Systems (TOIS)*, vol. 24, pp. 229–264, 1999.

[71] B. Yuwono and D. Lee, “Server ranking for distributed text retrieval systems on the internet,” in *Proc of Database Systems for Advanced Applications*, 1997, pp. 41 – 49.

[72] R. Balakrishnan and S. Kambhampati, “SourceRank: Relevance and trust assessment for deep web sources based on inter-source agreement,” in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 227–236.

[73] F. Abbaci, J. Savoy, and M. Beigbeder, “A methodology for collection selection in heterogeneous contexts,” in *Proc of Information Technology: Coding and Computing (ITCC)*, 2002.

[74] L. Si, R. Jin, J. Callan, and P. Ogilvie, “Language modeling framework for resource selection and results merging,” in *Proc of Information and Knowledge Management (CIKM)*, 2002.

[75] D. Aksoy, “Information source selection for resource constrained environments,” *SIGMOD Rec.*, vol. 34, no. 4, pp. 15–20, 2005.

[76] H. Dai, F. Zhu, E. P. Lim, and H. Pang, “Detecting anomalies in bipartite graphs with mutual dependency principles,” in *2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 171–180.

[77] M. Gupta, P. Zhao, and J. Han, “Evaluating event credibility on twitter,” in *Proceedings of the 2012 SIAM International Conference on Data Mining.* SIAM, 2012, pp. 153–164.

[78] Z. Xu, Y. Liu, J. Xuan, H. Chen, and L. Mei, “Crowdsourcing based social media data analysis of urban emergency events,” *Multimedia Tools and Applications*, pp. 1–18, 2015.

[79] C. Ye, H. Wang, H. Gao, J. Li, and H. Xie, *Truth Discovery Based on Crowdsourcing*, 2014, pp. 453–458.

[80] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng, “Truth discovery on crowd sensing of correlated entities,” in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SENSYS)*, 2015, pp. 169–182.

[81] S. Gu, C. Pan, H. Liu, S. Li, S. Hu, L. Su, S. Wang, D. Wang, T. Amin, R. Govindan et al., “Data extrapolation in social sensing for disaster response,” in *Distributed Computing in Sensor Systems (DCOSS), 2014 IEEE International Conference on*, 2014, pp. 119–126.

[82] P. G. Ipeirotis, F. Provost, and J. Wang, "Quality management on amazon mechanical turk," in *Proceedings of the ACM SIGKDD Workshop on Human Computation*, 2010, pp. 64–67.

[83] Z. He, J. Cao, and X. Liu, "High quality participant recruitment in vehicle-based crowdsourcing using predictable mobility," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, 2015, pp. 2542–2550.

[84] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas, "Crowdsourcing for multiple-choice question answering." in *In AAAI*, 2014, pp. 2946–2953.

[85] L. Su, Q. Li, S. Hu, S. Wang, J. Gao, H. Liu, T. F. Abdelzaher, J. Han, X. Liu, Y. Gao, and L. Kaplan, "Generalized decision aggregation in distributed sensing systems," in *2014 IEEE Real-Time Systems Symposium*, 2014, pp. 1–10.

[86] X. Wang, Q. Z. Sheng, X. S. Fang, X. Li, X. Xu, and L. Yao, "Approximate truth discovery via problem scale reduction," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, 2015.

[87] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han, "Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[88] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren, "Cloud-enabled privacy-preserving truth discovery in crowd sensing systems," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SENSYS)*, 2015.

[89] R. W. Ouyang, L. M. Kaplan, A. Toniolo, M. Srivastava, and T. J. Norman, "Aggregating crowdsourced quantitative claims: Additive and multiplicative models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 7, pp. 1621–1634, 2016.

[90] R. W. Ouyang, M. Srivastava, A. Toniolo, and T. J. Norman, "Truth discovery in crowdsourced detection of spatial events," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 1047–1060, 2016.

[91] X. Wang, Q. Z. Sheng, L. Yao, X. Li, X. S. Fang, X. Xu, and B. Benatallah, "Truth discovery via exploiting implications from multi-source data," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*, 2016.

[92] Y. Li, N. Du, C. Liu, Y. Xie, W. Fan, Q. Li, J. Gao, and H. Sun, "Reliable medical diagnosis from crowdsourcing: Discover trustworthy answers from non-experts," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, 2017.

