THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

MAKING SENSE OF SOCIAL EVENTS BY EVENT MONITORING, VISUALIZATION AND UNDERLYING COMMUNITY PROFILING

Hongyun Cai

Master of Science

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2016

School of Information Technology and Electrical Engineering

Abstract

With the prevalence of intelligent devices, social networks have been playing an increasingly important role in our daily life. Various social networks (e.g., Twitter, Facebook) provide convenient platforms for users to explore the world. In this thesis, we study the problem of multi-perspective analysis of social events detected from social networks. In particular, we aim to make sense of the social events from the following three perspectives: 1) what are these social events about; 2) how do these events evolve along timeline; 3) who are involved in the discussions on these events. We mainly work on two categories of social data: the user-generated contents such as tweets and Facebook posts, and the users' interactions such as the follow and reply behaviours among users. On one hand, the posts reveal valuable information that describes the evolutions of miscellaneous social events, which is crucial for people to understand the world. On the other hand, users' interactions demonstrate users' relationships among each other and thus provide opportunities for analysing the underlying communities behind the social events. However, it is not practical to manually detect social events, monitor event evolutions or profile the underlying communities from the massive amount of social data generated everyday. Hence, how to efficiently and effectively extract, manage and analyse the useful information from the social data for multi-perspective social events understanding is of great importance.

The social data is dynamic source of information which enables people to stay informed of what is happening now and who are the active and influential users discussing these social events. For one thing, social data is generated by people worldwide at all time, which may make fast identification of events even before the mainstream media. Moreover, the continuous stream of social data reflects the event evolutions and characterizes the events with changing opinions at different stages. This provides an opportunity to people for timely responses to urgent events. For another, users are often not isolated in social networks. The interactions between users can be utilized to discover the communities who discuss each social event. Underlying community profiling provides answers to the questions like who are interested in these events, and which group of people are the most influential users in spreading certain event topics. These answers deepen our understanding of the social events by considering not only the events themselves but also the users behind these events.

The first research task in this thesis is to monitor and index the evolving events from social textual contents. The social data cover a wide variety of events which typically evolve over time. Although event detection has been actively studied, most existing approaches do not track the evolution of events, nor do they address the issue of efficient monitoring in the presence of a large number of events. In this task, we detect events based on the user-generated textual contents and design four event operations to capture the dynamics of events. Moreover, we propose a novel event indexing

structure, called Multi-layer Inverted List, to manage dynamic event databases for the acceleration of large-scale event search and update.

The second research task is to explore multiple features for social events tracking and visualization. In addition to textual contents utilized in the first task, social data contains various features, such as images and timestamps. The benefits of incorporating different features into event detection are twofold. First, these features provide supplemental information that facilitates the event detection model. Second, different features describe the detected events from different aspects, which enables users to have a better understanding with more vivid visualizations. To improve the event detection performance, we propose a novel generative probabilistic model which jointly models five different features. The event evolution tracking is achieved by applying the maximum-weighted bipartite graph matching on the events discovered in consecutive periods. Events are then visualized by the representative images selected based on our three defined criteria.

The third research task is to detect and profile the underlying social communities in social events. The social data not only contains user-generated contents which describe the events evolutions, but also comprises various information on the users who discuss these events, such as user attributes, user behaviours, and so on. Comprehensively utilizing this user information can help to group similar users into communities, and enrich the social event analysis from the community perspective. Motivated by the rich semantics about user behaviours hidden in social data, we extend the community definition as a group of users who are not only densely connected, but also having similar behaviours. Moreover, in addition to detecting the communities, we further profile each of the detected communities for social events analysis. A novel community profiling model is designed to detect and characterize a community by both *content profile* (what a community is about) and *diffusion profile* (how it interacts with others).

Declaration by Author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my research higher degree candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis.

Publications during candidature

Journal papers:

- **Hongyun Cai**, Zi Huang, Divesh Srivastava and Qing Zhang. "Indexing Evolving Events from Tweet Streams", *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 27(11), pp. 3001-3015, 2015.

- Xuefei Li, **Hongyun Cai**, Zi Huang, Yang Yang and Xiaofang Zhou. "Social Event Identification and Ranking on Flickr", *World Wide Web Journal (WWWJ)*, 18(5), pp. 1219-1245, 2015.

Conference papers:

- **Hongyun Cai**, Zi Huang, Divesh Srivastava and Qing Zhang. "Indexing Evolving Events from Tweet Streams (Extended Abstract)", In *Proceedings of 32nd IEEE International Conference on Data Engineering (ICDE)*, 2016.

- **Hongyun Cai**, Yang Yang, Xuefei Li and Zi Huang. "What are Popular: Exploring Twitter Features for Event Detection, Tracking and Visualization". In *Proceedings of 23rd ACM International Conference on Multimedia (ACM MM)*, pp. 89-98, 2015.

- **Hongyun Cai**, Zhongxian Tang, Yang Yang and Zi Huang. "EventEye: Monitoring Evolving Events from Tweet Streams" (Demo). In *Proceedings of 22nd ACM International Conference on Multimedia (ACM MM)*, pp. 747-748, 2014.

- **Hongyun Cai**, Zi Huang, Xiaofeng Zhu, Qing Zhang and Xuefei Li. "Multi-Output Regression with Tag Correlation Analysis for Effective Image Tagging". In *Proceedings of 19th International Conference on Database Systems for Advanced Applications (DASFAA)*, pp. 31-46, 2014.

- Xuefei Li, **Hongyun Cai**, Zi Huang, Yang Yang and Xiaofang Zhou. "Spatio-temporal Event Modelling and Ranking". In *Proceedings of 14th International Conference on Web Information System Engineering (WISE)*, pp. 361-374, 2013. **[Best Paper Award]**.

- Bicheng Luo, Zi Huang, **Hongyun Cai** and Yang Yang. "Imagilar: A real-time image similarity search system on mobile platform" (Demo). In *Proceedings of 14th International Conference on Web Information System Engineering (WISE)*, pp. 535-538, 2013.

Under review:

- **Hongyun Cai**, Vincent Zheng, Fanwei Zhu, Kevin Chen-Chuan Chang and Zi Huang. "From Community Detection to Community Profiling", submitted to VLDB 2016.

Publications included in this thesis

Hongyun Cai, Zi Huang, Divesh Srivastava and Qing Zhang. "Indexing Evolving Events from Tweet Streams", *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 27(11), pp. 3001-3015, 2015. - incorporated as Chapter 3.

Contributor	Statement of contribution
Hongyun Cai (Candidate)	Designed algorithm (80%) Discussed problem formalization (70%) Designed experiments (80%) Wrote the paper (70%)
Zi Huang	Designed algorithm (20%) Discussed problem formalization (10%) Designed experiments (10%) Revised the paper (10%)
Divesh Srivastava	Discussed problem formalization (10%) Designed experiments (10%) Revised the paper (20%)
Qing Zhang	Discussed problem formalization (10%)

Hongyun Cai, Zi Huang, Divesh Srivastava and Qing Zhang. "Indexing Evolving Events from Tweet Streams (Extended Abstract)", In *Proceedings of 32nd IEEE International Conference on Data Engineering (ICDE)*, 2016. - incorporated as Chapter 3.

Contributor	Statement of contribution
Hongyun Cai (Candidate)	Designed algorithm (80%) Discussed problem formalization (70%) Designed experiments (80%) Wrote the paper (70%)
Zi Huang	Designed algorithm (20%) Discussed problem formalization (10%) Designed experiments (10%) Revised the paper (10%)
Divesh Srivastava	Discussed problem formalization (10%) Designed experiments (10%) Revised the paper (20%)
Qing Zhang	Discussed problem formalization (10%)

Hongyun Cai, Yang Yang, Xuefei Li and Zi Huang. "What are Popular: Exploring Twitter Features for Event Detection, Tracking and Visualization". In *Proceedings of 23rd ACM International Conference on Multimedia (ACM MM)*, pp. 89-98, 2015. - incorporated as Chapter 4.

Hongyun Cai, Zhongxian Tang, Yang Yang and Zi Huang. "EventEye: Monitoring Evolving Events from Tweet Streams". In *Proceedings of 22nd ACM International Conference on Multimedia (ACM MM)*, pp. 747-748, 2014. - incorporated as Chapter 4.

Contributor	Statement of contribution
Hongyun Cai (Candidate)	Designed algorithm (90%) Discussed problem formalization (70%) Designed experiments (80%) Wrote the paper (70%)
Yang Yang	Discussed problem formalization (10%) Revised the paper (10%) Designed experiments (10%)
Xuefei Li	Designed algorithm (10%) Designed experiments (10%)
Zi Huang	Discussed problem formalization (20%) Revised the paper (20%)

Contributor	Statement of contribution
Hongyun Cai (Candidate)	Designed algorithm (90%) Discussed problem formalization (70%) Designed experiments (60%) Wrote the paper (70%)
Zhongxian Tang	Discussed problem formalization (10%) Designed experiments (30%)
Yang Yang	Designed algorithm (10%) Designed experiments (10%)
Zi Huang	Discussed problem formalization (20%) Revised the paper (30%)

Hongyun Cai, Zi Huang, Xiaofeng Zhu, Qing Zhang and Xuefei Li. "Multi-Output Regression with Tag Correlation Analysis for Effective Image Tagging". In *Proceedings of 19th International Conference on Database Systems for Advanced Applications (DASFAA)*, pp. 31-46, 2014. - incorporated as Chapter 4.

Contributor	Statement of contribution
Hongyun Cai (Candidate)	Designed algorithm (70%) Discussed problem formalization (60%) Designed experiments (70%) Wrote the paper (70%)
Zi Huang	Discussed problem formalization (20%) Revised the paper (20%)
Xiaofeng Zhu	Designed algorithm (30%) Designed experiments (20%)
Qing Zhang	Discussed problem formalization (10%) Revised the paper (10%)
Xuefei Li	Discussed problem formalization (10%) Designed experiments (10%)

Contributions by others to the thesis

For all the published research works included in this thesis, the principle advisor of the author, Dr. Zi Huang, has provided valuable insights in the overall as well as the technical details. She also assisted with both the refinement of the idea and the pre-submission edition.

For all of the research problems in this thesis, the principle advisor of the author, Dr. Zi Huang, and the associate advisor Dr. Qing Zhang, assisted in providing guidance for problem formulation, idea refinement as well as reviewing and polishing the presentation.

Prof. Divesh Srivastava has provided very constructive comments to help me refine my work. Dr. Xuefei Li and Zhongxian Tang participated in conducting experiments. Dr. Yang Yang and Dr. Xiaofeng Zhu helped problem formulation and paper refinement.

Statement of parts of the thesis submitted to qualify for the award of another degree

"None"

Acknowledgments

I would never have been able to finish my dissertation without the guidance of my advisors, help from the colleagues at DKE and support from my friends and my family. It is a pleasure to express my gratitude to all of them in my acknowledgement.

My first and heartfelt gratitude goes to my principal advisor Dr. Zi Huang whose constant encouragement, excellent guidance, patient and support from the initial to the end enabled me to go through the three and half years of my PhD life. Thank you for enlightening me the first glance of research while I was still a graduate student. I will never forget the sleepless nights when you were there accompanying me to work late before deadlines.

I would also like to express my special appreciation and thanks to my associate advisor, Dr. Qing Zhang, for his support, entertainment and caring which support me academically and emotionally during these stressful and difficult moments. I appreciate the opportunity to work together with other scientists in CSIRO.

I would like to express my gratitude to DKE, UQ and CSIRO for offering me the opportunity to experience the charm of academics. This research would not have been possible without the financial support from those agencies.

I am sincerely grateful to Prof. Divesh Srivastava and Prof. Kevin Chen-Chuan Chang, for their insightful comments from rich experiences. I appreciate all their contributions of time and ideas for my research work. I would like to express my gratitude to Prof. Heng Tao Shen for his constructive comments about my work from his immense knowledge in many areas, and more importantly for his valuable advices on research attitude. I am very thankful to Dr. Vincent W. Zheng, who is my best role model for a scientist and mentor, for his tremendous academic support, continuous understanding and inspiration. Acknowledge also goes to Prof. Xiaofang Zhou for the insightful discussions about the research. I also enjoy the interesting lunch talks with Prof. Xue Li. Special thanks go to Dr. Xiaofeng Zhu, Dr. Yang Yang, Dr. Jiajun Liu, Dr. Sen Wang, Dr. Fanwei Zhu for their constructive advice and help of my work. My sincere thanks also go to other advisors in DKE group for their persistent help and support.

I have been blessed with a friendly and cheerful group of friends and colleagues. Xuefei Li, you are my biggest surprise in Australia. Thank you for being in every single of my treasured memories in the past three years. Mei Li and Yiya Wang, thank you for always being the most supportive listener remotely. Sayan Unankard, thank you for taking me to the best Thai restaurants in Brisbane and always caring and encouraging me like my elder brother. Kun Zhao, Jiping Wang, Ling Chen, Wei Wang, Chao Li and Jiewei Cao, I am grateful for the many wonderful dinners we had together and for our memorable trips into the islands and beaches. Thank you for making my time at Brisbane

enjoyable. I would like to offer my regards and blessings to all my colleagues in ADSC. Yong Pei, Zhengjia Fu, Xiaoyan Yang, Chen Liang and Penghe Chen, thank you for taking care of me and making me feel at home in Singapore.

Finally, to my beloved parents Yongsheng Cai and Meiyun Li, thank you for always being there for me. Your spirit support, unconditional trust and love make me who I am now. I would not have made it this far without you.

Keywords

social event understanding, event monitoring, event indexing, social community profiling

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080609, Information Systems Management, 80%

ANZSRC code: 080109, Pattern Recognition and Data Mining, 20%

Fields of Research (FoR) Classification

FoR code: 0806, Information Systems, 80%

FoR code: 0801, Artificial Intelligence and Image Processing, 20%

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we study the problem of multi-perspective analysis of social events. To make sense of the social events detected from social networks, we approach the event understanding problem by analysing events from different angles, including event contents, event evolutions and users who involved in the event discussion. Specifically, we have the following three major research tasks: textual event monitoring and indexing, social event tracking and visualization, and the underlying community profiling. In this chapter, we introduce the background, research challenges, and our contributions to each of the three research tasks. The thesis organization is given at the end.

1.1 Background

With the prevalence of intelligent devices and the rapid advancement of information technology, social networks have been playing an increasingly important role in people's daily life in recent decades. Various social networks (e.g., Twitter, Instagram, Facebook, etc.) provide convenient platforms for users to share information, to communicate with each other and to explore the world. Over the past decades, there has been a burst of growth in both the popularity of social network users and the volume of social media data. In this thesis, we study the problem of multi-perspective analysis of social events from social networks. In particular, we aim to make sense of the social events from the following three perspectives: 1) what are these social events about; 2) how do these events evolve along timeline; 3) who are involved in the discussions related to these events. We mainly work

on two categories of social data: the user-generated contents (noted as *posts* in this thesis) such as tweets, Facebook posts, and the *users' interactions* such as the follow and retweet behaviours among users. On one hand, the quickly updated posts reveal the up-to-date information on breaking news, hot discussions, public opinions, and so on. This valuable information describes the evolutions of miscellaneous events that are happening locally and globally, which is crucial for people to understand the world. On the other hand, people may not only be interested in what is happening now but also in the users and communities who actively discuss these events. For example, the company would like to have a knowledge of people's sentiments and opinions on the events related to their products. The government may be interested in the influential users who spread the information of certain emerging political events. Users' interactions demonstrate users' relationships among each other and thus provide opportunities for detecting and analysing the underlying communities behind the social events. However, it is not practical to manually detect social events, monitor event evolutions or profile the underlying communities from the massive amount of social data generated everyday. Hence, how to efficiently and effectively extract, manage and analyse the useful information from the social data for multi-perspective social events understanding is of great importance.

The social data streams are dynamic sources of information which enable individuals, companies and even government organizations to stay informed of what is happening all over the world and who are the active and influential users discussing these social events. For one thing, social data is generated by people worldwide at all time, which may make fast identification of events even before the mainstream media. Moreover, events typically evolve over time. The continuous stream of social data reflects the development of events and characterizes the events with changing opinions at different stages. This provides an opportunity to people for timely responses to urgent events. For another, users are often not isolated in social networks. There are different kinds of interactions between users, such as follow, retweet, reply, et al. These interactions can be utilized to discover the communities who discuss each social event, in which a community is defined as a group of users who are densely connected with each other and share similar topics of interest. The correctly detected communities help to make sense of the social events from the community perspective. Profiling the underlying communities behind the events discussions provides answers to the questions like who are interested in these events, how do people interact with those who share similar ideas and those who

hold different opinions, and which group of people are the most influential users in spreading certain event topics. These answers deepen our understanding of the social events by considering not only the events themselves but also the users behind these events.

We take the match between Lee Sedol and Google's AlphaGo as an example for illustration. AlphaGo challenged South Korean professional Go player Lee Sedol with five games from 9 March to 15 March 2016. A lot of posts about this series of games appeared on various social networks. People's focuses also changed as this event unfolded. For example, before the first game, people were interested in whom would be the winner. After AlphaGo won the first game, some people were amazed by the development of advanced technology. But some others worried that AI would reign the human world someday. After a series of failures of Lee, some people stood out to judge the fairness of this game. As can be seen from the example, the general interests of posts from social networks can precisely reflect the evolution of events. Moreover, we can group users into different communities based on their topics of interest and their interactions. This will provide the additional event information like users' sentiments and community level information diffusion strength, thus give us a more comprehensive knowledge and understanding of these events. In this example, people from the IT community may have more positive attitude on AlphaGo than those from the Go community. In addition to providing a community perspective to describe these events, the correctly detected communities can also benefit a lot of other applications, such as marketing research and public policy making. For example, those who have an open mind to AI are more likely to be the potential buyer of an AI product.

The event detection and analysis has long been studied in the data mining community. In contrast to the well-written, structured, and edited traditional data (e.g., news, articles, etc.), social data contains large amounts of informal, irregular and abbreviated words, a large number of spelling and grammatical errors, and a large proportion of them are meaningless messages [5]. Moreover, the posts are usually much shorter in length (e.g., tweets are restricted in 140 characters). These characteristics pose new challenges to the social event detection problem which are different from those faced by the conventional event detection tasks. How to take advantage of the up-to-date and comprehensive information hidden in the noisy and short posts is the main challenge for social event detection. Moreover, the events detected from the quickly updated social data streams are typically evolving

with time. How do we efficiently capture the dynamic changes of social events from the stream data is not trivial. The event monitoring algorithms for such kind of dynamic data have to be efficient, scalable and incremental. Last but not least, users are either actively interacting with others to express their opinions in the discussion of the social events or spreading the event relevant information in the social networks. How can we comprehensively model the heterogeneous user interactions and diffusion behaviours for underlying community profiling so as to understand the events from the user perspective remains challenging.

In this thesis, we focus on the research problem of analysing social events by event detection, event monitoring and underlying community profiling. Social data consists of multiple features (e.g., text, image, location, timestamp, etc.), which describe events from different perspectives. Comprehensively modeling these features helps to identify events from the social data streams accurately. As we have introduced before, events are dynamically changing over time. We can either detect events and monitor their evolutions simultaneously or conduct these two tasks one after the other. The first option is suitable for the stream processing. Given a continuous data stream, the arrival of a new post will trigger one or more of the following four event operations: the creation of a new event, the growth of an existing event, the split of an event and the merges of several events. These four event operations inherently record the event evolutions. The pipeline option is suitable for batch processing, in which we periodically detect events from a collection of posts and then track the event evolutions by analysing the relationships between the events detected in consecutive periods. Apart from event contents and event evolutions, analysing the underlying communities for each social event is another crucial perspective towards event understanding since users are one of the most important elements of the social events. In addition to posting contents from which events can be detected, users actively interact with each others in the social networks. A holistic modeling of the heterogeneous user interactions helps to detect and profile the communities who discuss these social events.