[93] P. Giridhar, S. Wang, T. F. Abdelzaher, J. George, L. Kaplan, and R. Ganti, "Joint localization of events and sources in social networks," in *Distributed Computing in Sensor Systems (DCOSS), 2015 International Conference on.* IEEE, 2015, pp. 179–188.

[94] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "Twitinfo: aggregating and visualizing microblogs for event exploration," in *Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, 2011, pp. 227–236.

[95] P. Giridhar, M. Amin, T. Abdelzaher, L. Kaplan, J. George, and R. Ganti, "Clarisense: Clarifying sensor anomalies using social network feeds," in *Pervasive Computing and Communications Workshops, 2014 IEEE International Conference on*, March 2014, pp. 395–400.

[96] P. Giridhar, M. T. Amin, T. Abdelzaher, D. Wang, L. Kaplan, J. George, and R. Ganti, "Clarisense+: An enhanced traffic anomaly explanation service using social network feeds," *Pervasive and Mobile Computing*, vol. 33, pp. 140–155, 2016.

[97] J. Nichols, J. Mahmud, and C. Drews, "Summarizing sporting events using twitter," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces.* ACM, 2012, pp. 189–198.

[98] S. Mukherjee, G. Weikum, and C. Danescu-Niculescu-Mizil, "People on drugs: credibility of user statements in health communities," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2014, pp. 65–74.

[99] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web.* ACM, 2010, pp. 851–860.

[100] P. Bogdanov, M. Busch, J. Moehlis, A. K. Singh, and B. K. Szymanski, "The social media genome: Modeling individual topic-specific behavior in social media," in *Proc. 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2013.

[101] S. Sikdar, B. Kang, J. O'Donovan, T. Hllerer, and S. Adalı, "Understanding information credibility on twitter," in *Proc. SocialCom*, 2013.

[102] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on twitter," in *Proc. WWW*, NY, USA, 2011, pp. 675–684.

[103] J. Lehmann, B. Gonçalves, J. J. Ramasco, and C. Cattuto, "Dynamical classes of collective attention in twitter," in *Proceedings of the 21st International Conference on World Wide Web*, 2012, pp. 251–260.

[104] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes, "Correlating financial time series with micro-blogging activity," in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, 2012, pp. 513–522.

[105] A. Gupta, H. Lamba, P. Kumaraguru, and A. Joshi, "Faking sandy: Characterizing and identifying fake images on twitter during hurricane sandy," in *Proceedings of the 22Nd International Conference on World Wide Web*, pp. 729–736.

[106] M. Imran, C. Castillo, F. Diaz, and S. Vieweg, "Processing social media messages in mass emergency: A survey," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 67:1–67:38, June 2015.

[107] D. Wang, T. Abdelzaher, L. Kaplan, and C. C. Aggarwal, "Recursive fact-finding: A streaming approach to truth estimation in crowdsourcing applications," in *33rd International Conference on Distributed Computing Systems (ICDCS)*, 2013.

[108] D. Wang, T. Abdelzaher, L. Kaplan, R. Ganti, S. Hu, and H. Liu, "Exploitation of physical constraints for reliable social sensing," in *IEEE Real-Time Systems Symposium*, 2013, pp. 212–223.

[109] D. Wang and C. Huang, "Confidence-aware truth estimation in social sensing applications," in *12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2015, pp. 336–344.

[110] C. Huang and D. Wang, "Link weight based truth discovery in social sensing," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks (IPSN)*, 2015, pp. 326–327.

[111] R. W. Ouyang, L. Kaplan, P. Martin, A. Toniolo, M. Srivastava, and T. J. Norman, "Debiasing crowdsourced quantitative characteristics in local businesses and services," in *Proceedings of the 14th International Conference on Information Processing in Sensor Networks (IPSN)*, 2015.

[112] C. Huang and D. Wang, "Unsupervised interesting places discovery in location-based social sensing," in *Distributed Computing in Sensor Systems (DCOSS), 2016 International Conference on*, 2016, pp. 67–74.

[113] C. Huang and D. Wang, "Exploiting spatial-temporal-social constraints for localness inference using online social media," in *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, 2016, pp. 287–294.