The rich and continuous flow of social data provides reliable information source about daily life stories, latest news and public opinions. In this thesis, our objective is to effectively detect and analyse social events from different perspectives. In particular, we aim to explore social data from different aspects to detect social events, monitor and visualize the event evolutions, as well as find and profile communities behind these events. Our research can cater to a large variety of different

needs. For instance, the detected events can help individuals to stay informed of the latest news. The event evolutions monitoring process provides the government the opportunity for timely responses at different stages. The community profiling helps companies to improve decision making and business intelligence. Specifically, we focus on the following three main research tasks in this thesis.

- Monitoring and indexing the evolving events detected based on the textual contents generated by users in social networks.
- Exploring multiple social data features to track and visualize the evolutions of social events.
- Jointly detecting and profiling social communities who discuss the social events by holistically modeling user links, user generated contents and user content diffusion.

1.2 Research Problems and Challenges

We systematically study the problem of textual event monitoring and indexing, social event tracking and visualization, and underlying social community profiling in this thesis. In this section, we introduce the motivations and challenges for each of the three research problems.

1.2.1 Textual Event Monitoring and Indexing

In this thesis, a social event is defined as a significant occurrence of users' anomalous activities in social networks, which is unusual relative to normal patterns of users' behaviours [38]. In the first task, we focus on the textual event, which are detected from the social data based on the user-generated textual contents. In particular, a textual event is a cluster of posts sharing similar textual information. Since social data stream is highly dynamic, the events detected from social data are evolving quickly. Event evolution exhibits event changes across successive time stamps. For example, public opinions for a U.S. presidential election may change over time. Its evolution can unfold a history of the event and characterize the event with changing opinions over time, providing an opportunity for timely responses at different stages. Thus, how to efficiently and effectively monitor event evolution from the social data streams is significant.

Given a social data stream, the event evolution monitoring problem aims to incrementally generate a set of events along the timeline and analyse the evolving relationships among them, e.g., which event is the ancestor of another one. Promptly and correctly detect the events and monitor their evolutions help to make sense of the influence and the trend of social events. The characteristics of social data pose two main challenges to event evolution monitoring. First, the textual content of a post is short and noisy, which may affect the quality of event tracking. Second, posts keep arriving in a streaming fashion. According to the online statistics, there are around 58 million tweets posted on Twitter per day on average. The event monitoring algorithms for such kind of dynamic social data have to be scalable and incremental without *a priori* knowledge. Existing event detection methods, such as the single-pass incremental clustering algorithm [3, 7, 41], can hardly be applied to event evolution monitoring due to the following two limitations. First, once a cluster is created, it can only be updated by inserting new tweets, which is too restrictive to capture the evolution of events. Second, it lacks indexing support. It needs to compare the newly arriving tweet with all existing events exhaustively. Without an index, the performance will deteriorate when the data size and the number of events grow quickly. However, indexing structures which are designed for long text documents such as Web pages cannot efficiently process short and rapidly arriving tweets. Most of the existing studies on event evolution monitoring focus on tracking the details of one event (e.g., [3, 59]). The peaks of the user activities are highlighted by analysing the posts collection belonging to a specific event, but the relationships among multiple events and the changes are missed in their work. To monitor evolutions among all relevant events, a subgraph-by-subgraph incremental tracking framework is proposed in [45]. Six primitive cluster evolution operations are defined to capture the dynamic changes of events efficiently. However, they only check the evolution of events when the time window moves and thus fail to monitor event evolutions in real time. Moreover, the setting of the time window length could be problematic.

1.2.2 Social Event Tracking and Visualization

In contrast with the traditional text documents (e.g., news articles) which only contain textual contents, social data consists of multiple features, such as image, timestamp, location, etc. This brings

opportunities which are not found in conventional media. In this task, we aim to explore various features to improve the social event detection and tracking performance. We use Twitter data for illustration purpose in this task. The benefits of incorporating different features into event detection are twofold. First, these features provide supplemental information that facilitates the event detection model. Second, different features describe the detected events from different aspects, which enables users to have a better understanding with more vivid visualizations.

Recent studies have already started to consider some of these features. However, existing methods have the following two main limitations. First, although a fast-growing percentage of tweets contain images now, few efforts have been devoted to study these images and utilize them in the event detection task. [18] is the first attempt to discover the correlations between the tweet's image and its text. It classifies the image tweets into the visually-relevant and visually-irrelevant tweets. Existing image-involved social event detection methods (e.g., [10, 91]) tend to use the whole image collections in their models despite that not all images are relevant and useful. Furthermore, the way they utilize images is to apply the bag-of-visual-words model on the extracted SIFT feature and then handle the generated visual words in the same way as textual words. This straightforward process of images may fail to fully capture the rich visual information embedded in images. Second, even though different features have been utilized in different models (e.g., spatio-temporal information [99], images [10, 11], hashtag [75]), no one has comprehensively exploited all these features in one framework.

Moreover, different from the well-written, structured traditional documents, social data contains a large number of meaningless and polluted content and are usually short. These characteristics present two more challenges to the event detection task. First, the posts are usually restricted in length, e.g., a tweet has maximum 140 characters. This leads to a problem that most existing topic models may not work any more for social event detection from such data streams. For example, Latent Dirichlet Allocation (LDA) [12] has been widely used for semantic-based event detection. Although LDA is a classic yet effective algorithm, its assumption that each document is a mixture of several topics is not valid for short tweets. Intuitively, a single tweet is usually about one topic [10]. Second, posts are often overwhelmed by meaningless words and irrelevant images [96]. Existing detection methods such as LTT [99] and MMLDA [10] take no measures to filter those noisy contents.

1.2.3 Social Community Profiling

In the previous two research tasks, we analyse the social events in terms of event contents and event evolutions by exploring various social data features. In addition to obtaining the information related to the events themselves, analysing the underlying communities for each social event is another crucial perspective towards event understanding. There are two reasons. First, underlying community profiling makes the descriptions of the events more comprehensive by analysing these social events from the user perspective. Second, community profiling provides the information diffusion strengths within one community and between communities. This means that we can not only have a knowledge of the events that are happening at the moment, but also further predict how the information about the similar events will diffuse in social networks in the future. Traditionally, the research focuses on community detection from user links [26, 42, 47]. However, it is common that, users not only *link* with each other, but also *publish* content and *diffuse* content with the other users. For example, in Twitter, users not only follow each other, but also tweet and retweet from others. These different user actions provide rich semantics about the user behaviours, thus motivating us to extend the definition of community. In particular, we define a community as a group of users who are not only densely connected, but also sharing similar behaviours, i.e., publishing similar content and diffusing information in a similar way. In this task, we study a novel problem of underlying community profiling for social event analysis, which aims to characterize a community by both its "internal property" of what the community is about, and "external property" of how it interacts with other communities. Particularly, given the user links, user published content and user information diffusion as inputs, our goal is to model: 1) a community *content profile* by the event topic distribution of a community; 2) a community *diffusion profile* by a distribution of the information diffusion strength from one community to another on a certain event topic.

Systematically solving the community profiling problem is challenging. To start with, since community profiling is based on "community", do we have to first detect communities and then profile them? Our answer is no, because profiling also helps detection. If the users are not only densely connected, but also sharing similar published content and similar diffusion behaviour, they are more likely to be from the same community. In other words, modeling the content and diffusion also helps

identify the communities. Thus, it is necessary to jointly model community profiling and community detection. Secondly, user links and user information diffusion are heterogeneous in modeling communities. How should we handle the heterogeneous user interactions in modeling the communities? On one hand, in modeling communities with user links, it is often assumed that the users within a community are densely connected, and those between two communities are loosely connected; i.e., the inter-community links are expected to be "weak" [26, 42, 47]. On the other hand, in modeling information diffusion, the "weak ties" theory has recognized that the inter-community interactions are important, and they may not necessarily be "weak" [29]. Thirdly, community-level diffusion (from the user aggregation perspective) does not account for all the possible information diffusion decisions. It is common to see some diffusion due to the topic being popular at the moment (e.g., presidential election) or the information coming from some celebrities. Such topic popularity (from the content perspective) and individual preference (from the user perspective) are the other two major factors in diffusion. We need to consider all these possible factors for comprehensive diffusion modeling, and thus accurately characterize the strength of community-level diffusion.

1.3 Approaches and Contributions

In this section, we summarize our proposed approaches and contributions for each of the problems we addressed in the three research tasks.

1.3.1 Indexing Evolving Textual Events

Our Approach

Our objective of this task is to effectively and efficiently detect emerging events, capture the changes of existing events and eventually reveal events' evolution paths over continuous social text data streams. In view of the lack of effective methods for monitoring evolving events from social data, we design four event operations to capture dynamic event evolution patterns, including creation, absorption, split and merge. These four operations are able to track event evolutions over time. Further, split and merge operations can also record event relationships in the evolution process. To implement

fast event search upon the arrival of new tweets, we propose a Multi-layer Inverted List (MIL) as an event indexing structure that can support both efficient event search (for the four event operations) and real-time event update. More specifically, a multi-layer structure is constructed based on the traditional inverted list indexing to support fast search and update for large-scale event databases. By exploiting the strong correlations among words in short tweets, the m-th layer in the structure contains the inverted lists for m-terms (i.e., a sequence of m words sorted in alphabetical order that frequently co-occur in tweets). It is designed in a way that more relevant yet shorter event lists are quickly found at the lowest layer, hence searching longer event lists at the upper layer can be largely avoided by our proposed pruning strategy. Note that MIL is an indexing structure with root (the first layer) at the top. Equipped with the MIL indexing structure, nearest neighbour search is implemented with effective upper bound pruning. A novel upper bound on event similarity is designed to significantly reduce the computational cost on the long event lists.

Contributions

- We use four event operations along with the stream clustering algorithm to effectively capture event evolution patterns over time.

- We propose a novel event indexing structure, referred to as Multi-layer Inverted List, to facilitate event search and event update for large-scale, dynamic event databases.

- We present efficient algorithms for nearest neighbour search with upper bound pruning to avoid a large proportion of expensive event similarity computations.

- We conduct an extensive performance study on dynamic event databases generated from over ten million tweets and the results demonstrate the superiority of our methods over existing methods.

1.3.2 Multi-feature Event Modeling: Detection, Tracking and Visualization

Our Approach

In this task, we aim to effectively detect social events by exploring various social data features, track the evolutions of the detected events and visualize each of them to provide a vivid and intuitive presentation. A novel probabilistic generative model is proposed for social event detection, which takes five different features including text, image, timestamp, location and hashtag as the input. Notably, only text is compulsory and the other four features may not be necessarily present. Events are then periodically detected. The event evolution tracking is achieved by applying the maximum-weighted bipartite graph matching on the events discovered in consecutive periods. In the final step, the events are visualized by summarizing images of all the posts related to the events, where the representative images are selected based on our defined three criteria. Different from the traditional topic models, in our model each short tweet is generated from one topic rather than a mixture of topics. Importantly, we design a two-layer filter framework to address the issue of the noisy social images. An SVM classifier is applied as a preliminary filter to remove noisy images, followed by a latent filter in our model which classifies the images as general images and specific images. Only the specific images are used for event detection. In other words, the first filter (SVM classifier) removes the stop images and the second filter (latent in the STM-TwitterLDA) removes the general images.

Contributions

- We design a novel generative probabilistic model for effective event detection from social data streams which jointly models various features. The intrinsic correlations among these features are fully explored for the performance improvements.
- We extensively study the impact of images on the social event detection performance and propose a two-layer filter framework to address the issue of the noisy social images.
- We define three criteria to select the representative images for event visualization and verify its quality by a case study.
- We conduct comprehensive experiments on a large real-world tweet dataset. The results prove

the advantages of our model over existing studies.

1.3.3 Community Profiling for Social Events Understanding

Our Approach

In this task, we propose a novel Community Profiling model to analyse the underlying communities who discuss the social events detected from social networks. Our model simultaneously detect and profile communities with the consideration of heterogeneous user interactions and comprehensive diffusion modeling. The three challenges mentioned in Section 1.2.3 are addressed as follows. For *joint profiling and detection*, we novelly introduce community assignments as the latent variables to holistically generate all the user links, published content and information diffusion together. As a result, we can do community detection, content profiling and diffusion profiling at the same time. For *heterogeneous user interactions*, we novelly design different distributions to generate the user links and diffusion links from the community assignments. In terms of the user link distribution, we particularly enforce that two users sharing similar community assignments are more likely to generate a user link between them. As for the diffusion link distribution, we allow the inter-community diffusion strength to be learned. For *comprehensive diffusion modeling*, we consider all the three factors, including community interaction strength, individual user preference and time-sensitive topic popularity, together to define our diffusion distribution. Moreover, we also allow the importance of each factor to be learned automatically from the data.

Contributions

- We study a new problem of community profiling for social event analysis, which provides a comprehensive description of events from the user perspective.
- We for the first time holistically model user links, user published content and user content diffusion together in a principled manner for profiling communities.
- We contribute a community profiling model to consider joint profiling and detection, heterogeneous user interactions and comprehensive diffusion modeling.

- We evaluate our model on real-world data sets, which has evidenced that our model outperforms the state-of-the-art baseline by 22.65% on link prediction task and 18.74% on diffusion prediction task.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review the literature related to the three research topics included in this thesis. Chapter 3 studies the textual event monitoring and indexing problem. We present a stream clustering algorithm to monitor evolving events from the continuous social text streams. A novel event indexing structure is proposed to support fast event search and real-time event update. Chapter 4 explores multiple social data features in addition to text for social event tracking and visualization. A probabilistic model is proposed to detect events from social data streams, and then the random walk algorithms are performed to track the events evolutions and select the representative images for event visualization. Chapter 5 focuses on the social community profiling for social events understanding. We design a novel generative model to jointly detect and profile communities from social data with the consideration of both heterogeneous user interactions and comprehensive diffusion modeling. Finally, we summarize the conclusions and future research directions of this thesis in Chapter 6.

Chapter 2

Literature Review

In this chapter, we provide an overview of the most relevant literatures to the three research problems included in this thesis. In particular, we review the related work on the textual event monitoring and indexing in Section 2.1, in which the state-of-the-arts methods for both textual event detection/tracking and text indexing are introduced. In section 2.2, we discuss the related work on two relevant topics of social event tracking and visualization: topic models for social event detection and representative image selection. Section 2.3 compares the literatures related to community profiling from three aspects: joint community profiling and detection, heterogeneous user interactions and diffusion granularity level.

2.1 Textual Event Monitoring and Indexing

In this section, we review related work on two topics: social event detection/tracking and text indexing, and then present the differences between existing work and ours at the end.

2.1.1 Textual Event Detection and Tracking

Event Detection. Recently, considerable efforts have been devoted to summarizing online textual streams such as tweets in the form of events [1, 81, 48, 49]. There are various ways to give a taxonomy of all these related works. As shown in [5], according to the *detection task*, the techniques can be classified into retrospective event detection (RED) [31] and new event detection (NED) [46]. Based

on the *event type*, they are categorized into specified [60] and unspecified [8] event detection. The former relies on the prior available information on the event of interest, while the latter detects events from the bursts or trends in Twitter streams. Depending on the *detection method*, there are supervised [73] and unsupervised [92] algorithms. In our work, we focus on NED, unspecified and unsupervised event detection. We undertake a brief review of several representative studies in the same category. To capture emerging events, [4] identifies the locally dense sub-graphs from a graph under a streaming model. In [92], daily signals for each word are constructed by applying wavelet analysis. The words are clustered into events using a modularity-based graph partitioning algorithm. This kind of daily detected events lose track of their developments over multiple days. In [2], the single-pass incremental clustering algorithm is adopted for event detection and the threshold is dynamically generated by the statistics of existing clusters. However, the influence of the cluster center shift is ignored, thus it fails to capture the dynamics of events. There are some efforts to event detection from the natural language processing community. In [70], a new event is identified if its similarity to the closest existing tweets is below a certain threshold. Then to track the development of the event, a real-time tracking component is designed to maintain a sliding window of tweets. The most informative terms about the event are used to retrieve new tweets about this event. After that, the extractive summarisation is performed to reduce the non-relevant and redundant tweets with respect to the existing event summary and then maintain a updated event summary in real-time. In [19], the authors propose a novel event-extraction method which automatically extract lexical-level and sentence-level features to train the multi-class classification via dynamic multi-pooling convolutional neural networks. Specifically, a word-representation model is designed to capture meaningful semantic regularities for words and a convolutional neural network is adopted to capture the sentence-level features. In [68], a generative model is designed to event schema induction. In addition to the head words, a list of attribute relations and a list of trigger relations are adopted to represent an entity for entity disambiguation. These algorithms mainly rely on how to extract more representative and effective key features from the entity-level or sentence-level to identify and track the events, while we focus on design more effective algorithms on the bag-of-word model (augmented term frequency vectors to represent each tweet) to monitor the dynamic events efficiently.

Event Evolution. Most of the work on event evolution tracks the details of one specified event

in real time. For example, in [3], the authors track the representative tweets of an event and mine geographical diffusion trajectory on the map. Instead of tracking one individual event, we focus on event evolution, aiming to monitor the event evolution among relevant events. To the best of our knowledge, [45] is the only effort to monitor event evolution. In their work, a subgraph-by-subgraph incremental tracking framework is proposed for event monitoring. A skeletal graph is designed to summarize the information within a fading time window in a dynamic network. However, the evolution graph is constructed by treating each event snapshot as a node and the trajectory between snapshots as paths. So the evolution of an event is only checked when the time window moves. This time window strategy faces the problem that long time window length may lead to losing track of highly dynamic events' evolutions and the short time window length will lead to storing redundant snapshots for steady events.

2.1.2 Text Indexing

Traditional Text Indexing. Text indexing is an important area and has been widely studied to facilitate textual information retrieval in various information systems. Among all the indexing structures (e.g., [106], [101]), the inverted file has proven to be the most effective structure for text similarity search [108]. The relevant research has been explored in many directions, including index construction, index maintenance, index storage, index compression, etc. We are particularly interested in index construction and maintenance. The traditional inverted file maintains a list for each word in the vocabulary, where each list contains the identifiers of the documents that contain the word and some other advanced information such as the in-document term frequencies. Different variants have been proposed to improve the performance. Phrase index (e.g., [15]) stores word sequences rather than individual words as the index terms to support fast phrase queries. But the word sequence in phrase index is a phrase which is an ordered list of words in natural language, while our index term is a set of frequently co-occurring words sorted in alphabetical order. [71] proposed a two-layer indexing structure over the partitions of author names and on the clusters inside each partition to efficiently retrieve an author query to its closest cluster for duplicate detection. Their indexing structure is constructed based on the offline cluster results while ours is used to accelerate the processing of online

clustering. The most relevant work to ours is the integration of the trie-tree structure and inverted file to index set-valued attributes for superset, subset and equality queries [87]. The key idea is to store the frequent items as a trie-tree structure in main memory, along with the corresponding inverted sublists. Hence the IO cost of transferring the disk pages with the inverted lists to main memory is reduced. Our work is complementary to theirs, since their index structure is designed to reduce the IO cost, while our in-memory index structure stores the upper bound information and provides tighter upper bounds in the lower layers to prune the long retrieved lists in upper layers so as to reduce the similarity computation cost.