[114] D. Wang, J. Marshall, and C. Huang, "Theme-relevant truth discovery on twitter: An estimation theoretic approach." in *ICWSM*, 2016, pp. 408–416.

[115] C. Huang and D. Wang, "Topic-aware social sensing with arbitrary source dependency graphs," in *Proceedings of the 15th International Conference on Information Processing in Sensor Networks (IPSN)*, 2016.

[116] G. Wang, S. Xie, B. Liu, and P. S. Yu, "Identify online store review spammers via social review graph," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 4, pp. 61:1–61:21, Sep. 2012.

[117] N. Agarwal, H. Liu, L. Tang, and S. Y. Philip, "Modeling blogger influence in a community," *Social Network Analysis and Mining*, vol. 2, no. 2, pp. 139–162, 2012.

[118] A. Etuk, T. J. Norman, M. ensoy, C. Bisdikian, and M. Srivatsa, "Tidy: A trust-based approach to information fusion through diversity," in *Proceedings of the 16th International Conference on Information Fusion*, 2013, pp. 1188–1195.

[119] F. Nel, L. M.-J., P. Capet, and T. Dellavallade, "Rumor detection and monitoring in open source intelligence: Understanding publishing behaviors as a prerequisite," in *Proc. Terrorism and New Media Conference*, 2010.

[120] D. Shah and T. Zaman, "Rumors in a network: Who's the culprit?" *IEEE Transactions on Information Theory*, vol. 57, pp. 5163–5181, 2011.

[121] F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan, "Epidemiological modeling of news and rumors on twitter," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis.* ACM, 2013, p. 8.

[122] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, "Sybilguard: Defending against sybil attacks via social networks," in *ACM SIGCOMM*, 2006, pp. 267–278.

[123] L. Shi, S. Yu, W. Lou, and Y. T. Hou, "Sybilshield: An agent-aided social network-based sybil defense among multiple communities," in *INFOCOM, 2013 Proceedings IEEE.* IEEE, 2013, pp. 1034–1042.

[124] D. Wang, M. Amin, T. Abdelzaher, D. Roth, C. Voss, L. Kaplan, S. Tratz, J. Laoudi, and D. Briesch, "Provenance-assisted classification in social networks," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 8, no. 4, pp. 624–637, 2014.

[125] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *ACM/IEEE Conf. on Information Processing in Sensor Networks*, 2012.

[126] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric sensing," *Philosophical Transactions of the Royal Society, Series A*, vol. 370, pp. 176–197, 2012.

[127] S. Sikdar, S. Adal, M. Amin, T. Abdelzaher, K. Chan, J.-H. Cho, B. Kang, and J. O'Donovan, "Finding true and credible information on twitter," in *17th International Conference on Information Fusion*, 2014.

[128] D. Wang, T. Abdelzaher, L. Kaplan, and R. Ganti, "Exploitation of physical constraints for reliable social sensing," in *Proc. Real-Time Systems Symposium (RTSS)*, 2013.

[129] D. Wang, L. Kaplan, T. Abdelzaher, and C. C. Aggarwal, "On scalability and robustness limitations of real and asymptotic confidence bounds in social sensing," in *9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2012.

[130] P. Netrapalli and S. Sanghavi, "Learning the graph of epidemic cascades," in *Proc. 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2012.

[131] D. M. Romero, B. Meeder, and J. Kleinberg, "Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter," in *Proc. 20th International Conference on World Wide Web (WWW)*, 2011.

[132] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proc. 9th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, 2003.

[133] N. Friedkin, *A Structural Theory of Social Influence.* Cambridge University Press, 2006.

[134] M. E. J. Newman, "The structure and function of complex networks," *SIAM REVIEW*, vol. 45, pp. 167–256, 2003.

[135] S. A. Myers and J. Leskovec., "On the convexity of latent social network inference," in *Proc. Neural Information Processing Systems (NIPS)*, 2010.

[136] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," in *Proc. 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010.

[137] G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Online community detection in social sensing," in *Proc. 6th ACM International Conference on Web Search and Data Mining (WSDM)*, 2013, pp. 617–626.

[138] G.-J. Qi, C. Aggarwal, and T. Huang, "Community detection with edge content in social media networks," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, April 2012, pp. 534–545.