Indexing for Social Data. Unlike traditional text indexing, the indexing structure for social data should be capable of ingesting the data rapidly and support efficient search and quick update on large-scale data. In [13], a retrieval engine which supports Twitter's real-time search is introduced. The authors focus on how to organize the inverted list indexing to support low-latency, high-throughput retrieval and how to implement concurrency management. In [16], a query-based classification approach is designed to distinguish between tweets that may appear as a search result with high probability and the noisy tweets. The former will be indexed by the inverted list in real-time, and the latter will be indexed in a background batch scheme. As we can see, all these indexing structures are constructed to index tweets and the search algorithms are designed to support fast tweet search on tweets data not event search on the detected events (clusters of tweets). The difference is that only a limited number of words exist in one tweet (even with weights) and the tweets search uses a boolean query language, i.e., search tweets containing (or not containing) specified search term(s); while for events, there might be hundreds of words in one event and only a few of them are dominant words (i.e., words with much higher weight than others). Moreover, event search is nearest neighbour search which is more complex than boolean queries. To the best of our knowledge, only one event indexing structure named variable dimensional extendible hash (VDEH) [99] has been proposed before. Given a query tweet, they use the fraction of matched tweets in an event to calculate the similarity between a query tweet and an event. VDEH stores highly similar tweets together in one bucket of the hash to accelerate the comparison between the query tweet and each event. Besides indexing, another approach for improving efficiency is the distributed processing topology [62]. However, for the highly dynamic Twitter data stream, detecting events immediately after the tweets are posted is of great importance.

Unlike the distributed system, our proposed in-memory indexing structure can process data in real time by avoiding disk IO.

Summary

Our work can be distinguished from previous work in several ways. First, in addition to event detection, we aim to track evolving events over time. Unlike previous work which checks the evolution patterns only when the time window moves, we record the evolution patterns and identify event relationships whenever events evolve. Second, we design an indexing structure for event databases to support event search and update. Although VDEH has been proposed to index events, the actual data indexed by VDEH contains all the posts in each event, which leads to large storage cost. In contrast, ours is an in-memory indexing structure which stores only the summary representation of the event. By avoiding the IO cost, we are able to process the highly dynamic social data in real time. Third, we design new and tight upper bounds on the Cosine similarity between posts and events to quickly reduce the search space for nearest neighbour search. Fourth, existing work on indexing social data mostly focuses on organizing the quickly incoming posts while our work aims to improve the performance of searching the detected events given posts.

2.2 Modeling Social Data for Social Event Tracking and Visualization

In this section, we review the existing studies from two aspects which are the two main components of our proposed work, topic model for social event detection and the representative image selection for event visualization.

2.2.1 Topic Models for Social Event Detection

Event detection in social networks has been extensively studied in recent years. As mentioned in [10], majority of these methods can be categorized into frequency-based, semantic-based, graph-based and machine learning-based. We focus on the semantic-based methods. More specifically,

a literature review is conducted on the social event detection via LDA-based topic model. Latent Dirichlet Allocation (LDA) [12] was first proposed for topic discovery in 2003. Ever since that, many extensions and variants (e.g., [97, 77]) have been proposed due to its effectiveness and high degree of modularity. Labeled LDA [75] presents a partially supervised learning model and incorporates hashtags as labels for their model. However, not all topics in posts include hashtags. TwitterLDA is proposed in [98] as the first topic model specifically designed for tweets data. Different from the traditional formal documents, tweets are short and noisy. In view of this, TwitterLDA makes two major corresponding adjustments. First, instead of representing a document as a distribution of multiple topics, they think that each tweet is only mapped to one topic. Second, they classify the words into background words and topic words. Background words are the noisy words commonly used in all tweets while topic words are the meaningful words from some topics. Inspired by TwitterLDA, TimeUserLDA [22] extends TwitterLDA via utilizing the time factor in addition to the text for bursty topics discovery. MMLDA [10] is another extension of TwitterLDA, which appends the tweet images in their model. SIFT features are extracted from the images in tweets, and the visual words are generated by applying the bag-of-visual-words model on these SIFT features. After that, the visual words are treated in the same way as textual words in their MMLDA model. However, images in tweets are not necessarily relevant to the text [18]. Moreover, representing an image as a bag-of-visual-words may not fully exploit the visual properties existing in images, given that recent visual features generated from sophisticated methods such as Convolutional Neural Networks (CNN) can represent images more effectively than traditional features like SIFT features in many recognition tasks [43]. A lot of efforts have also been devoted to deal with the high dimensional visual feature [103], [105]. In [99], a Location-Time constrained Topic (LTT) model is proposed by extending the TOT model [90] and considering the latitude and longitude information. Three tweet features are considered in their Location-Time constrained Topic (LTT) model, including text, time and geography information. Nevertheless, LTT follows the standard LDA framework and the problem caused by the above two characteristics of the tweets are not addressed in their model. Besides, tweet images are ignored in their work.

2.2.2 Representative Image Selection

Images provide users a more vivid description of an event and representative image selection is an effective way to summarize a set of images for event visualization. There exist various image involved tasks, e.g., [104], [102]. Different summarization objectives are considered in existing work. In [93], authors design three criteria: relevance, visual consistency and burstiness to personalize trending image search suggestion. However, the diversity is not considered. [37] improves the diversity of the search results by ranking the set of near-duplicate video clusters. Similarly, [10] applies normalized cut for images clustering and then adopts manifold ranking algorithm to rank the images within each cluster. The most representative image is selected from each cluster to construct the representative images set. Diversity is also considered in [82] to measure the concept subgraph redundancies. However, for social event visualization, social images are quite noisy and some images are not relevant enough to represent the event [18]. These less relevant images should be removed from the images set first. In this sense, a visual coherence based filter should be built before the image diversity is taken into account. In our work, we use a random walk process [35] to get the visual coherence score, which is considered together with a visual relevance score to filter out the less relevant images. After that, the bipartite graph hitting time [63] is adopted to achieve image result diversity.

Summary

Our work is different from the existing work in the following important ways. First, our topic model is specifically designed for the social data such as tweets. Although TwitterLDA has tackled the problem of short and noisy tweets contents, many other features such as images are not taken into account. Our model is capable of jointly modeling five different features with the noisy words and images filtered. Second, we use the CNN feature in our model rather than the bag-of-visual-words feature as in previous studies. CNN feature has been proved to generate consistent superior results compared to others for a wide range of visual recognition and classification tasks on different datasets [69, 76]. Furthermore, our model is well designed to be fit for the CNN feature. Third, we propose three criteria for representative image selection. In contrast to other algorithms, our schema first generates a candidate set based on the visual relevance and coherence in order to remove the less

relevant images. This is very important for the messy social images. After that, the representative images are selected from the candidates with the consideration of diversity which is omitted in [93].

2.3 Social Community Detection and Profiling

In this section, we compare the representative work related to social community profiling problem from three aspects: joint community profiling and detection, heterogeneous user interactions and diffusion granularity level.

2.3.1 Community Profiling vs. Detection

So far as we know, although there are many community detection algorithms, the problem of community profiling has not been studied. Traditionally, community detection focuses on using user-user links to find compact user communities [26, 42]. They overlook the content of users, thus unable to explain what a community is in terms of content. The recent community detection algorithms start to consider the user published content [54, 78, 79, 80]. However, they do not characterize the community interactions as we do. There are also some recent studies that consider user attributes, instead of user published free text, for community detection [61, 84, 94, 100]. In addition to lack of modeling community-level interactions, they also do not pursue further content analysis, such as topic modeling, like us.

2.3.2 Heterogeneous vs. Homogeneous User Interactions

The majority of community detection algorithms consider a single type of user interaction, e.g., the user followship in Twitter [80], Flickr [78] and Facebook/Google+ [61, 95], the co-authorship in DBLP [94, 100], the email exchange in the Enron Email corpus [80]. Although some community detection work manages to model different kinds of objects and thus different kinds of object interactions (e.g., author writes papers and publishes it in a venue [84]), it neither really models multiple types of interactions w.r.t. a single object like “user” nor only clusters “users”. Besides, it does not model information diffusion. There are a few pioneer studies that consider multiple types of user

interactions; e.g., users dig/submit/comment events and reply event comments from other users [52], two users directly contact each other or co-contact/co-subscribe a third user in YouTube [86], users follow/reply/retweet other users in Twitter [79]. However, these studies tend to model different kinds of user interactions in the same way, while we try to differentiate them in modeling. Besides, they do not model information diffusion too.

2.3.3 Community-level vs. Individual-level Diffusion

Most information diffusion studies focus on individual-level user interactions [23, 51, 53, 56]. Such individual-level diffusion modeling is arguably limited and sensitive to noise [36]. Recently, there has been some work starting to consider diffusion at the group scale [24], but its groups are predefined and it has no content information in modeling. CRM [34] proposes a community role model to model social networks and considers both individual and community factors when modeling diffusion. However, they define the community as a multinomial distribution over all user links and thus cannot guarantee the community modularity. Moreover, various user behaviours are summarized as user actions on certain topics, which does not fully exploit the rich content information.

Summary

To the best of our knowledge, we are the first to holistically model the user links, user published content and user content diffusion to profile the communities. There have been some attempts to leverage user published contents [54, 78, 80] or user attributes [84, 94, 100], together with user links, for community detection. But their goals are community detection rather than profiling, and they especially lack diffusion profile modeling. CRM [34] models both user links and diffusion links. However, they do not fully exploit the rich semantics in the content information. The closest work to ours on modeling community-level diffusion is COLD [36]. However, there are several major differences between COLD and our work: 1) COLD's goal is providing a robust representation by community to explain the diffusion, while our goal is to fundamentally profile a community, so as to answer both what the communities are and how they interact in terms of information diffusion; 2) COLD only considers a

single type of user interaction, while we consider multiple types of user interactions; 3) COLD over-simplifies the mechanism of how communities interact to produce the information diffusion, while we systematically model both community-level and individual-level factors in diffusion.

Chapter 3

Indexing Evolving Textual Events

3.1 Introduction

With the ever-accelerating development of modern technologies, social networking services have become increasingly popular nowadays. The social data streams provide a variety of real-world and real-time information on social events that dynamically change over time. Social event detection aims at discovering meaningful events from the sheer amount of social data. The correctly detected events can benefit many important applications such as emergency management, business marketing, public opinion survey, etc. Although social event detection has been actively studied, how to efficiently monitor evolving events from continuous social data streams remains open and challenging. Most of existing approaches do not track the evolution of events, nor do they address the issue of efficient monitoring in the presence of a large number of events. In this chapter, we capture the dynamics of events using four event operations (creation, absorption, split and merge), which can be effectively used to monitor evolving events. Moreover, we propose a novel event indexing structure, called Multi-layer Inverted List (MIL), to manage dynamic event databases for the acceleration of large-scale event search and update. We thoroughly study the problem of nearest neighbour search using MIL based on upper bound pruning, along with incremental index maintenance. Extensive experiments have been conducted on a large-scale real-world tweet dataset and the results demonstrate the promising performance of our event indexing and monitoring methods on both efficiency and effectiveness.

The remainder of this chapter is organized as follows. Section 3.2 formally defines the problem.

TABLE 3.1: Event Properties

Property	Notation	Description
Size	$\|E\|$	the number of posts in E
Radius	r_E	the smallest similarity from the posts in E to the cluster center
Starting time	ts_E	the initial creation time of E
Evolving period	te_E	the time duration of E from its creation to the latest update
Previous events	$prev_E$	the events where E is derived from
Next events	$next_E$	the events derived from E

TABLE 3.2: Notations for Textual Event Monitoring and Indexing

Notation	Description
Ej	the j-th event
E_i	the i-th dimension in the vector representation of E
$\overline{E_i}$	the i-th largest value among all dimensions of E
$\overline{Ej_i}$	the i-th largest value among all dimensions of Ej

Section 3.3 details event evolution monitoring strategies. MIL and its search algorithms are discussed in Section 3.4. Experimental results are shown in Section 3.5 and we conclude in Section 3.6.

3.2 Problem Formalization

In this section, we will introduce some basic concepts and the research problem we aim to address in this work.

Post. A post is a small piece of text, which is usually represented by the vector space model [41]. The augmented term frequency [58] is applied to our representation because it has been proven to show the best performance in representing dynamic, short and noisy posts [88]. Assuming the vocabulary used in our vector space model is a bag of words $\{w_1, w_2, ..., w_n\}$, we represent a post as below.

Definition 3.1. *Given a post e, it is represented by a vector $<tf(w_1), tf(w_2), ..., tf(w_n)>$, where $tf(w_i)$, the value of the i-th dimension of e, indicates the augmented term frequency of the corresponding word w_i in e, which is calculated as*

$$tf(w_i) = \begin{cases} 0.5 + \frac{0.5 \times f(w_i,e)}{\max\{f(w,e):w \in e\}} & , f(w_i, e) > 0 \\ 0 & , f(w_i, e) = 0 \end{cases} \tag{3.1}$$

where $f(w_i, e)$ is the term frequency of w_i in e, $i = 1..n$, and n is the vocabulary size. $tf(w_i) = 0$ if w_i does not appear in e.

Event. An event is defined as a cluster of posts sharing similar textual information.

Definition 3.2. *Given a similarity metric φ[1] and a similarity constraint θ, an event E consisting of a set of posts $\{e1, e2, \cdots\}$ needs to satisfy the following condition that*

$$\forall e \in E \rightarrow \varphi(e, E.center) \geq \theta,$$

where $E.center$ is defined as the mean vector of all post vectors belonging to E, in which each value is the mean of the augmented term frequencies of the corresponding word.

We simply represent an event E by its center vector $E.center$, which also applies the vector space model. Thus, the constraint θ can be considered as the smallest acceptable similarity between the event and its member posts. A number of important event properties are also recorded, which are described in Table 3.1. We assume that there is no overlap between events. In other words, any single post only belongs to one event. This is reasonable because of the small lengths of posts such as tweets. More notations related to E used in the following sections are listed in Table 5.1.

The Problem. The problem we aim to address in this work is to effectively and efficiently detect emerging events, capture the changes of existing events and eventually reveal events' evolution paths over continuous social data streams. Given a social data stream, we propose to incrementally generate a set of events $S = \{E1, E2, \cdots\}$ along the timeline and analyse the evolving relationships among them, e.g., which event is the ancestor of another one. Four operations are defined in our work to describe the evolving patterns: creation, absorption, split and merge. The evolving paths are recorded by the event properties $prev_E$ and $next_E$ (see Table 3.1). For example, if $E1$ splits into $E2$ and $E3$, we will set $next_{E1} = \{E2, E3\}$, $prev_{E2} = E1$ and $prev_{E3} = E1$.

[1] $Cosine$ similarity is used in our work.

3.3 Monitoring Evolving Events

Social events are highly dynamic and evolving quickly. It is important to capture their dynamics and record their evolutions, which help to make sense of the influence and the trend of social events. In this section, we will define four event evolution operations, based on which evolving events are captured incrementally.

3.3.1 Event Evolution Operations

We define four event evolution operations including creation, absorption, split and merge, to reflect event changes upon the arrival of new posts. Given a set of N existing events $\{Ej|j = 1..N\}$ and a new arriving post e, we first find the most similar event to e according to the predefined similarity metric φ and denote it as E_{NN}. Based on the similarity constraint θ on the events, the following operations are triggered by the arrival of e (similar to the operations in [30]).

Create: If $\varphi(e, E_{NN}) < \theta$, the similarity constraint on events is not satisfied. Thus, a new event is created, which consists of the single e.

Absorb: If $\varphi(e, E_{NN}) \geq \theta$, the new post e is supposed to be absorbed by E_{NN}. However, when e joins E_{NN}, the event center may shift, resulting in the change of the smallest similarity between the member posts and the event. If the updated smallest similarity value still satisfies the constraint θ, i.e., not smaller than θ, E_{NN} will be updated by absorbing e. Otherwise, E_{NN} will no longer be a valid event and the operation **Split** will be triggered.

Split: If the updated smallest similarity value between the member post and E_{NN} is smaller than θ by absorbing e, the bisecting *k*-Means clustering [83] will be performed on the posts contained by E_{NN} to generate two new events.

Merge: When the operation split is conducted, the newly split events may be merged with other existing events if the merged events satisfy the similarity constraint θ. In this case, the split events need to check whether any existing events can be merged, and the merged events are regarded as new events.

The time complexity to adjust the event center when absorbing a new post is linear to the number of words in the updated event. Obviously, split and merge operations indicate the evolving relationships among events. Whenever a split or merge operation happens, event properties $prev_E$ and $next_E$ of new events are updated accordingly, which record the event evolution path and reveal how events are evolving over time.

To give a better explanation of the above four operations for monitoring evolving events, we will use a real-world example, i.e., Edward Snowden's leakage of NSA documents. When Snowden just left Hawaii for Hong Kong, a lot of posts were posted to discuss him. These posts form the event "$E1$ (Snowden, NSA, Hong Kong, USA)".[2] After Snowden flew to Russia, the focus of new posts changed as well. At the beginning, these posts were absorbed by $E1$ because they were close to $E1$ by sharing a lot of common words like "Snowden", "Edward", and "USA". However, when more and more new posts were involved, the smallest similarity between posts and the event center of $E1$ was getting smaller and smaller due to the shifting focus of the new arriving posts, which made the content of the event $E1$ more and more diverse. Eventually, the focus of $E1$ changed from "Hong Kong" to "Hong Kong, Russia". At one moment, $E1$ did not satisfy the similarity constraint θ anymore after receiving the new posts. Hence $E1$ was split into "$E2$ (Snowden, Russia, USA, Putin)" and "$E3$ (Snowden, Hong Kong, NSA, leak)". Once the split operation was triggered, the newly split events (i.e., $E2$ and $E3$) could be merged with other nearby events. For example, $E2$ was merged with "$E4$ (Russia, USA, Obama, Putin)" to generate the new event $E5$ which satisfied the threshold requirement.

The only parameter in defining these four event evolution operations is the similarity constraint θ. Instead of predefining θ, we aim to automatically adjust it according to the statistics of previous query posts' nearest neighbours' similarity records. The idea is to maintain the mean μ and standard deviation σ of all the previous query posts' largest similarity values to the database events. We follow the settings in [2], and the threshold is dynamically calculated as $\mu - 3\sigma$. Statistically, the 3σ range to the mean covers more than 99.7% posts in the event for normally distributed posts. That is to say, when the similarity between a new post and its nearest neighbour event is smaller than θ, it is highly unlikely for the post to be relevant to its most similar event and a new event should be

[2]Note that, rather than using a formal vector space model, in this example, we only use four most frequent words of an event as its representatives for illustration purpose.

created. The mean and standard deviation of the largest similarity values of previous query posts can be maintained dynamically by their zeroth, first and second order moments continuously, which corresponds to the number of posts, the sum of their largest similarity values, and the sum of squared largest similarity values respectively. They can be additively calculated over the social data stream, thus are easily maintained in the streaming scenario. Denoting the three moments as M_0, M_1, and M_2 respectively, the mean μ and standard deviation σ can be obtained by these moments as $\mu = M_1/M_0$ and $\sigma = \sqrt{M_2/M_0 - (M_1/M_0)^2}$.