[139] C. X. Lin, Q. Mei, Y. Jiang, J. Han, and S. Qi, "Inferring the diffusion and evolution of topics in social communities," in *Proc. ACM SIGKDD Workshop on Social Network Mining and Analysis (SNAKDD)*, 2011.

[140] A. Pal and S. Counts, "Identifying topical authorities in microblogs," in *Proc. 4th ACM International Conference on Web Search and Data Mining (WSDM)*, 2011.

[141] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts, "Everyone's an influencer: Quantifying influence on twitter," in *Proc. 4th ACM Intl. Conference on Web Search and Data Mining (WSDM)*, 2011.

[142] M. J. Franklin, B. Trushkowsky, P. Sarkar, and T. Kraska, "Crowd-sourced enumeration queries," in *Proc. 2013 IEEE International Conference on Data Engineering (ICDE)*, 2013.

[143] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *ACM International Conference on Management of data (SIGMOD)*, 2011.

[144] J. Kulshrestha, M. Zafar, L. Noboa, K. Gummadi, and S. Ghosh, "Characterizing information diets of social media users," in *International AAAI Conference on Web and Social Media*, 2015.

[145] A. Hermida, F. Fletcher, D. Korrell, and D. Logan, "Your friend as editor: the shift to the personalized social news stream," in *Future of Journalism Conference*, 2011.

[146] S. Flaxman, S. Goel, and J. Rao, "Filter bubbles, echo chambers, and online news consumption," *Public Opinion Quarterly*, p. nfw006, 2016.

[147] C. Grevet, L. G. Terveen, and E. Gilbert, "Managing political differences in social media," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing.* ACM, 2014, pp. 1400–1408.

[148] P. Barberá, J. T. Jost, J. Nagler, J. A. Tucker, and R. Bonneau, "Tweeting from left to right is online political communication more than an echo chamber?" *Psychological science*, 2015.

[149] A. J. Morales, J. Borondo, J. C. Losada, and R. M. Benito, "Measuring political polarization: Twitter shows the two sides of venezuela," *Chaos*, vol. 25, no. 3, 2015.

[150] K. Garimella, G. De Francisci Morales, A. Gionis, and M. Mathioudakis, "Quantifying controversy in social media," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ser. WSDM '16, 2016.

[151] S. K. Heather Roy, Elizabeth Bowman and T. Abdelzaher, "Investigating social bias in social media information transmission," in *21st International Command and Control Research and Technology Symposium*, 2016.

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

[153] E. Colleoni, A. Rozza, and A. Arvidsson, "Echo chamber or public sphere? predicting political orientation and measuring political homophily in twitter using big data," *Journal of Communication*, vol. 64, no. 2, pp. 317–332, 2014.

[154] J. An, D. Quercia, and J. Crowcroft, "Partisan sharing: facebook evidence and societal consequences," in *Proceedings of the second ACM conference on Online social networks.* ACM, 2014, pp. 13–24.

[155] E. Bakshy, S. Messing, and L. A. Adamic, "Exposure to ideologically diverse news and opinion on facebook," *Science*, vol. 348, no. 6239, pp. 1130–1132, 2015.

[156] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *J. Artif. Int. Res.*, vol. 50, no. 1, May 2014.

[157] C. Levallois, "Umigon: sentiment analysis on tweets based on terms lists and heuristics," in *Proc. 7th International Workshop on Semantic Evaluation*, June 2013.

[158] Y. Choi, Y. Jung, and S.-H. Myaeng, “Identifying controversial issues and their sub-topics in news articles,” in *Pacific-Asia Workshop on Intelligence and Security Informatics.* Springer, 2010, pp. 140–153.

[159] Y. Mejova, A. X. Zhang, N. Diakopoulos, and C. Castillo, “Controversy and sentiment in online news,” *arXiv preprint arXiv:1409.8152*, 2014.

[160] J. S. Morgan, C. Lampe, and M. Z. Shafiq, “Is news sharing on twitter ideologically biased?” in *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, ser. CSCW ’13, 2013.

[161] Wikipedia, “United kingdom european union membership referendum, 2016,” Jan 2017. [Online]. Available: https://en.wikipedia.org/w/index.php?title=United_Kingdom_European_Union_membership_referendum,_2016&oldid=760173015

[162] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[163] “Twitter Search API,” Jan 2017. [Online]. Available: https://dev.twitter.com/rest/public/search

[164] “Introducing json,” http://www.json.org.