Besides the automatically generated threshold θ, we also define θ_U, an upper bound of θ to prevent an unnecessarily high θ value. For example, the burst of a hot event will lead to a large number of high similarity values to the event. This will make the value of θ keep increasing, and other less intensive events will fail to absorb relevant posts due to the extremely high value of the threshold setting. θ_U can help keep θ values in a reasonable range by filtering the outliers. The process of tuning θ_U is detailed in Section 3.5.2. We do not worry about the low θ value because θ generally goes up as new posts are inserted.

3.3.2 Incremental Event Evolution

Based on the above four event evolution operations, the events will be updated with the streaming arrivals of posts. Two algorithms are designed in our work to implement the four operations and update the events dynamically. The whole process is described in Algorithm 1.

Algorithm 1 starts with an empty event set S. For the very first post, it is taken as an event to be added into the event set (or database), followed by initializing order moments (lines 1-3). For any of the following posts, its nearest neighbour event and the corresponding similarity φ_{max} is first found by conducting the nearest neighbour search on the events set (line 5). The threshold θ is then dynamically computed based on M_0, M_1 and M_2 (line 6). If θ is greater than the upper bound θ_U, θ is updated to be θ_U (line 7). If the nearest neighbour event E_{NN} has smaller similarity to e than θ, a new event is created and added into the event set (lines 9). Otherwise, e is absorbed by E_{NN} (line 11). If the updated radius of E_{NN} is smaller than θ, E_{NN} is split into two new and smaller events by adopting the bisecting *k*-Means clustering [83], followed by merging events (lines 13-15). Finally,

Algorithm 1: Process a New Post

Input: Events set S, New post e, Zeroth, first and second order moments M_0, M_1, M_2, Threshold upper bound θ_U
Output: S, M_0, M_1, M_2

```
1  if S = ∅ then
2      S.add(CreateEvent(e));
3      M_0 = M_1 = M_2 = 0;
4  else
5      E_NN, φ_max ← NearestNeighbourSearch(S,e) ;
6      θ ← M_1/M_0 − 3 × sqrt(M_2/M_0 − (M_1/M_0)^2);
7      if θ > θ_U then θ ← θ_U;
8      if φ_max < θ then
9          S.add(CreateEvent(e));
10     else
11         E_NN.AbsorbEvent(e);
12         if r_{E_NN} < θ then
13             E1, E2 ← E_NN.SplitEvent();
14             S.remove(E_NN);
15             S.MergeEvent(θ,E1, E2);
16         end
17     end
18 end
19 M_0 ← M_0 + 1;
20 M_1 ← M_1 + φ_max;
21 M_2 ← M_2 + φ_max × φ_max;
22 return S, M_0, M_1, M_2;
```

three moments are updated incrementally (lines 19-21).

Algorithm 2 presents the steps to check whether the split events $E1$ and $E2$ should be merged with similar events. Given a split event E, the events with the Cosine similarity to it higher than θ are retrieved and added into a candidates set which are sorted by their similarities with E for merge operation (line 2). Then these candidate events are merged with E one by one (line 3-16) and the smallest similarity between the member posts and the new merged event is checked every time (line 5). The merge procedure for E will stop when the new merged event does not satisfy the similarity constraint θ of forming an event (lines 5-7) or when all events in the candidate set are merged (lines 13-15). When the merge procedure finishes, event properties are updated based on the merged event.

Instead of just defining the four event operations, we further work on how to support each of the

Algorithm 2: MergeEvent

Input: Similarity threshold θ, Event $E1$, Event $E2$
Output: Events set S

```
for E ∈ {E1, E2} do
    C ← GetSortedMergeCandidate(E);
    for i ← 1 to |C| do
        E* ← Merge(E,C[i]);
        if r_{E*} < θ then
            S.add(E);
            break ;
        else
            S.remove(E);
            S.remove(C[i]);
            E ← E*;
        end
        if i == |C| then
            S.add(E);
        end
    end
end
return S;
```

four operations efficiently. The details are introduced in the following section.

3.4 Event Indexing and Search

One key process in the incremental event evolution monitoring is to find the nearest neighbour of a new post (in the case of creating and absorbing) or an event (in the case of merging), which is basically a nearest neighbour search. Because posts are coming in a highly dynamic stream fashion, it is essential to implement the nearest neighbour search efficiently. In this section, we present the event indexing structure named Multi-layer Inverted List (MIL) which is proposed to index the social events data and support efficient nearest neighbour search for these events. MIL is distinct from the traditional inverted list in several ways.

The inverted file is recognized as the most efficient index structure for text similarity search [108]. As a document is represented as a bag of words, the inverted file maps the words to the documents that contain them. Given a number of query words, this structure highly reduces the search space by limiting the query processing being conducted to the documents which contain at least one of the

query words. For event indexing, with the arrival of a new post, we aim to find its closest event in order to check if it belongs to any existing event. It is essentially a nearest neighbour search. Thus, a straightforward way to organize events for search is to build an inverted file on them. However, given the fact that posts are arriving so quickly nowadays, a huge number of events may be generated and updated frequently. The conventional inverted file is no longer sufficient to support large-scale dynamic event management because the lists returned by the inverted file may be still too long. Thus, a more efficient and scalable index structure is necessary. The proposed Multi-layer Inverted List (MIL) structure organizes events in different layers guided by different information-specific levels, which provides highly efficient search performance for large-scale event sets.

3.4.1 Index Construction

Different from the conventional inverted file which usually creates entries for individual words, the proposed Multi-layer Inverted List (MIL) consists of multiple layers, where each entry on the m-th layer corresponds to a set of m words referred to as an m-term. An m-term entry points to a list of events, each of which contains all words in this m-term. In more detail, as illustrated in Figure 3.1, we build up the entries for 1-terms on layer 1, the entries for 2-terms on layer 2, and so on. More importantly, an $(m+1)$-term is generated by adding a strongly correlated word to its m-term so that the m-th layer naturally points to the $(m+1)$-th layer. For example in Figure 3.1, w_1 stands for "NSA", a 1-term on layer 1. A 2-term w_1w_3 representing "NSA, Obama" is generated and pointed from w_1, and a 3-term $w_1w_3w_{10}$ representing "NSA, Obama, Snowden" is generated and pointed from w_1w_3. The terms on each layer are sorted in alphabetical order. Each m-term entry in this index structure points to a list of events containing the corresponding m words. For instance, the events $E1$, $E6$, $E12$ and $E21$ shown in Figure 3.1 all contain the words "NSA" and "Obama". One important feature of MIL is that the event list pointed by an $(m+1)$-term is a subset of the event list pointed by its corresponding m-term. So the events from a shorter list stored on the lower layer (bigger m, contains more words in the index terms) are more likely to be relevant to the new post (query post) because they share more of the same words. This feature motivates our proposed pruning based search strategy which enables fast search by first inspecting most relevant and shorter event lists on the lowest layer,

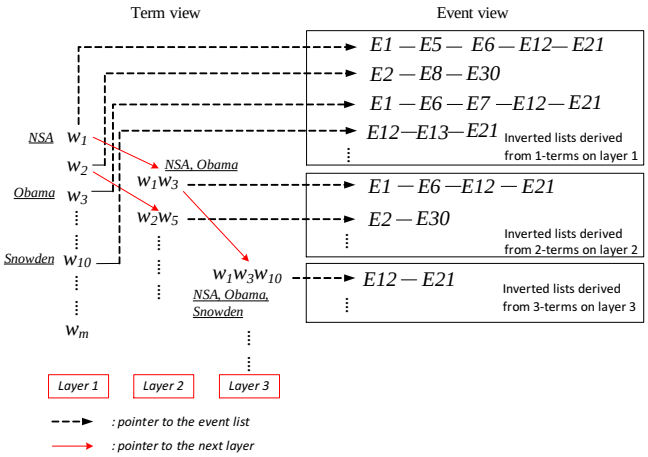


FIGURE 3.1: The Multi-layer Inverted List Structure

so as to get a larger similarity value to avoid exhaustive accesses to longer event lists on the upper layers, as will be detailed in Section 3.4.3. Obviously, the most upper layer (i.e., m=1) of MIL alone can be regarded as the conventional inverted file.

m-term Generation

The next question is how to generate m-terms on different levels in the proposed indexing structure. We create m-terms according to the vocabulary which is constructed based on the words contained in the collected posts and which grows dynamically as new posts arrive. Clearly, not all the words should be selected into the vocabulary, as many of them are trivial and meaningless even after removing the stop words. Thus, how to determine a meaningful word vocabulary is the first step. An intuitive way to measure the significance of words is using their tf·idf values. However, if the tf·idf value of a word is stable over time, most likely it does not indicate the occurrence of events. Emergence of events is usually reflected by the change of words' usage [92]. Given this observation, wavelet analysis [92] is applied in our work to capture the change of the words' usage in posts. The comparison between the tf·idf based algorithm and the wavelet analysis based algorithm is presented in the experiments.

Essentially, for each word w, a signal S_w can be generated by observing its df·idf values in a certain time period, denoted as:

$$S_w =< s_{w,t_1}, s_{w,t_2}, ..., s_{w,t_k} >$$

where $\{t_i|i = 1..k\}$ are a sequence of observation time points within the time window $[t_1, t_k]$ and s_{w,t_i} is a df·idf value of the word w at the time point t_i, which is defined as

$$s_{w,t_i} = \frac{|e \in T_{t_i} : w \in e|}{|e \in T_{t_i}|} \times log \frac{\sum_{j=1}^{i} |T_{t_j}|}{\sum_{j=1}^{i} |e \in T_{t_j} : w \in e|} \tag{3.2}$$

where T_{t_i} is the collection of posts with timestamps in the time duration $(t_{i-1}, t_i]$. Notably, df is used here because multiple appearances of the same word in one short post usually indicate the occurrence of the same event [92].

Referring to [92], based on S_w we generate a new signal S'_w to capture the entropy change of the original signal. It is defined as

$$S'_w =< s'_{w,t_1}, s'_{w,t_2}, ..., s'_{w,t_k} >$$

where

$$s'_{w,t_i} = \begin{cases} \frac{H_{t_i} - H_{t_i-1}}{H_{t_i-1}} & \text{, if } H_{t_i} > H_{t_i-1} \\ 0 & \text{, otherwise.} \end{cases}$$

H denotes the H-Measure of signal S, which is a normalized value of Shannon wavelet entropy. More details can be found from [92]. By calculating the auto-correlation value of S'_w, we can determine whether S'_w is a trivial signal. If the usage of the word w is stable over time, its associated signal S'_w is flat, resulting in a low auto-correlation. S'_w and w will be considered as trivial. In this way, we create the vocabulary by removing the trivial words. Every word in the vocabulary forms a 1-term. For each new word in the arriving posts, its signal is monitored over the time duration and added into the vocabulary if its auto-correlation is high.

For $m > 1$, m-terms are generated by finding the m-sized frequent word sets from the 1-terms periodically. One option is to re-construct m-terms whenever a significant increase in the vocabulary size is detected. A widely used method FP-growth [33] is applied to achieve efficient and scalable frequent itemset mining. The support threshold λ for FP-growth is tuned in Section 3.5.2.

The proposed indexing structure consists of m layers. Since each individual post is generally a small piece of text consisting of 2-10 meaningful keywords, the length of the largest frequent item set is usually small. m is normally in the range of 2-3, as indicated from our experimental results. The effect of m with various values on indexing performance will be further discussed in the experiments.

Event List

Each m-term points to a list of events, where event ids are stored in the list. Some events can contain multiple m-terms. For example, $E12$ (in Figure 3.1) is on the lists derived from w_1, w_3, w_{10}, w_1w_3, w_1w_{10}, w_3w_{10} and $w_1w_3w_{10}$ as it contains all these words. It is expected that the lengths of lists on the $(m+1)$-th layer are far shorter than those on the m-th layer.

Given a post e, to identify the event it belongs to, we need to calculate the similarity between e and each individual event. It is critical to reduce the search space as much as we can to support efficient identification. Basically, we only consider event lists from m-term entries, where e contains the corresponding m words. To further reduce the search cost by avoiding exhaustive and expensive similarity computations, we will calculate the similarity upper bound for the remaining events with respect to e. In order to compute upper bounds, the information about each event also needs to be maintained in MIL.

On the m-th layer in MIL, for each event E in the list pointed to by an m-term, the necessary information to calculate its upper bound (Equation 3.3) with respect to a post is the sum of the largest m values and the $(m+1)$-th largest value among all dimensions in E. Denoting the i-th largest value in E as $\overline{E_i}$, the additional information stored in the indexing structure at the m-th layer for E includes a value-pair $< \sum_{i=1}^{m} \overline{E_i}, \overline{E_{m+1}} >$.

Take the event $E12$ (in Figure 3.1) as an example. Its associated value-pair on the entry w_1 is $< \overline{E12_1}, \overline{E12_2} >$, the value-pair on the entry w_1w_3 is $< \overline{E12_1} + \overline{E12_2}, \overline{E12_3} >$ and the value-pair on the entry $w_1w_3w_{10}$ is $< \overline{E12_1} + \overline{E12_2} + \overline{E12_3}, \overline{E12_4} >$.

3.4.2 Incremental Index Maintenance

Whenever there is an update on events upon the arrival of a new post, in addition to the event database, the indexing structure needs to be updated as well. We maintain the update of the index incrementally. Specifically, after any of the four event evolution operations happens, the index will be correspondingly updated by the new or updated events. For an MIL structure with m layers ($m \leq 3$ as discussed in Section 6.2), the time complexity of the index maintenance is $O(N^m)$, where N is the number of words in the updated event. In Algorithm 1, the update of MIL will happen after line 2, line 9 and line

15. The new (or updated) events will be inserted into the MIL structure at different layers by matching the terms in MIL. Note that the vocabulary (i.e., the 1-terms) is incrementally enlarged by including the new non-stop-words in the new events, and m-terms at lower layers are periodically generated from posts. It is important to note that split or merged events are regarded as new events which are inserted into the database and indexed by MIL. The old events (events before split or merge) will be dropped from MIL (line 14 in Algorithm 1 and line 9-10 in Algorithm 2). For example, if $E1$ is split into $E2$ and $E3$, $E1$ will be removed from MIL while $E2$ and $E3$ will be inserted into MIL. In this way, at any point in time, only the active events at that moment are searchable through MIL.

Next, we present the algorithms for nearest neighbour search (i.e., line 5 in Algorithm 1) through MIL. The detailed upper bound estimation based on the information stored in MIL for the search scenario is explained.

3.4.3 Nearest Neighbour Search

For a new arriving post or newly split event, we are first interested in identifying its most similar event from the existing events indexed by MIL, corresponding to nearest neighbour search. To reduce the computational cost, it is critical to design an effective similarity upper bound for quick candidate pruning to avoid full similarity computations. In this subsection, we will discuss the upper bound estimation and the corresponding search strategy for nearest neighbour search, for a newly arriving post e. The procedure for a newly split event is the same.

Upper Bound Calculation

In the proposed MIL indexing structure, an event E may appear in multiple event lists across different layers. As we mentioned before, the events lists on the lower layer are shorter and more likely to be relevant to the given post e. Hence, the idea here is to associate the event with different similarity upper bounds at different layers for e. Such a pruning based search strategy along with the proposed multi-layer structure enable efficient search by first inspecting most relevant event lists on the lowest layer, to avoid exhaustive accesses to longer event lists on the upper layers.

We adopt the Cosine similarity to measure the similarity between an event E and a post e, denoted

as $\varphi(e, E)$. Notably, in this work, all the vectors (both e and E) are normalized. So the similarity between e and E is calculated as $\varphi(e, E) = \sum_{k=1}^{n} e_k \times E_k$, where e_k denotes the k-th dimension in e. The estimation of upper bound of the Cosine similarity $\varphi(e, E)$ for nearest neighbour search is described in the following lemma.

Lemma 3.1 (Upper Bound)**.** *Given a post e and an event E on the event list at the m-th layer, the following similarity, denoted as $\varphi_u^m(e, E)$, is an upper bound of $\varphi(e, E)$:*

$$\varphi_u^m(e, E) = (\sum_{i=1}^{m} \overline{E_i} + \overline{E_{m+1}} \times (Z - m)) \times \overline{e_1} \tag{3.3}$$

where $\overline{E_i}$ is the i-th largest value among all dimensions of E, $\overline{e_1}$ is the largest value among all dimensions in e, and Z is the number of dimensions which have non-zero values in both E and e.

Proof. When the similarity calculation is required between e and E on the list pointed to by an m-term, it indicates that e and E have at least m common words, which correspond to m dimensions. By assuming that the values of these m dimensions in E are the largest m values among all dimensions, the similarity contributed by them is no greater than

$$\sum_{i=1}^{m} \overline{E_i} \times \overline{e_1} \tag{3.4}$$

where $\overline{e_1}$ is the largest value among all dimensions in e.

Besides these m words, E and e may still share some other common words. Denote the total number of common words shared by e and E as Z. It is just the number of dimensions which have non-zero values in both E and e. The similarity contribution from the other $(Z - m)$ common words is no greater than

$$\overline{E_{m+1}} \times \overline{e_1} \times (Z - m). \tag{3.5}$$

Taking into account the above two parts, we derive that $\varphi_u^m(e, E) \geq \varphi(e, E)$. □

When m increases, the upper bound $\varphi_u^m(e, E)$ decreases monotonically for the event E with respect to e. In other words, a tighter upper bound for E is provided for a larger m.

Given that m is generally far smaller than the number of words in posts, the upper bound computation in Equation 3.3 (with time complexity $O(1)$) is expected to be more efficient than the Cosine similarity computation which has a time complexity linear to the number of words in the post. This has been verified in our experiments (Figure 3.8). Similar to threshold algorithm (TA) [25], our upper bound is calculated during the process of nearest neighbour search. However, TA needs to maintain multiple sorted lists of objects. For highly dynamic social data, frequently updating these sorted lists leads to non-ignorable extra time cost especially for the long lists. Next, we discuss the search strategy for a new post to find the most similar event from MIL, by utilizing the upper bound established above.

Search Strategy

The search against the proposed indexing structure MIL is to find the most similar event E from all existing events for a new arriving post e according to the Cosine similarity measure (i.e., the NearestNeighbourSearch method at line 5 in Algorithm 1). Given a query, the events at the $(m+1)$-th layer in MIL are not guaranteed to have higher similarities to the query than those at the m-th layer. However, it is more likely that higher similarities can be obtained at lower layers because the more dimensions/words shared by the query and an event, the more likely the Cosine similarity is large. Hence, by searching the terms with larger m values first, high similarities are more likely to be obtained in short time since events lists at lower layers are much shorter. The biggest advantage of doing so is to approach the most similar event's true similarity as soon as possible so that more events can be quickly pruned based on their established upper bounds to the query. If an event's upper bound is equal to or smaller than the currently found maximum event similarity to the query, it can be safely pruned without computing its actual Cosine similarity.