[165] L. A. Adamic and N. Glance, “The political blogosphere and the 2004 u.s. election: Divided they blog,” in *Proceedings of the 3rd International Workshop on Link Discovery*, 2005.

[166] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe, “Predicting elections with twitter: What 140 characters reveal about political sentiment.” *ICWSM*, vol. 10, pp. 178–185, 2010.

[167] A. Gruzd and J. Roy, “Investigating political polarization on twitter: A canadian perspective,” *Policy & Internet*, vol. 6, no. 1, pp. 28–45, 2014.

[168] D. Garcia, A. Abisheva, S. Schweighofer, U. Serdült, and F. Schweitzer, “Ideological and temporal components of network polarization in online political participatory media,” *Policy & Internet*, vol. 7, no. 1, pp. 46–79, 2015.

[169] J. K. Lee, J. Choi, C. Kim, and Y. Kim, “Social media, network heterogeneity, and opinion polarization,” *Journal of Communication*, vol. 64, no. 4, 2014.

[170] P. Barberá, "How social media reduces mass political polarization. evidence from germany, spain, and the u.s." American Political Science Association annual meeting, 2015. [Online]. Available: http://pablobarbera.com/static/barbera_polarization_APSA.pdf

[171] M. W. Macy, J. A. Kitts, A. Flache, and S. Benard, "Polarization in dynamic networks: A hopfield model of emergent structure," *Dynamic social network modeling and analysis*, pp. 162–173, 2003.

[172] P. H. C. Guerra, W. Meira Jr, C. Cardie, and R. Kleinberg, "A measure of polarization on social media networks based on community boundaries." in *Proc. International AAAI Conference on Weblogs and Social Media*, 2013.

[173] L. Akoglu, "Quantifying political polarity based on bipartite opinion networks," in *International AAAI Conference on Web and Social Media*, 2014.

[174] "Collect great content to share," Jan 2017. [Online]. Available: http://paper.li

[175] J. Lee, A. Kapoor, M. T. A. Amin, Z. Wang, Z. Zhang, R. Goyal, and T. Abdelzaher, "Infomax: An information maximizing transport layer protocol for named data networks," in *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, 2015.

[176] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631, 2013, p. 1642.

[177] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *City*, vol. 1, no. 2, p. 1, 2007.

[178] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *Proceedings of Workshop on Text Mining, 6th ACM SIGKDD International Conference on Data Mining*, 2000.

[179] "Sentiment140 - A Twitter Sentiment Analysis Tool," Jul 2016. [Online]. Available: http://http://www.sentiment140.com//

[180] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[181] R. E. Schapire and Y. Freund, *Boosting: Foundations and algorithms.* MIT press, 2012.

[182] Z.-H. Zhou, *Ensemble methods: foundations and algorithms.* CRC press, 2012.

[183] D. Estrin, "Participatory sensing: applications and architecture [internet predictions]," *Internet Computing, IEEE*, vol. 14, no. 1, 2010.

[184] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "Peir, the personal environmental impact report, as a platform for participatory sensing systems research," in *Proceedings of the 7th international conference on Mobile systems, applications, and services.* ACM, 2009, pp. 55–68.

[185] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "GreenGPS: A participatory sensing fuel-efficient maps application," in *8th Intl. Conf. on Mobile Systems, Applications, and Services*, 2010.

[186] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing.* Springer, 2010, pp. 138–155.

[187] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn, "The rise of people-centric sensing," *Internet Computing, IEEE*, vol. 12, no. 4, pp. 12–21, 2008.

[188] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos, "Anonysense: A system for anonymous opportunistic sensing," *Pervasive and Mobile Computing*, vol. 7, no. 1, pp. 16–30, 2011.

[189] M. F. Goodchild, "Citizens as sensors: the world of volunteered geography," *GeoJournal*, vol. 69, no. 4, pp. 211–221, 2007.

[190] A. H. Lipkus, "A proof of the triangle inequality for the tanimoto distance," *Journal of Mathematical Chemistry*, vol. 26, 1999.

[191] M. Shindler, A. Wong, and A. Meyerson, "Fast and accurate k-means for large datasets," in *Neural Information Processing Systems*, 2011.

[192] K. Clarkson, "Nearest-neighbor searching and metric space dimensions," in *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, 2006, pp. 15–59.

[193] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *23rd International Conference on Very Large Data Bases*, 1997, pp. 426–435.

[194] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[195] C. C. Aggarwal and C. K. Reddy, *Data Clustering: Algorithms and Applications*, 1st ed. Chapman & Hall/CRC, 2013.

[196] L. Cayton, "Fast nearest neighbor retrieval for bregman divergences," in *Intl. Conference on Machine Learning*, 2008, pp. 112–119.

[197] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[198] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," in *6th Annual ACM Symposium on Principles of Distributed Computing*, 1987, pp. 1–12.

[199] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *1st Intl. Conf. on Cloud Computing*, 2009.

[200] "Apache thrift," Jan 2015. [Online]. Available: https://thrift.apache.org/

[201] "Memcached," Jan 2015. [Online]. Available: http://memcached.org

[202] "Hadoop," Jan 2015. [Online]. Available: http://hadoop.apache.org/

[203] "Spark," Sep 2014. [Online]. Available: http://spark.apache.org/

[204] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," in *2nd USENIX conference on Hot topics in cloud computing*, 2010.

[205] "Apache mesos," Jan 2015. [Online]. Available: http://mesos.apache.org/

[206] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *9th USENIX Networked Systems Design and Implementation*, 2012.

[207] X. Yu, Q. Fu, L. Zhang, W. Zhang, V. Li, and L. Guibas, "Cabsense: creating high-resolution urban pollution maps with taxi fleets," *ACM MobiSys, Taipei*, 2013.

[208] Y. Jiang, X. Xu, P. Terlecky, T. Abdelzaher, A. Bar-Noy, and R. Govindan, "Mediascope: Selective on-demand media retrieval from mobile devices," in *ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN '13, 2013, pp. 289–300.

[209] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-`means++`," *Proc. VLDB Endow.*, vol. 5, no. 7, 2012.

[210] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey, "Scatter/gather: A cluster-based approach to browsing large document collections," in *Research and Development in Info. Retrieval*, 1992.

[211] N. Ailon, R. Jaiswal, and C. Monteleoni, "Streaming k-`means` approximation," in *Neural Information Processing Systems*, 2009.

[212] A. Guerrieri and A. Montresor, "DS-`means`: Distributed data stream clustering," in *Intl. Conf. on Parallel Processing*, 2012.

[213] S. Bandyopadhyay, C. Giannella, U. Maulik, H. Kargupta, K. Liu, and S. Datta, "Clustering distributed data streams in peer-to-peer environments," *Inf. Sci.*, vol. 176, no. 14, pp. 1952–1985, July 2006.

[214] D. Pelleg and A. W. Moore, "X-`means`: Extending k-`means` with efficient estimation of the number of clusters," in *International Conference on Machine Learning*, 2000, pp. 727–734.

[215] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, 2008.

[216] S. Li, M. T. Amin, R. Ganti, M. Srivatsa, S. Hu, Y. Zhao, and T. Abdelzaher, "Stark: Optimizing in-memory computing for dynamic dataset collections," in *In Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS'17)*, 2017.

[217] B. Shao, H. Wang, and Y. Li, "Trinity: A distributed graph engine on a memory cloud," in *Intl. Conference on Management of Data*, 2013.

[218] "Apache Storm," Sep 2014. [Online]. Available: http://storm.apache.org/

[219] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, "Scaling memcache at facebook," in *Networked Systems Design and Implementation*, 2013, pp. 385–398.

[220] F. Yang, E. Tschetter, X. Léauté, N. Ray, G. Merlino, and D. Ganguli, "Druid: A real-time analytical data store," in *Proceedings of the 2014 ACM SIGMOD Intl. Conf. on Management of Data*, 2014, pp. 157–168.

[221] Q. Duong, S. Goel, J. Hofman, and S. Vassilvitskii, "Sharding social networks," in *6th Intl. Conf. on Web Search and Data Mining*, 2013.

[222] I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, “Approx-Hadoop: Bringing approximations to mapreduce frameworks,” in *Intl. Conf. on Arch. Support for Prog. Lang. and Operating Systems*, 2015.