The general idea is to perform depth-first traversal in MIL and filter the events based on upper bound pruning. Let us elaborate it with our running example, as in Figure 3.2 where a partial 3-layer MIL is depicted. Given a new post e containing three keywords "NSA", "Obama" and "Snowden", we first locate the lowest layer that e can reach in the indexing structure, by traversing the terms starting with "NSA". In this example, it can reach the third layer with a 3-term containing all three words. The event list pointed to by the 3-term is then accessed. Next, the event list pointed to by the 2-term

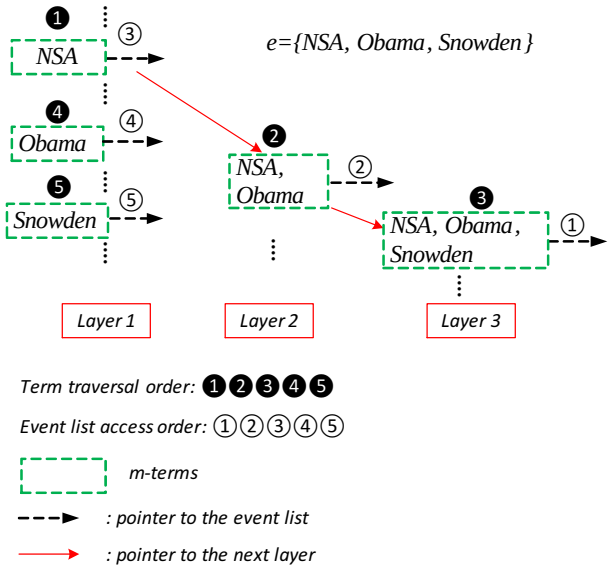


FIGURE 3.2: An Example for the Term Traversal Order and the Event List Access Order in MIL

containing "NSA" and "Obama" is accessed, and followed by the three 1-terms' lists to be accessed. In Figure 3.2, the term traversal order and the event list access order are shown in black and blank circles, corresponding to the depth-first pre-order and the depth-first post-order traversal, respectively.

Multiple event lists at each layer can be combined into a single list. Assuming m layers have been traversed in MIL, we can have m combined lists. The events belonging to more than one layer should only be retained in the list at their lowest layer (largest m) where the tightest upper bounds for events are computed. Meanwhile, the same event at the same layer is recorded only once in the combined list. For the example shown in Figure 3.1 and for the query post e containing three words "NSA", "Obama" and "Snowden", $E12$ and $E21$ only appear in the third-layer combined list, $E1$ and $E6$ are recorded in the second-layer combined list, and $E5$, $E7$, $E13$, and $E14$ are stored in the first-layer combined list.

After we get the combined event lists, the next step is to compute their upper bounds φ_u^m for the query, based on which effective pruning of events can be performed. By avoiding a large number of expensive Cosine similarity computations, the nearest neighbour, i.e. the most similar event, can be found quickly. The combined list at the lowest level is processed first, followed by its next upper level until the most upper level. The largest similarity value found so far, denoted as $\varphi_{\max}$, is maintained during the process. If the upper bound of an event is greater than $\varphi_{\max}$, the event is accessed and the actual Cosine similarity is calculated to check whether the nearest neighbour and $\varphi_{\max}$ need to

be updated. Otherwise, it can be safely pruned since its similarity upper bound is already equal to or smaller than $\varphi_{\max}$.

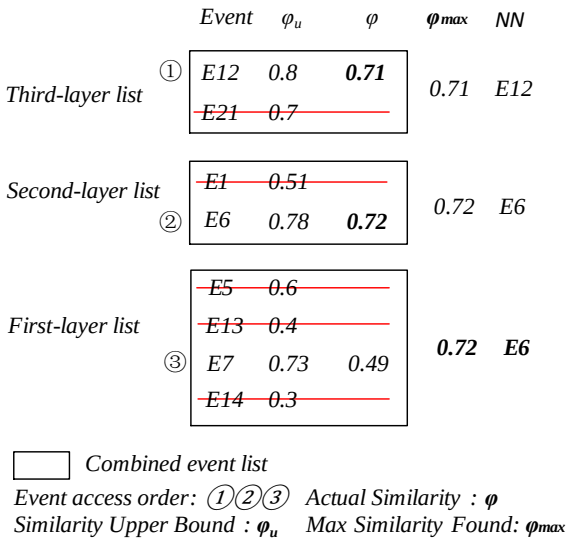


FIGURE 3.3: An Example for Upper Bound Pruning

An example for upper bound pruning is illustrated in Figure 3.3, where the third-layer to the first-layer combined lists are $\{E12,E21\}$, $\{E1,E6\}$, and $\{E5,E7,E13,E14\}$ respectively. Their upper bounds φ_u are also indicated. The third-layer list is first processed. After the event $E12$ is accessed, $\varphi_{\max}$ and the nearest neighbour are updated to be 0.71 and $E12$. Since the upper bound of $E21$ is smaller than current $\varphi_{\max}$, $E21$ can be pruned directly without computing its Cosine similarity. Then the second-layer list is processed. Since $E1$ has a smaller upper bound than $\varphi_{\max}$, it is pruned. $E6$ is accessed for Cosine similarity computation since its upper bound is greater than $\varphi_{\max}$. The actual similarity of $E6$ is 0.72, which is greater than $\varphi_{\max}$. Therefore, $\varphi_{\max}$ and the nearest neighbour are updated to be 0.72 and $E6$ respectively. Finally, the first-layer list is processed, and only $E7$ is accessed. $E6$ remains as the final result for the nearest neighbour. In this example, only three events ($E12$, $E6$ and $E7$) are required to calculate the actual similarities to the query based on upper bound pruning, with an additional cost on upper bound computations.

Notably, the upper bound calculation requires the known value of Z for each event. In our implementation, give a query, Z can be simply calculated by retrieving all its 1-term event lists and counting the number of occurrences across all the lists for each event. The search algorithm is outlined in Algorithm 3.

Algorithm 3: Nearest Neighbour Search

Input: Index I, Post e
Output: Nearest Neighbour Event E_{NN}

```
1  Lists[m] ← GenerateLayeredLists(e, I);
2  Z[] ← ComputeZ(e, I);
3  φ_max ← 0;
4  for i ← m to 1 do
5      for E ∈ Lists[i] do
6          φ_u^i(e, E) ← ComputeUpperBound(e, E, Z);
7          if φ_u^i(e, E) > φ_max then
8              φ ← ComputeCosineSimilarity(e,E);
9              if φ > φ_max then
10                 φ_max ← φ ;
11                 E_NN ← E ;
12             end
13         end
14     end
15 end
16 return E_NN;
```

In Algorithm 3, by traversing the indexing structure MIL, the maximum number of layers reached by the query post e, i.e. m, and the combined list at each layer are first generated (line 1), followed by computing Z value for each event which shares at least one common word with e (line 2). To find the nearest neighbour to e, φ_{max} is initialized to be 0 (line 3). The combined event lists are then accessed in the bottom-up order (lines 4-15), given that events at lower layers potentially have higher similarities than those at upper layers. For each event in the i-th layer list, its upper bound φ_u^i is computed (line 6). If $\varphi_u^i > \varphi_{max}$, its actual similarity to the query is computed and compared with φ_{max} to see whether the currently maintained nearest neighbour and its similarity to the query need to be updated (lines 7-13). Otherwise, the event is filtered without similarity computation.

The time complexity of the algorithm is linear to the number of events that share at least one common word with the query. However, the proposed upper bound pruning strategy starting from lower to upper layers can greatly reduce the search cost by saving a large number of the Cosine similarity computations. As the query post's length increases, such reduction becomes more significant since the Cosine similarity computation gets more expensive. This will be further verified by our experiment results.

3.5 Experiments

In this section, we report the results of an extensive performance study conducted on a large real-world tweet dataset. The experiments are designed to verify the effectiveness of the event evolution monitoring process and the efficiency of the index structure.

3.5.1 Set Up

Dataset

We collected 11,121,112 tweets posted from July 25th, 2013 to November 30th, 2013 using Twitter Search API. There are two types of APIs provided by Twitter to crawl tweets, Stream API and Search API. The former one can only return a very small fraction of the total volume of tweets at any given moment, based on which events can rarely be detected and monitored. Thus, we use Search API to crawl tweets by giving a list of search terms to get more "meaningful" tweets. We construct the search terms list by including a set of event-relevant words (e.g., accident, hurricane, etc.), as well as the top 10 hottest trend words from the G20 countries in every two hours. The content of each tweet is preprocessed by removing stop words, stemming, and obtaining nouns and verbs only. The tweets are incrementally processed in the order of their posted time to simulate the dynamic tweets stream, and the generated events can be reported at any time. To evaluate the performance of the proposed indexing method, we generate four subsets in sizes of 2 million, 4 million, 6 million and 8 million tweets in terms of their posted time. 100,000 tweets are randomly picked as new coming tweets (i.e., queries), with the length ranging from 2 to 10 words, to test the effect of query length on the efficiency of nearest neighbour search. Note that the data volume is relatively small on Twitter scale, this is because the tweets that can be crawled by Twitter API are limited and we further restrict the collected tweets to specified countries and search terms. Considering that 58 million tweets are posted everyday on average, we will extrapolate our results to the entire tweet stream in a sliding window of two weeks later.

Ground Truth Generation

Due to the lack of ground truth for event detection and evolution monitoring from Twitter, the evaluation metrics for TDT (Topic Detection and Tracking) cannot be directly used for our work. We first build up the ground truth (the top-N hottest events that happened on a particular day) from the tweet stream. In our data collection process, each tweet is collected based on a search term. Given a set of daily collected tweets, the sum of re-tweeted numbers is calculated for every search term as the mentioned times. We pick a list of search terms with the top-N largest mentioned times as the event ground truth for that day, referred to as ground-truth-N. It is not practical to construct ground truth for event evolution monitoring. Thus a user study is applied in the evaluation phase.

Performance Indicators

Three metrics are adopted to evaluate the effectiveness of the process of monitoring evolving events.

- Purity: The purity is defined as the percentage of labelled objects of the majority class in each cluster for all the clusters. As a standard measure of the cluster quality, it can be used to check the quality of an event which is basically a cluster of tweets.

- Precision: Precision is a classical and effective measure in information retrieval to evaluate the accuracy. In our work, it is calculated by dividing the number of events existing in both retrieved top-N events from our algorithm and the corresponding ground-truth-N by N.

- Coverage: Coverage is the proportion of the total ground truth events that have been detected. Specifically, it is calculated by dividing the number of different events existing in both retrieved top-N events from our algorithm and the corresponding ground-truth-N by N. In other words, when similar events correspond to one ground truth event, we will count it once only. The reason we choose coverage in addition to precision is that we want to know how many ground truth events are detected correctly, not how many detected events are ground truth events. The former is more important in real life, since recommending similar events is useless.

Three measures are used to evaluate the performance of the index structure for nearest neighbour search.

- Pruning power: Pruning power is the percentage of events that are pruned in the filtering step of search processing.

- Search time: Search time is the total query response time for a post to get its nearest neighbour event from the index. It dominates the overall time for processing a new post.

- Index update time: Index update time is the total time spent on updating the index after an event is updated by a new post.

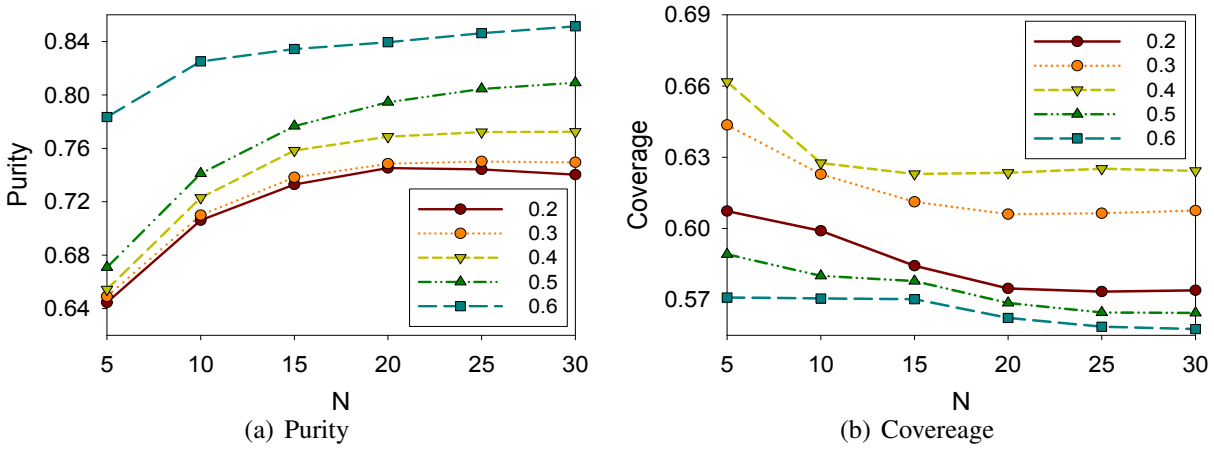
Compared Methods

We design two comparison strategies in the experiments for evaluating the performance of the evolving events monitoring and the index structure respectively.

Event evolution monitoring. We propose four operations to monitor evolving events over time. Hence we compare our event evolution monitoring (EEM) method with the traditional single-pass incremental clustering method to show the effectiveness of the four operations. There are two variants for single-pass incremental clustering (SPIC) algorithm: the one with predefined threshold [41] and the one with automatically adjusted threshold [2], referred to as SPIC1 and SPIC2 respectively. Moreover, to compare with the state-of-the-art event evolution tracking method, eTrack [45] which is based on graph partition algorithm is implemented and compared. To demonstrate the superior 1-terms generated by the wavelet analysis algorithm, the EEM-TFIDF is compared, which is our proposed event evolution method with the 1-terms consisting of the words with high tf·idf values.

Index structure. We compare our Multi-layer Inverted List (MIL) with three other methods. The first one is the traditional inverted list indexing structure (IL), which is classic and effective for text retrieval. The remaining two methods are IL with different upper bounds for the Cosine similarity. One is the upper bound proposed in [6], referred to as IL_{φ_1}. The other is our proposed upper bound without the multi-layer structure, referred to as IL_{φ_2}. Notably, the indexing structure in [87] is not compared because their method is proposed to reduce the IO cost which is not appropriate to be tested in a main memory setting.

All experiments are implemented on a desktop with Intel(R) Core (TM) i7-3770 CPU @3.40GHz processor and 8GB memory. The operating system is Windows 7 Enterprise (64-bit).



(a) Purity (b) Covereage

FIGURE 3.4: Tuning θ_U

TABLE 3.3: Data Statistics

			m=1		m=2		m=3	
# Tweets	# Events	Avg. Size	Avg. Length	# Terms	Avg. Length	# Terms	Avg. Length	# Terms
2 million	88,723	12.82	68.86	8,798	13.51	189,721	9.56	11,858
4 million	121,451	14.05	78.42	11,019	15.73	332,806	10.76	26,032
6 million	165,260	14.58	89.79	13,625	18.79	475,226	13.40	42,829
8 million	203,608	14.74	97.98	15,693	20.52	625,470	14.43	57,099
10 million	260,925	14.78	109.78	18,275	23.56	774,864	15.31	66,225

3.5.2 Tuning of Parameters

Two parameters need to be tuned in our method. They are θ_U for the event evolution monitoring and λ for the MIL index structure. The 439,880 tweets posted from July 25th to July 31st are used for the tuning of parameters and the remaining 10,681,232 tweets are used for testing.

θ_U controls the radius threshold for events to avoid the side effect of extremely large threshold values. The effect of θ_U on purity and coverage for different θ_U values in the range of 0.2 to 0.6 is shown in Figure 3.4. In Figure 3.4(a), the purity increases as θ_U goes up for different top-N hottest events. This is because the higher the threshold is, the less noise will be inserted into the clusters. In Figure 3.4(b), the coverage for event detection first increases when the threshold rises. It reaches its peak when θ_U=0.4 and then goes down. The reason is that when large events are split into small events due to the high threshold, the top few largest events cannot be correctly tracked anymore. In consideration of both coverage and purity, we set 0.4 as the default value for θ_U.

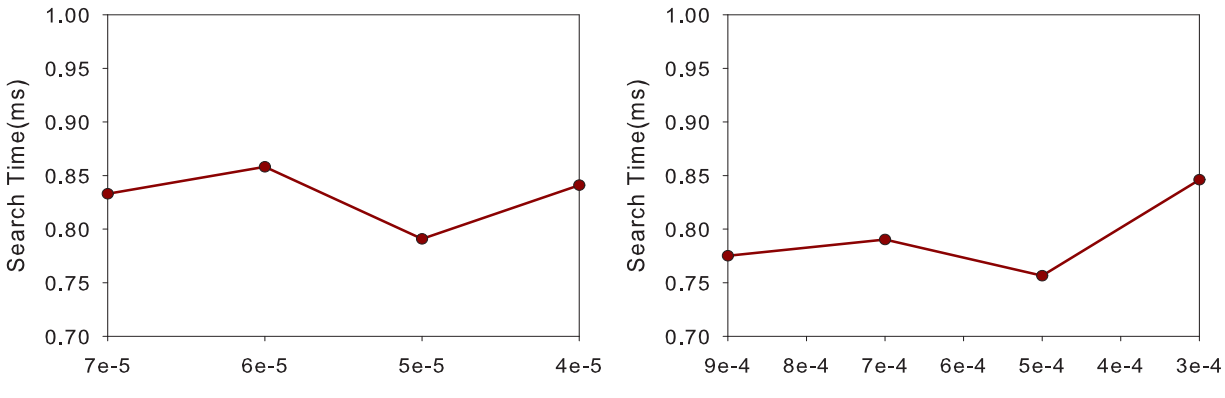
λ is the support threshold for FP-growth which is used to control the generation of m-terms at

each layer in MIL. The 1-terms are generated by wavelet analysis based on the goal to make more than 99% of tweets contain at least two words (after removing stop-words). This is because tweets containing just one word may lead to ambiguity or misunderstanding about its topic. From the 1-terms, we generate 2-terms by tuning λ to get the best performance of the 2-layer MIL. Similar idea is applied to tune λ for different layers in MIL. Figure 3.5 shows the results for m=2 and 3 respectively, from which we set $\lambda = 0.00005$ and 0.0005 to generate 2-terms and 3-terms respectively.

θ_U and λ are used to generate events and terms. Table 3.3 shows the statistics for five tweet datasets, where Avg. Size means the average number of tweets in each event, Avg. Length means the average length of the event lists at each layer of MIL, and # Terms means the number of terms at each layer. We do not show the results for $m > 3$, as their numbers of terms generated are very small and most tweets cannot reach those extremely low layers. As we can see, we have less than 20,000 1-terms (i.e., the vocabulary size) for the largest dataset. The average event list length for the first layer is mostly around 100. For the second layer, the number of 2-terms increases dramatically, while the average list length drops quickly to less than 24 even for the largest dataset. This is reasonable since there are many tweets that could share two common words. However, the possibility for tweets to share more than two words plunges, as indicated by the number of 3-terms in the table, and the average list length is further reduced to be around 10 or slightly more.

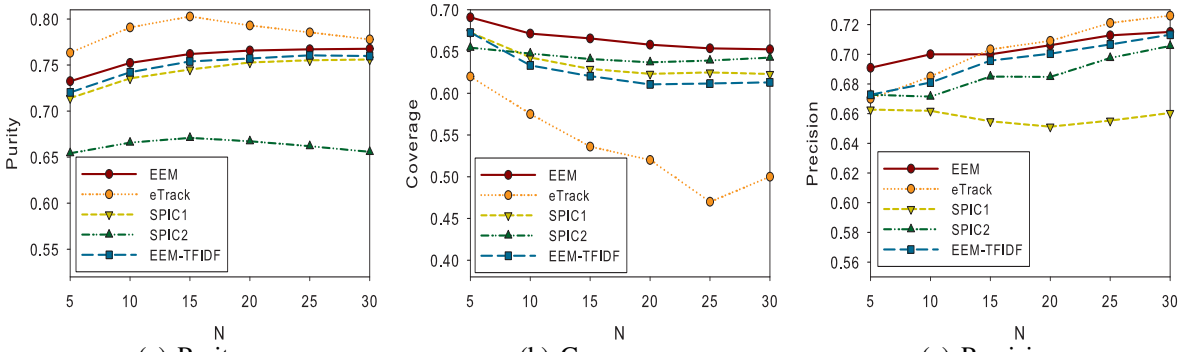
Comparing the index size of MIL with the traditional IL, i.e., the first layer of MIL, MIL consumes much larger space than IL. For the largest dataset, IL occupies about 8MB space, while MIL takes about 80MB. However, compared with the main memory size of 8GB, such space cost is rather small. Therefore, we assume all events and the index structure are fully maintained in main memory for the experiments. Despite the disadvantage of space requirement for MIL, we focus on its search performance improvements.

From the space consumption of different sizes of datasets in our experiments, we can extrapolate that for the entire tweet stream in a sliding window of two weeks (812 million tweets), MIL takes about 6.2 GB space. And quite a large percentage of events (e.g., 62% in our datasets) only contain one tweet. These events are created by the noisy tweets which are not relevant to any event. By filtering out these singleton events out of the sliding window, the space consumption will be further reduced. Hence our method is capable of processing the entire tweet stream in reality with a sliding



(a) $m = 2$ (b) $m = 3$

FIGURE 3.5: Tuning λ



(a) Purity (b) Coverage (c) Precision

FIGURE 3.6: Event Detection Performance

window of a fortnight under our current computer configurations.

3.5.3 Results on Event Evolution Monitoring

Our proposed method can detect events from social data stream and monitor their evolutions. In this experiment, we first test the performance of the event detection and then verify the performance of evolution monitoring. A special dataset named "Tweets-Lite" is used in this experiment, which consists of 500,634 tweets collected from August 1st, 2013 to August 15th, 2013.

Event Detection

In this part, different event detection methods are compared. The coverage and purity are reported for different N values from 5 to 30, where N means the top-N hottest events detected. For each

day, the top-N hottest events are monitored and the corresponding purity, precision and coverage are calculated. The average purity, precision and coverage for different N are reported as the final results.

As illustrated in Figure 3.6, our method EEM outperforms almost all the others in all the three performance indicators except for purity of eTrack. In general, SPIC1 is better in purity but worse in coverage. This is because the similarity threshold can be adjusted to make the cluster smaller so that the noise in an event is less. In this case, the purity will be high. However, big events will be separated into several similar small events. Hence the coverage will decrease since some biggest events are not accurately captured. EEM adopts the automatic threshold. It is also able to trigger the split or merge operation to have a good balance on event sizes and noises for event evolution monitoring. eTrack shows the highest purity but relatively low coverage. The high purity comes from their defined skeletal cluster which effectively removes the noises in the clusters. However, in their method, multiple recommended events correspond to the same ground truth event leading to the unsatisfactory coverage value. As for the algorithm of filtering words to generate the 1-terms, the wavelet analysis algorithm (EEM) outperforms the term frequency based algorithm (EEM-TFIDF) in all three measurements: purity, precision and coverage. This shows the effectiveness of the wavelet analysis. Moreover, we further check the words removed by wavelet analysis and find that around 3% of event-relevant keywords are incorrectly filtered. However, considering that there are 83% noisy words removed in total, which significantly saves the space and time consumption, the little sacrifice in accuracy is acceptable. Unlike in coverage, eTrack shows comparable performance to EEM in precision (Figure 3.6(c)). Comparing Figure 3.6(b) and Figure 3.6(c), we can find that around 70% events detected by eTrack are ground truth events, however 10% - 20% of them are similar (duplicate). As for EEM, the percentage of duplicate events is under 5%.

Evolution Monitoring

In this subsection, we focus on the performance study on the event evolution monitoring.

We first compare the performance of split and merge operations between the state-of-the-art evolution monitoring method eTrack and our method EEM. We manually check the evolution results of these two methods within the time window of two days. Moreover, SPIC1 is also inspected to further check the influence of split and merge operations on traditional single pass incremental clustering

TABLE 3.4: Event Evolution Statistics

	No. of			
Methods	Events	Split	Merge	Running Time (ms)
eTrack	459	75	6	285177
EEM	2465	74	17	84970
SPIC1	2412	-	-	81095

TABLE 3.5: Event Evolution Operations Case Study

Factors	Creation	Absorption	Split	Merge
No. of Occurrences	13792	485642	1200	104
Avg. Event Size	-	-	52.08	3.22

algorithm. Table 3.4 lists the statistics of event evolution operations. We can see that EEM generates more events than eTrack due to the skeletal cluster designed in eTrack. The Running Time shows the time cost on processing all the tweets within the two-day time window. The proposed EEM outperforms eTrack by reducing more than 70% time cost. Comparing the performance of EEM and SPIC1 shown in both Figure 3.6 and Table 3.4, it is obvious that split and merge operations improve the event evolution performance on purity, coverage and precision with only slight additional time cost.

To better understand the event evolution details of our EEM, we also conduct a case study on the dataset "Tweets-Lite" as shown in Table 3.5 and Figure 3.7. "No. of Occurrences" is the number of occurrences of each operation. The "Avg. Event Size" is the average size of the events before they are split or merged. From Table 3.5, we can see that most tweets are absorbed by an existing event. Generally, split operations happen on the large events while the merge operations happen on the small events. Only around 8% split operations are followed by a merge operation.

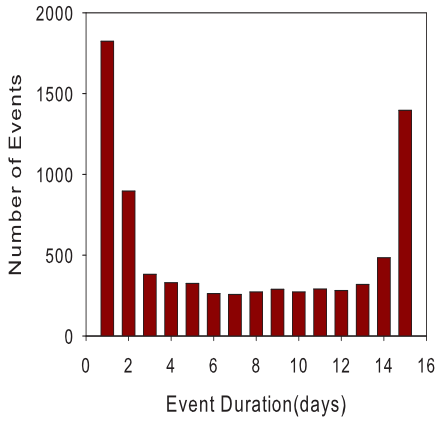


FIGURE 3.7: Distribution of Event Duration

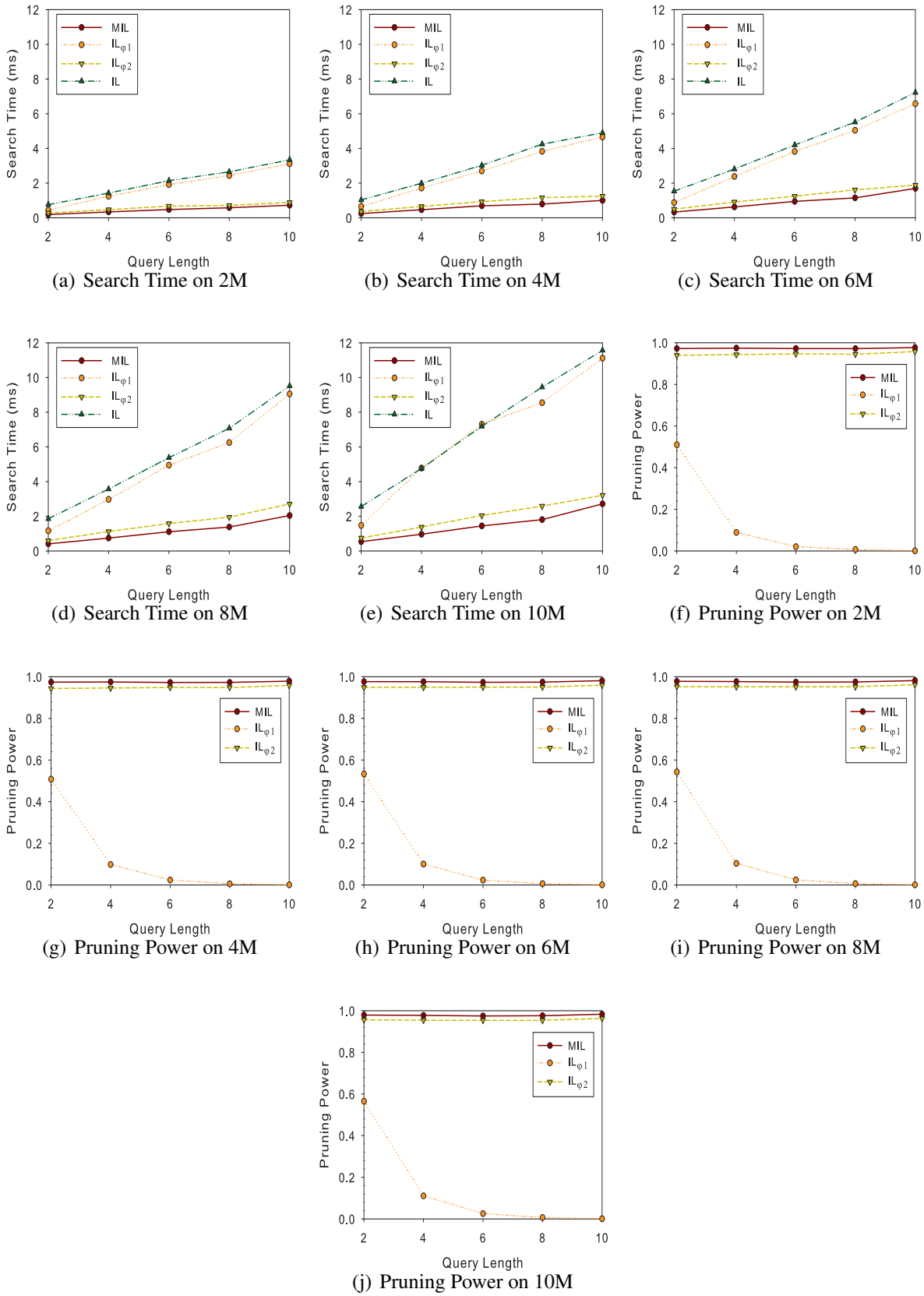
The distribution of the event durations is shown in Figure 3.7. We can observe that, 23.1% events last just one day, and this number decreases to half when we count the events that last for two days. 17.7% events are long-running events which are active all the time. For the other durations, the number of events stays stable.

3.5.4 Results on Indexing

In this subsection, we report the comparison results on the indexing performance for the nearest neighbour search. To show the effect of data size, we conduct experiments on the five datasets as mentioned in Section 3.5.1. Their events and terms are first generated and then indexed by MIL. The effect of the query length is also examined, given the range from 2 to 10.

The search time and the pruning power of different methods with different lengths of query posts on different sizes of datasets are shown in Figure 3.8. Figures 3.8(a)-3.8(e) show the results of search time for different index methods on different sizes of datasets. Clearly, MIL achieves the best search time and larger datasets lead to more search time for all methods. However, the increments for our method MIL are much smaller than the traditional IL. Equipped with two different upper bounds, IL can be further improved by IL_{φ_1} and IL_{φ_2}. Our proposed upper bound significantly outperforms the upper bound proposed in [6]. By applying the multi-layer structure, the search time can be further improved. As the query length increases, the superiority of MIL is better demonstrated. Figures 3.8(f)-3.8(j) show the pruning power of IL_{φ_1}, IL_{φ_2}, and MIL. IL does not prune any events from their full similarity computations. All datasets show similar trends. Our proposed upper bound is able to prune over 95% of the events and together with the multi-layer structure it can further increase the pruning power to be 98% for all different query lengths and data sizes, showing the strong robustness of MIL. $\text{IL}_{\varphi 1}$ can achieve less than 60% pruning power and its capability drops quickly as query length increases. The pruning power is close to zero when the query length reaches 8.

We also test the index update time whenever there is an update on events as described in Section 3.4.2. As shown in Figure 3.9, the index update time is minor compared to the search time. This is because in addition to the time spent on retrieving events list from MIL, search time also includes the time on upper bound calculation and a small number of similarity calculations, while the index



(a) Search Time on 2M (b) Search Time on 4M (c) Search Time on 6M

(d) Search Time on 8M (e) Search Time on 10M (f) Pruning Power on 2M

(g) Pruning Power on 4M (h) Pruning Power on 6M (i) Pruning Power on 8M

(j) Pruning Power on 10M

FIGURE 3.8: Effect of Query Length and Data Size for Nearest Neighbour Search

TABLE 3.6: Index Maintenance Time After Four Event Operations

Case	Creation (ms)	Absorption (ms)	Split (ms)	Merge (ms)
Average	0.003	0.188	0.020	0.007
Worst	0.497	49.580	0.700	0.620

TABLE 3.7: Running Time Analysis

Case	Nearest Neighbour Search (ms)	Split (ms)	Merge (ms)
Average	1.754	23.857	6.917
Worst	273.732	124.350	468.523

update time only includes the time spent on updating events in MIL. As the data size increases, the index update time grows as well. Due to the multi-layer traversal, MIL takes slightly more time than the one layer IL. However, compared with the significant gain in performance of nearest neighbour search, additional cost for index update is acceptable.

We further test the average and maximum index maintenance time after each of the four event operations on dataset "Tweets-Lite" and the results are listed in Table 3.6. The time cost on the index maintenance depends on the number of words contained in the event that is to be updated in MIL. For creation, the new event contains just one post. From Table 3.5 we can see that the event size is also small for merge. Hence the average index maintenance times for creation and merge are much smaller than for the other two operations. The large number of words in events will lead to a considerable increase in index maintenance time. That's why absorption takes significantly longer time than others to maintain MIL. For the worst case, index maintenance time after creation is the smallest among the four operations because the event to be updated in the indexing structure only contains one post (the number of words is limited).

To better understand the four event operations in the proposed framework, a running time analysis is conducted for these operations. Note that for creation and absorption, the processing time basically comes from the nearest neighbour search. Hence in Table 3.7 the average and maximum processing time of the nearest neighbour search, split and merge are listed. Theoretically, nearest neighbour search, split (bisecting K-means) and merge all have a linear time complexity (linear in the number of the events retrieved from MIL, linear in the number of posts contained in the split event and linear in the the number of merge candidate events retrieved from MIL respectively). From Table 3.7 we can see that, on average nearest neighbour search takes the least time, and split takes the most.

This is because split (bisecting K-means) makes multiple iterations on each run while the other two operations are just one-pass. In the worst case, merge is the slowest operation. The reason is that, after merging with each single candidate event, the radius of the newly merged event will be calculated to check if the stop condition is met. This will be time-consuming if the merge candidate list is long.

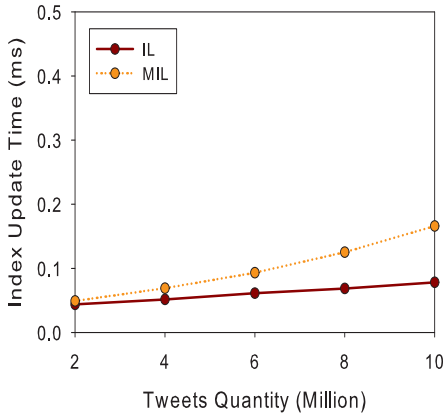


FIGURE 3.9: Effect of Data Size on Index Update Time

3.6 Summary

In this chapter, we have presented a novel event monitoring method along with a multi-layer inverted indexing structure to efficiently and effectively index evolving events from social textual data streams. Four operations are designed to capture the dynamics of events over time. A multi-layer structure, MIL, is proposed to improve the performance of the traditional inverted file and used to accelerate the event evolution monitoring process. Extensive experiments are conducted on real-world tweet datasets to verify the novelty of our methods. Althougth the proposed MIL has been proven to be efficient in both the nearest neighbour search process and index maintainance process, there exist a few limitations. First, the space consumption of the proposed MIL is much larger than the traditional inverted list indexing structure due to its multi-layer structure as mentioned in Section 3.5.2. Given that the big data problem is inevitable in the social data mining area, how to reduce the space compexity of MIL is an interesting direction for the future work. One solution is to apply a time decay function on the detected events so as to move the faded events from memory to hard disk. More details are provided in Section 6.2.2. Second, MIL is designed for the short documents like tweets

and not good at processing long documents. When the lengths of the documents increase, the number of layers of MIL will go up as well to match the longer frequent words sets. this will lead to a larger space consumption.

Chapter 4

Multi-feature Event Modeling: Detection, Tracking and Visualization

4.1 Introduction

In Chapter 3, the events are detected and monitored based on textual feature only. However, in addition to text contents, social data consists of various different features such as image, location, timestamp and so on. These features can not only provide supplemental information that facilitates the event detection model, but also describe the detected events from different aspects with more vivid visualizations. Hence, how to make the most of different features to improve the performance of social event detection and visualization is of great importance. Nevertheless, existing approaches do not take advantage of these features. Take Twitter as an example, around 36 percent of the total tweets contain images [28]. Most event detection methods tend to represent an image as a bag-of-visual-words and then process these visual words in the same way as textual words. This may not fully exploit the visual properties of images. In this chapter, we thoroughly study the impact of images on social event detection for different event categories using various visual features. A novel topic model which jointly models five features (text, image, location, timestamp and hashtag) is designed to discover events from the sheer amount of social data. Moreover, the evolutions of events are tracked by linking the events detected on adjacent days and each event is visualized by representative images selected on three predefined criteria. The framework is illustrated in Figure 4.1.

Spatio-temporal multimodal LDA is first applied for event detection from the collected social data, followed by adopting the maximum-weighted bipartite graph matching for event evolution tracking. Finally, the representative images are selected for event visualization. Extensive experiments have been conducted on a real-world tweet dataset to verify the effectiveness of our method.

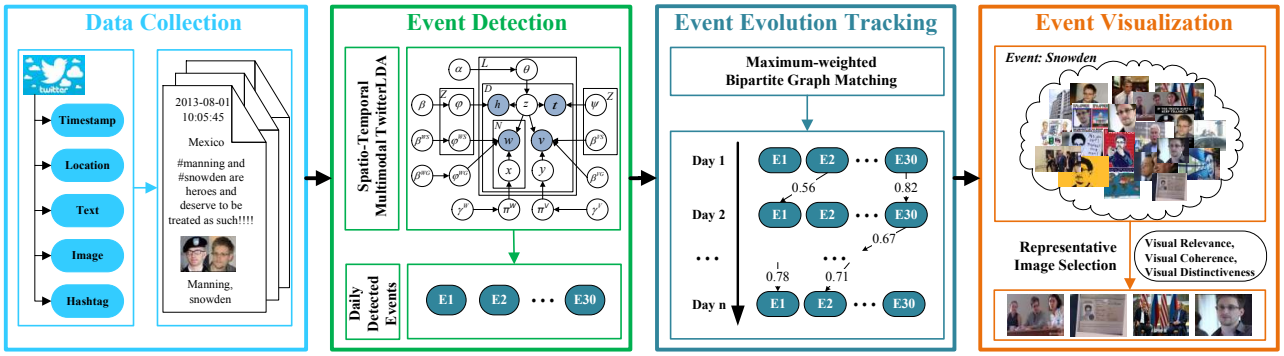


FIGURE 4.1: The Proposed Framework for Social Event Detection, Tracking and Visualization.

The remainder of this chapter is organized as follows. Our proposed social event detection model is presented in Section 4.2, followed by the event evolution tracking and visualization approach in Section 4.3. The experimental results are shown in Section 4.4 and Section 4.5 concludes the chapter.

4.2 Social Event Detection

In this section, we first formally define some concepts in this work and then present the proposed STM-LDA model for event detection from social data.

4.2.1 Notations and Definitions

Table 5.1 summarizes the notations used in this work for our proposed topic model and the corresponding descriptions.

Post. A post is a five-tuple (d, v, t, h, l) that denotes a post is posted with text d, image v and hashtag h at time t in location l. The scale of the location is adjustable, which could be a country, a state or even a user-defined area. It is set as country in this work.

Word Type. The words from posts are categorized into three types: the stop word, the general word and the specific word. The stop words are a set of predefined words to be removed in the

Notation	Description
L	Total number of locations
D_l	Total number of posts in location l
$N_{l,d}$, $C_{l,d}$	Total number of words and hashtags in l's d-th post
M	Dimensionality of the visual feature
Z, W, H	Total number of topics, words, and hashtags
z, w, h	Label for topic, word, and hashtag
v, t	Visual feature vector and timestamp
x, y	Indicator of general or specific for word and image
φ^{WG}, β^{VG}	General word and image distribution
φ^{WS}, β^{VS}	Specific word and image distribution
φ	Hashtag distribution
θ	Location-specific topic distribution
π^W, π^V, ψ	Bernoulli and Beta distribution
β, β^{WG}, β^{WS}, α, γ^W, γ^V	Dirichlet priors

TABLE 4.1: Notation for Social Event Tracking and Visualization

data pre-process phase, for example "a", "we", "haha" etc. The general words are the noisy words frequently and uniformly appearing in many events and thus not useful for event detection. The specific words are the distinguishable words for describing different events. The general words and specific words are automatically recognized by our topic model.

Image Type. Three types of images are defined, namely, the stop image, the general image and the specific image. The definition of each image type is similar to the corresponding word type. The only difference is that stop images are defined as a set of images belonging to some predefined categories since we cannot list all the exact images to be filtered in the data pre-process step. The stop images must meet two requirements: they are visually similar to each other when they appear in different events, and they must be easily recognized by certain methods. After analysing the images in social data, we define the following four categories including cartoon, landscape, diagram and the screenshot of text as the stop images categories and use an SVM classifier to identify them. The general images are the noisy images which bring negative effect on event detection. The noises could be caused by the users (the irrelevant images) or the characteristics of images (the unidentifiable images for events). The specific images are meant to describe the event visually. Only the specific images are used for event detection.

Topic. Given a collection of posts, a topic z generated by our topic model is a combination of a multinomial word distribution φ_z^{WS}, a multinomial hashtag distribution φ_z, a Beta timestamp

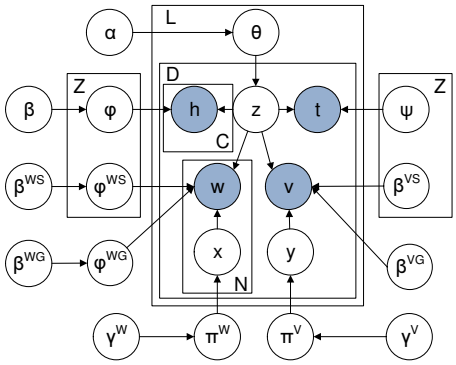


FIGURE 4.2: Graphical Model of STM-LDA

distribution ψ_z, and a Dirichlet image distribution β_z^{VS}. All the above four distributions are location-specific.

Event. An event is a set of posts which are semantically coherent to each other. Specifically, an event is defined as a group of posts labelled as the same topic z by the topic model.

4.2.2 Spatio-Temporal Multimodal LDA

Suppose we are given a continuous social data stream as the input, and each post consists of five features: text, image, timestamp, location and hashtag. The objective of this work is to model the topics latent in the social data by exploiting different social data features so as to detect events from these posts automatically. Afterwards, the evolutions of these events along the timeline are tracked and each event is visualized using the summarized information and the representative images. In this work, STM-LDA is proposed as a novel topic model for social event detection. Its graphical model is shown in Figure 4.2.

Formally, we assume that there are Z topics in total in social data. In contrast to the original LDA, each post is only assigned to one topic in our model. Unlike TwitterLDA which presumes that each single user has a specific topic distribution, we assume that users in the same location share the same topics of interest (considering that these users usually share the similar cultural background and live in the same environment), hence the posts collected from the same location share one topic distribution θ, i.e., a location-specific topic distribution. Our assumption is more applicable for the general event detection because the posts from each single user are very sparse in the whole posts collection. In our model, words (images) are divided into general words (images) and specific words (images). Two

latent random variables x and y work as the general/specific indicators for words and images. Each topic z in our model has a mixture of four distributions decided by four social data features: a multinomial hashtag distribution φ_z, a multinomial distribution over specific words φ_z^{WS}, a Beta timestamp distribution ψ_z, and a Dirichlet distribution over specific images β_z^{VS}. Note that instead of representing images as bag-of-visual-words and sampling visual words using the Dirichlet-multinomial distribution as in existing image-involved topic models, we adopt the state-of-the-art CNN feature to represent images and model the normalized CNN feature by the Dirichlet distribution. As shown in Figure 4.2, there are three distributions not conditioned on the topic z: the Dirichlet-multinomial distributions over general words β^{WG} and φ^{WG}, and the Dirichlet distribution over general images β^{VG}. These are the topic-independent distributions shared by the general words and images because they do not belong to any specific topics.

The proposed STM-LDA is different from TwitterLDA mainly from two aspects. First, STM-LDA models topics as location-specific distributions while TwitterLDA uses user-specific distributions. Second, STM-LDA utilizes multiple social data features including location, time, text, hashtag, and image, while TwitterLDA considers user and text only.

4.2.3 The Generative Process

When a post is generated, a topic is first assigned based on the topic distribution from the user's location. Next, given the topic, the word(s), image, hashtag(s), timestamp are then generated based on their own distributions. Notably, when the words and images are generated, they are selected from either the general or the specific sets of words and images. The generative process for all posts is as follows, where Dir() and Multi() represent Dirichlet and multinomial distributions respectively.

1. Draw $\varphi^{WG} \sim \text{Dir}(\beta^{WG})$ indicating the general word distribution. Draw $\pi^W \sim \text{Dir}(\gamma^W)$ and $\pi^V \sim \text{Dir}(\gamma^V)$ denoting the Bernoulli distributions that determine the selections between the general words and specific words, and between the general images and specific images.

2. For each topic $z = 1, \cdots, Z$, draw $\varphi_z^{WS} \sim \text{Dir}(\beta_z^{WS})$ and $\varphi_z \sim \text{Dir}(\beta_z)$, denoting the specific word and hashtag distribution for topic z.

3. For each location $l = 1, \cdots, L$

(a) Draw $\theta_l \sim \text{Dir}(\alpha)$, indicating distribution of topics over the posts collection in location l.

(b) For the d-th post in location l, $d = 1, \cdots, D_l$

i. Draw $z_{l,d} \sim \text{Multi}(\theta_l)$, corresponding to the topic assigned for each post.

ii. For the n-th word in the post, $n = 1, \cdots, N_{l,d}$

- Draw a variable $x_{l,d,n} \sim$ Bernoulli(π^W) as an indicator for general or specific word;
- Draw $w_{l,d,n} \sim$ Multi (φ^{WG}) if $x_{l,d,n}$ = 0, and $w_{l,d,n} \sim$ Multi ($\varphi^{WS}_{z_{l,d}}$) if $x_{l,d,n}$ = 1.

iii. For the c-th hashtag of the post, $c = 1, \cdots, C_{l,d}$

- Draw $h_{l,d,c} \sim$ Multi ($\varphi_{z_{l,d}}$), a Multinomial distribution conditioned on the topic $z_{l,d}$.

iv. Draw a timestamp $t_{l,d} \sim$ Beta($\psi_{z_{l,d}}$), a Beta distribution conditioned on the topic $z_{l,d}$.

v. Draw a variable $y_{l,d} \sim$ Bernoulli(π^V) as an indicator for general or specific image.

vi. Draw $v_{l,d} \sim$ Dir(β^{VG}) if $y_{l,d}$ = 0, and $v_{l,d} \sim$ Dir ($\beta^{VS}_{z_{l,d}}$) if $y_{l,d}$ = 1.

4.2.4 Inference

In our proposed STM-LDA, there are three latent variables to be inferred from the observed social data features $\{w, v, t, h\}$, including the topic z, the general-specific (word and image) indicators x and y. Collapsed Gibbs sampling [27] is adopted as in other topic models for the approximate model inference to obtain these latent variables due to its effectiveness and simplicity.

Sampling $z_{l,d}$

We firstly sample the topic label for the d-th post in location l. Let i denote $\{l, d\}$, we use the following equation to sample z_i:

$$p(z_i|\boldsymbol{Z}_{\neg i}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}) \propto \frac{p(\boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}|\ominus)}{p(\boldsymbol{Z}_{\neg i}, \boldsymbol{W}_{\neg i}, \boldsymbol{V}_{\neg i}, \boldsymbol{X}_{\neg i}, \boldsymbol{Y}_{\neg i}, \boldsymbol{T}_{\neg i}, \boldsymbol{H}_{\neg i}|\ominus)} \tag{4.1}$$

where $\ominus$ is the set of all the parameters.

To derive the above conditional probability, we start by computing the following joint probability:

$$
\begin{aligned}
&p(\boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}|\ominus) \\
&= p(\boldsymbol{W}|\boldsymbol{Z},\boldsymbol{X},\beta^{WG},\beta^{WS})p(\boldsymbol{X}|\gamma^{W})p(\boldsymbol{V}|\boldsymbol{Z},\boldsymbol{Y},\beta^{VG},\beta^{VS})p(\boldsymbol{Y}|\gamma^{V})p(\boldsymbol{T}|\boldsymbol{Z},\psi)p(\boldsymbol{H}|\boldsymbol{Z},\beta)p(\boldsymbol{Z}|\alpha) \\
&= \frac{\Delta(n^{\boldsymbol{W}}_{x=0}+\beta^{WG})}{\Delta(\beta^{WG})} \prod_{z=1}^{Z} \frac{\Delta(n^{\boldsymbol{W}}_{z,x=1}+\beta^{WS})}{\Delta(\beta^{WS})} \frac{\Delta(n^{\boldsymbol{X}}_{(.)}+\gamma^{W})}{\Delta(\gamma^{W})} \prod_{l=1}^{L}\prod_{d=1}^{D} p(v_i|\beta^{VG}) \prod_{l=1}^{L}\prod_{d=1}^{D} p(v_i|\beta^{VS}_{z_i}) \frac{\Delta(n^{\boldsymbol{Y}}_{(.)}+\gamma^{V})}{\Delta(\gamma^{V})} \\
&\quad \times \prod_{l=1}^{L}\prod_{d=1}^{D} p(t_i|\psi_{z_i}) \prod_{z=1}^{Z} \frac{\Delta(n^{\boldsymbol{H}}_{z}+\beta)}{\Delta(\beta)} \prod_{l=1}^{L} \frac{\Delta(n^{\boldsymbol{Z}}_{l}+\alpha)}{\Delta(\alpha)}
\end{aligned}
\tag{4.2}
$$

where $n^{\boldsymbol{W}}_{x=0}$ and $n^{\boldsymbol{W}}_{z,x=1}$ denote two W-dimensional vectors in which each value is the number of times every word is sampled as general word and the number of times each word is sampled as specific word with topic z respectively. $n^{\boldsymbol{X}}_{(.)}$ is a 2-dimensional vector denoting the number of times $x = 0$ and $x = 1$ occurs respectively. Similarly, $n^{\boldsymbol{Y}}_{(.)}$ is the counter for y. $n^{\boldsymbol{H}}_{z}$ is a H-dimensional vector and each value in it denotes the number of times each hashtag is sampled as topic z. $n^{\boldsymbol{Z}}_{l}$ is a Z-dimensional vector in which each element denotes the number of times that each topic occurs in the posts from location l. v_i and t_i represent the normalized visual feature vector and the timestamp of the d-th post in location l.

Given Equation 4.1 and Equation 4.2, the probability of assigning a topic z to z_i can be estimated as follows:

$$
\begin{aligned}
&p(z_i = z|\boldsymbol{Z}_{\neg i}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}) \\
&\propto \frac{\prod_{w=1}^{W}\prod_{p=1}^{n^w_i}(n^w_{z,x=1}+\beta^{WS}-p)}{\prod_{q=1}^{N_i}(\sum_{w=1}^{W} n^w_{z,x=1}+W\beta^{WS}-q)} \frac{\prod_{m=1}^{M} v_{i,m}^{\beta^{VS}_{z,m}-1}}{\Delta(\beta^{VS}_z)} \frac{(1-t_i)^{\psi_{z1}-1}t_i^{\psi_{z2}-1}}{B(\psi_{z1},\psi_{z2})} \frac{\prod_{h=1}^{H}\prod_{p=1}^{n^h_i}(n^h_z+\beta-p)}{\prod_{q=1}^{C_i}(\sum_{h=1}^{H} n^h_z+H\beta-q)} \frac{n^z_{l,\neg i}+\alpha}{\sum_{z=1}^{Z} n^z_l+T\alpha-1}
\end{aligned}
\tag{4.3}
$$

where N_i and C_i denote the number of words and hashtags occurs in the d-th post from location l. n^w_i and n^h_i denote the number of times word w and hashtag h occur in the d-th post from location l. $n^w_{z,x=1}$ and n^h_z are the number of times specific word w and hashtag h is sampled as topic z. $n^z_{l,\neg i}$ is the number of times topic z occurs in the posts from location l not including the d-th post. t_i and $v_{i,m}$ are the timestamp and the m-th dimension of the visual feature vector for the d-th post from location l.

Sampling $x_{l,d,n}$ and $y_{l,d}$

Next, we sample the general-specific indicators x and y. Let i be $\{l, d\}$ and j be $\{l, d, n\}$, given Equation 4.2, x_j is sampled as follows:

$$\begin{aligned} p(x_j|\boldsymbol{X}_{\neg j}, \boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}) &\propto \frac{p(\boldsymbol{Z},\boldsymbol{W},\boldsymbol{V},\boldsymbol{X},\boldsymbol{Y},\boldsymbol{T},\boldsymbol{H}|\ominus)}{p(\boldsymbol{Z},\boldsymbol{W}_{\neg j},\boldsymbol{V},\boldsymbol{X}_{\neg j},\boldsymbol{Y},\boldsymbol{T},\boldsymbol{H}|\ominus)} \\ &\propto \frac{\Delta(n^{\boldsymbol{W}}_{x=0}+\beta^{WG})}{\Delta(n^{\boldsymbol{W}}_{x=0,\neg j}+\beta^{WG})} \prod_{z=1}^{Z} \frac{\Delta(n^{\boldsymbol{W}}_{z,x=1}+\beta^{WS})}{\Delta(n^{\boldsymbol{W}}_{z,x=1,\neg j}+\beta^{WS})} \frac{\Delta(n^{\boldsymbol{X}}_{(.)}+\gamma^{W})}{\Delta(n^{\boldsymbol{X}}_{\neg j}+\gamma^{W})} \end{aligned} \tag{4.4}$$

Accordingly, the probability of assigning a binary label 1 to x_j as a specific word indicator is estimated as below:

$$p(x_j = 1|\boldsymbol{X}_{\neg j}, \boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}) \propto \frac{n^{x=1}_{\neg j}+\gamma^{W}}{\sum_{x=0}^{1} n^{x}_{\neg j}+2\gamma^{W}} \frac{n^{w_j}_{z_i,x=1,\neg j}+\beta^{WS}}{\sum_{w=1}^{W} n^{w}_{z_i,x=1,\neg j}+W\beta^{WS}} \tag{4.5}$$

where $n^{x=1}_{\neg j}$ denotes the number of times $x = 1$ occurs for the posts set not including the n-th word in the d-th post from location l. $n^{w_j}_{z_i,x=1,\neg j}$ denotes the number of times the specific word w_j is sampled as topic z_i for the posts set not including the n-th word in the d-th post from location l.

The probability of assigning a binary label 0 to x_j is similar to Equation 4.5 by replacing β^{WS}, $n^{x=1}_{\neg j}$ and $n^{w_j}_{z_i,x=1,\neg j}$ with β^{WG}, $n^{x=0}_{\neg j}$ and $n^{w_j}_{x=0,\neg j}$ correspondingly. $n^{x=0}_{\neg j}$ and $n^{w_j}_{x=0,\neg j}$ denotes the number of times $x = 0$ occurs and the number of times word w_j is sampled with label 0 for the posts set not including the n-th word in the d-th post from location l.

Next we will introduce the process of sampling $y_{l,d}$. Given Equation 4.2, to sample y_i, the following equation is used:

$$\begin{aligned} p(y_i|\boldsymbol{Y}_{\neg i}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{T}, \boldsymbol{H}) &\propto \frac{p(\boldsymbol{Z},\boldsymbol{W},\boldsymbol{V},\boldsymbol{X},\boldsymbol{Y},\boldsymbol{T},\boldsymbol{H}|\ominus)}{p(\boldsymbol{Z}_{\neg i},\boldsymbol{W}_{\neg i},\boldsymbol{V}_{\neg i},\boldsymbol{X}_{\neg i},\boldsymbol{Y}_{\neg i},\boldsymbol{T}_{\neg i},\boldsymbol{H}_{\neg i}|\ominus)} \\ &\propto p(v_i|\beta^{VG}) \prod_{z=1}^{Z} p(v_i|\beta^{VS}_{z}) \frac{\Delta(n^{\boldsymbol{Y}}_{(.)}+\gamma^{V})}{\Delta(\gamma^{V})} \end{aligned} \tag{4.6}$$

Accordingly, the probability of assigning a binary label 1 to y_i is estimated as below:

$$p(y_i=1|\boldsymbol{Y}_{\neg i}, \boldsymbol{Z}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{Y}, \boldsymbol{T}, \boldsymbol{H}) \propto \frac{n^{y=1}_{\neg i}+\gamma^{V}}{\sum_{y=0}^{1} n^{y}_{\neg i}+2\gamma^{V}} \frac{\prod_{m=1}^{M} v_{i,m}^{\beta^{VS}_{z_i,m}-1}}{\Delta(\beta^{VS}_{z_i})} \tag{4.7}$$

where $n_{\neg i}^{y=1}$ denotes the number of times $y = 1$ occurs for the posts set not including the d-th post from location l.

Similarly, the probability of assigning a binary label 0 to y_i can be derived by using β^{VG}, $n_{\neg i}^{y=0}$ corresponding to $\beta_{z_i}^{VS}$ and $n_{\neg i}^{y=1}$ in Equation 4.7. $n_{\neg i}^{y=0}$ denotes the number of times $y = 0$ occurs for the posts set not including the d-th post from location l.

4.2.5 Parameter Estimation

We use the following variational EM [21] procedure to estimate the parameters of our STM-LDA model.

1. (E-step) For each post, use Gibbs sampling as described before to find the optimizing values of the variational parameters θ, φ^{WS}, φ^{WG}, φ, π^{W} and π^{V}.

2. (M-step) Maximize the joint likelihood of the model with respect to the model parameters α, ψ, γ^{W}, γ^{V},β, β^{WG}, β^{WS},β^{VS} and β^{VG} given the variational parameters fixed as computed in the E-step.

Notably, the parameters of some well-studied Dirichlet distribution are set fixed as $\alpha = 50/T$, $\beta^{WG} = \beta^{WS} = 0.1$ for simplicity as in existing work [99]. The other variational parameters updated after each Gibbs sampling iteration are estimated as following [1].

[1]The estimation of ψ_z utilizes the method of moments.

$$\begin{cases} \psi_{z1} = \bar{t}_z(\frac{\bar{t}_z(1-\bar{t}_z)}{s_z^2} - 1), \\ \psi_{z2} = (1-\bar{t}_z)(\frac{\bar{t}_z(1-\bar{t}_z)}{s_z^2} - 1), \\ \theta_{l,z} = \frac{n_l^z + \alpha}{\sum_{z=1}^{Z} n_l^z + Z\alpha}, \\ \varphi_w^{WG} = \frac{n_{(.)}^w + \beta^{WG}}{\sum_{w=1}^{W} n_{(.)}^w + W\beta^{WG}}, \\ \varphi_{z,w}^{WS} = \frac{n_z^w + \beta^{WS}}{\sum_{w=1}^{W} n_z^w + W\beta^{WS}}, \\ \pi_x^W = \frac{n_{(.)}^x + \gamma^W}{\sum_{x=0}^{1} n_{(.)}^x + 2\gamma^W}, \\ \varphi_{z,h} = \frac{n_z^h + \beta}{\sum_{h=1}^{H} n_z^h + H\beta}, \end{cases} \tag{4.8}$$

where $\bar{t}_z$ and s_z^2 are the sample mean and the biased sample variance of the timestamps belong to topic z respectively. Note that these two values can be maintained incrementally by their zeroth, first and second order moments (denoting as M_0, M_1 and M_2) as $\bar{t}_z = M_1/M_0$ and $s_z^2 = M_2/M_0 - (M_1/M_0)^2$. n_l^z is the number of times topic z is sampled for the posts from location l. $n_{(.)}^w$ is the number of times word w is sampled. n_z^w and n_z^h are the number of times w and h are sampled given the topic z. $n_{(.)}^x$ is the number of times label x is sampled. The estimation of π_y^V is similar to π_x^W, with counting $n_{(.)}^y$ as the number of times label y is sampled.

To estimate the parameter β^{VS} and β^{VG}, the Newton-Raphson method is applied due to its linear time performance in computing the inverse of Hessian matrix [64]. The update rule for parameter β^{VG} is as follows:

$$\beta_m^{VG\,new} = \beta_m^{VG\,old} - (\boldsymbol{H^{-1}g})_m = \beta_m^{VG\,old} - \frac{g_m - b}{q_{m,m}} \tag{4.9}$$

where $b = \frac{\sum_{m=1}^{|M|}(g_m/q_{m,m})}{1/(N^{VG}\Psi'(\sum_{m=1}^{|M|}\beta_m^{VG}))}$ and $g_m = N^{VG}(\Psi(\sum_{m=1}^{|V|}\beta_m^{VG}) - \beta_m^{VG}) + \sum_{i \in General\text{Image}} v_{i,m}$. N^{VG} is the number of general images, Ψ, Ψ' and δ denote the digamma function, the trigamma function and the delta function. The update rule for β^{VS} is similar, with N_z^{VS} being the number of images sampled as topic z.

4.3 Event Tracking and Visualization

The social events are dynamically changing over time, hence tracking the event evolution is no less important than detecting events. Besides, event visualization is also of great importance for providing users a vivid and intuitive presentation of an event.

4.3.1 Event Evolution Tracking

Inspired by [55], the maximum-weighted bipartite graph matching (MWBGM) is applied for tracking the evolution of events in consecutive days. Formally speaking, a bipartite graph $G = (V, E)$ is constructed with two sets of nodes which correspond to the two sets of events detected in two successive days. The edge between two nodes represents the relevance between two events. In this work, each event is represented as a set of words, among which each word has a value showing its probability of being assigned to this event. The event relevance score between two events (i.e., the weight of the edge in the graph) is then defined as $event\ relevance = \sum_{w \in I} p(w) / \sum_{w \in U} p(w)$, where I represents the intersection set of words in the two events and U denotes the union word set. $p(w)$ is the maximum probability value of the word w being assigned to the two events. The objective of MWBGM is to find a subset of the events from the graph so that each event is only linked to at most one event in the adjacent day and the multiplication of the weights between the final event pairs is maximized. The linkages between those events in the final event pairs indicate the evolutions of these events. For instance, event 1 is detected on day 1 and event 2 is detected on the next day. If event 1 and event 2 form an event pair after applying MWBGM, it means that event 1 evolves into event 2 on the second day. Finally, the event evolutions over time are tracked by the event chains formed from the individual event pairs. In this work, the MWBGM problem is addressed by the Kuhn-Munkres algorithm [65] which estimates the maximum sum of the log probabilities of the final event pairs' weights.

4.3.2 Event Visualization

In addition to providing supplemental information for event detection, images can be further used for event visualization since they describe the event in a much more straightforward way. How to select

the most representative images for an event from a large number of disordered candidates is the key for event visualization.

Representative Image Selection

Three aspects are considered when choosing the representative images to visualize a target event: visual relevance, visual coherence and distinctiveness.

Visual Relevance When discussing an event, a same image might be posted by different users. The more times an image is posted, the more relevant it is to the event. In other words, the visual relevance $P_r(I_k)$ is determined by the occurrence number N_k of an image I_k. Therefore the visual relevance $P_r(I_k)$ is defined as $P_r(I_k) = \frac{N_k}{M}$, where M is the total number of images in the images set of the event.

Visual Coherence An image with more near neighbours is more coherent to the event and should have a higher probability to be the representative image. Visual coherence is defined under the above assumption as in [93] and is calculated by a random walk process [35] in this work. Each node I_k represents an image. There exists an edge $e_{k,l}$ between I_k and I_l if I_l is one of the K-nearest neighbours of I_k. The weight of the edge $e_{k,l}$ is calculated as the visual similarity $Sim(I_k, I_l)$ divided by the sum of visual similarities among I_k and all its K-nearest neighbours. Visual coherence $P_c(I_k)$ is formulated as $P_c(I_k) = (\alpha E + \frac{(1-\alpha)}{M}\Lambda^T)P_c(I_k)$, where E is the transition matrix constructed by $e_{i,j}$, and Λ is a column vector with the i-th element being the textual similarity between the post that image I_i belongs to and the target event. M is the total number of images in the image set of the event. The optimal solution is the eigenvector of $(\alpha E + \frac{(1-\alpha)}{M}\Lambda^T)$ with the largest eigenvalue. Notably, in [93] the initial scores for random walk are set to be equal for all the images while we define the initial score as the textual score of the post that each image belongs to for boost.

Distinctiveness Intuitively, a set of images with diverse contents are more informative than images with duplicate contents. This makes distinctiveness an important criterion for representative image selection. The visual relevance and visual coherence score are summed up (i.e., $P_r(I_k) + P_c(I_k)$) to get a representative image candidates set containing the more relevant images. Suppose a total of K representative images are needed. We firstly select the most representative image as the one with the largest sum of visual relevance and coherence value. After that, the remaining $K - 1$ representative

images are selected from the image candidates set of the event one by one. The remaining images must carry diverse contents in order to describe the event from different perspectives. In [40], the bipartite graph hitting time is applied to personalized query suggestion with diversity awareness. Similarly, to achieve the representative images diversity, we construct a bipartite graph $G = (V, E)$ by two sets of nodes $V = V_1 \cup V_2$. One set (V_2) is the images which have already been selected as the representative images and the other set (V_1) contains all the rest images. A random walk is performed on this bipartite graph where the transition probability is defined as $p_{ij} = \sum_{k \in V_2} \frac{w(i,k)}{d_i} \frac{w(k,j)}{d_k}$, and $d_i = \sum_{k \in V_2} w(i,k)$.

The following linear system is obtained for computing the hitting time [63]:

$$\begin{cases} h_i = 0, & \text{if} i \in V_2 \\ h_i = 1 + \sum_{j \notin V_2} p_{ij} h_j, & \text{if} i \notin V_2 \end{cases} \tag{4.10}$$

The hitting time of an image in the image set V_1 is the number of steps before the random walk first reaches the selected image set V_2. At each iteration, the image in the set V_1 with the largest hitting time is chosen as the next representative image and moved from V_1 to the selected image set V_2. This avoids picking the near neighbours of those already selected images in V_2, leading to the diversity of the representative images in V_2. Finally, a list of K diverse representative images is generated.

4.4 Experiments

In this section, we experimentally evaluate the performance of our proposed STM-LDA model on a large real-world tweet dataset. The experiments are designed to verify the effectiveness of the three main components of our framework: social event detection, event evolution tracking and event visualization.

4.4.1 Dataset

Our experiments are conducted on a large real-world dataset which consists of 10,681,232 tweets posted from August to November in 2013. The tweets are collected by Twitter Search API given a list of search terms. The reason we choose Search API instead of Stream API is that the latter one only

Vocabulary size	569741
Number of tweets	10681232
Number of images	2130562
Number of stop words	816
Number of stop images	594829
Number of countries	16
Average number of words per tweet	5
Percentage of tweets with hashtag	48%
Percentage of tweets with latitude/ longitude	7%

TABLE 4.2: Statistics of the Tweet Dataset.

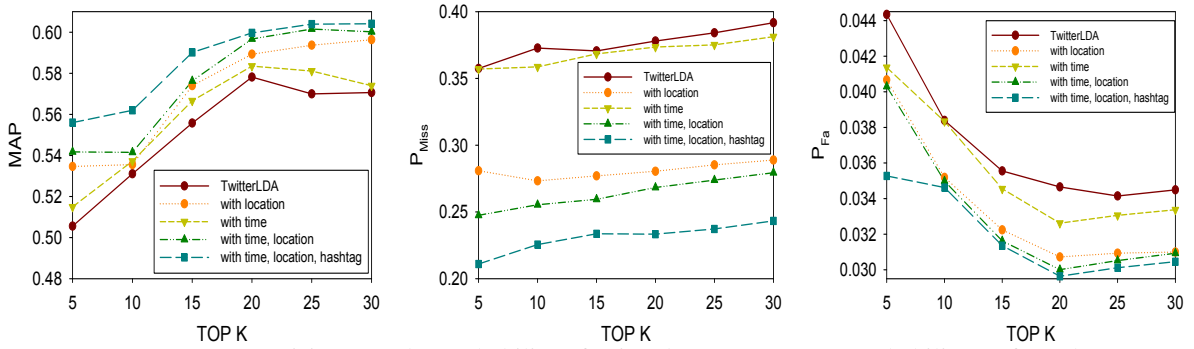
returns a small fraction of the total volume of tweets at any given moment, based on which events can rarely be detected. The search terms list we used is constructed by including a set of predefined event-relevant words (e.g., accident, earthquake, etc.), as well as the top 10 hottest trending words from the G20 countries provided by Twitter in real time. For each crawled tweet, five features (text, image, location, timestamp, hashtag) are recorded. The text of each tweet is preprocessed by removing the stop words, stemming and obtaining nouns and verbs only using the Stanford POS Tagger[2]. As for the images, convolutional neural networks (CNNs) have become more and more popular for extracting visual features from images recently. In this work, we use the widely used 'ImageNet network' which is trained on 1.3 million images [43]. Specifically, Caffe [39] is used to extract a 4096 dimensional visual feature from the layer (named "fc") before the classification layer. This visual feature is referred as "CNN" feature in this work. We define four categories including cartoon, landscape, diagram and the screenshot of text as the stop image categories and manually labelled 6600 images to train an SVM classifier for the stop image filtering. Specifically, 4600 images are used to train an SVM classifier with Gaussian kernel and the trained classifier is tested on the rest 2000 images showing a satisfying result (i.e., accuracy: 0.9017, precision: 0.8892 and recall: 0.9642). More details on the statistics of this dataset are summarized in Table 4.2.

4.4.2 Comparisons

Four competitor approaches are compared in our experiments to demonstrate the effectiveness of the event detection performance.

- **LDA**[12]:The original LDA is implemented as a baseline model. We adopt the same strategy

[2]http://nlp.stanford.edu/software/tagger.shtml



(a) Mean Average Precision (b) Probability of Missed Tweet Detections (c) Probability of False Tweet Alarms

FIGURE 4.3: Effect of Time, Location and Hashtag

as in LTT [99] to form events from the topics detected by LDA.

- **TwitterLDA**[98]: TwitterLDA is the first topic model specifically designed for tweets data. They assume that each short tweet is only assigned to one topic.

- **MMLDA**[10]: This is a topic model for Twitter subtopic discovery. Tweet images are first considered in this work, where each image is represented as a bag-of-visual-words derived from SIFT feature. We compare MMLDA to verify the superiority of the CNN feature in our proposed model over the traditional SIFT feature based methods.

- **LTT**[99]: This is the state-of-the-art work in topic modeling for Twitter event detection, which considers geographic information in addition to the text and time factor.

4.4.3 Performance on Event Detection

In this section, we first introduce the ground truth for event detection and the three performance indicators. After that, we study the impact of different Twitter features on event detection performance and evaluate the effectiveness of our event detection model in different aspects.

Ground Truth Generation

Due to the lack of ground truth for social event detection, we build up the ground truth (i.e., the top K hottest events on a particular day) based on the search terms used for collecting the tweets.

Intuitively, the more tweets crawled based on a search term, the hotter the event related to this term is. We calculate the sum of re-tweeted numbers for every search term in the daily collected tweets as the mentioned times. The K search terms with the top K largest mentioned times constitute the K ground truth events for that day.

Evaluation Metrics

Mean average precision (MAP) for a set of queries is the mean of the average precision for each query. In this work, we detect a list of ranked events daily. MAP is then calculated as the mean of the average event detection precision value of each day.

$$MAP = (\sum_{q=1}^{Q} \text{AveP}(q))/Q \tag{4.11}$$

where Q is the number of days we recommend events and the average precision (AveP) is defined as $AveP = \frac{\sum_{i=1}^{K}(\text{Precision}(i)\times\text{rel}(i))}{\text{number of relevant events}}$, where K is the number of detected events and Precision(i) is the precision at the top i events in the detected list. rel(i) equals to one if the the i-th event in the list is a relevant event, zero otherwise.

MAP evaluates the performance of the top K events recommendation, while the following two metrics verify the quality of one event. For a specific event, the tweets which should be assigned to this event are the *targets*, and all the other tweets are *nontargets*.

Probability of missed tweet detections (P_{Miss}) is the fraction of *missed tweet detections* over all targets:

$$P_{Miss} = \frac{\text{number of missed tweet detections}}{\text{number of targets}} \tag{4.12}$$

where a missed tweet detection is defined as a target tweet that is not assigned to the corresponding event.

Probability of false tweet alarm errors (P_{Fa}) is the percentage of *false tweet alarms* over all nontargets:

$$P_{Fa} = \frac{\text{number of false tweet alarms}}{\text{number of nontargets}} \tag{4.13}$$

where a nontarget tweet that is falsely detected in the corresponding event is called a false tweet alarm.

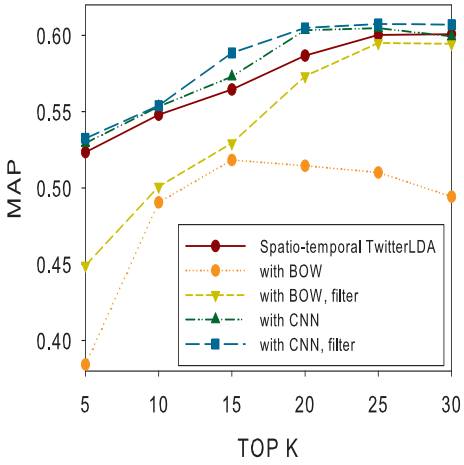


FIGURE 4.4: Effect of Visual Features

It is obvious that an effective event detection algorithm should have a large MAP but small P_{Miss} and P_{Fa}.

Effect of Time, Location and Hashtag

To evaluate the impact of different Twitter features in the proposed model, we provide five groups of results, including the original TwitterLDA as a baseline, TwitterLDA with time, TwitterLDA with location, Spatio-temporal TwitterLDA and Spatio-temporal TwitterLDA with hashtag. The effect of images will be verified separately in the next two subsections as it is a main contribution of this work. As shown in Figure 4.3, each Twitter feature improves the performance of the text-only based event detection (i.e., the original TwitterLDA) on all the three measurements. Among them, location information brings bigger improvements than the time factor, especially for P_{Miss} and P_{Fa} (i.e., relative 24% and 6% further declines respectively). With the consideration of the hashtag, MAP further increases around 3% when $K < 20$. Notably, the combination of all these features consistently generates the best results. This demonstrates the importance of each factor and the power of fusing them when detecting events from social networks.

Effect of Visual Feature

Image is one of the key factors in the proposed model and the choice of visual feature to represent images is thus of great importance. To test the effect of visual feature, we conduct experiments on all image tweets (i.e., tweets which contain images). Two different visual features are tested.

One is the bag-of-visual-words derived from the SIFT feature as generally used in existing studies and the other is the state-of-the-art CNN feature. Moreover, to demonstrate the advantage of the stop image filter component, the results for the whole image set and for the filtered image set are shown respectively. As illustrated in Figure 4.4, the bag-of-visual-words model on SIFT feature even deteriorates the performance. This shows that representing tweet images as visual words and straightforwardly processing them in the same way as textual words are inapplicable for social event detection. Moreover, for the bag-of-visual-words model, the lack of stop image filter even causes a fall in MAP when $K > 15$, which demonstrates that the noisy images affect the performance more seriously for larger K values. In contrast to the bag-of-visual-words model, CNN feature benefits the event detection by improving the MAP by 2% to 5%. In addition, using the stop image filter further increases the MAP by up to 3%.

Effect of Images on Different Categories

Images are supposed to have diverse impacts on the events in different categories. We manually divide all ground truth events in August 2013 into eleven major categories, including public holiday, ceremony, sport, celebrity, place of interest, product, weather, breaking news, trending topics, entertainment and politics. The distribution of the ground truth events in different categories is shown in Figure 4.5. To show the impact of images on different categories, the event detection performances for different categories with and without images are demonstrated in Figure 4.6. We can observe that images bring negative effect on four categories, including public holiday, sport, celebrity and trending topics. The reason is that, in these four categories, images are either visually similar to each other (for sport, celebrity) or containing too many noises (for public holiday, trending topics). Hence these images do not benefit the event detection process. Among all the other categories, ceremony and politics benefit most from tweet images (5.8% and 7.8% improvements in MAP respectively). This is because that the intra-event image similarity within an event and inter-event image diversity across events in these two categories are larger than those in other categories.

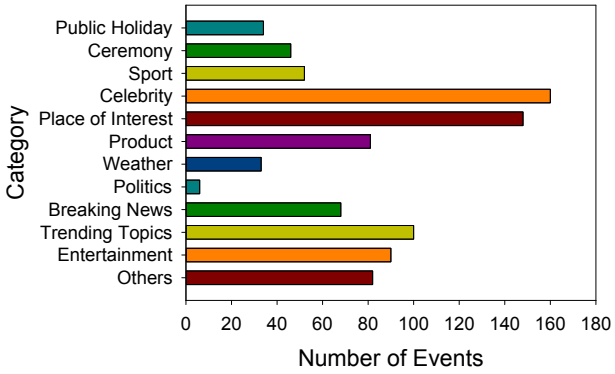


FIGURE 4.5: Category Distribution of the Ground Truth Events

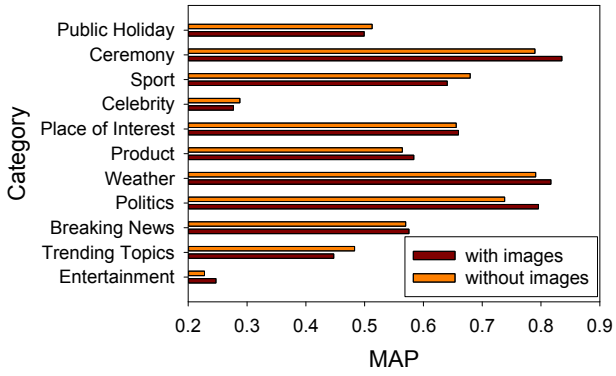
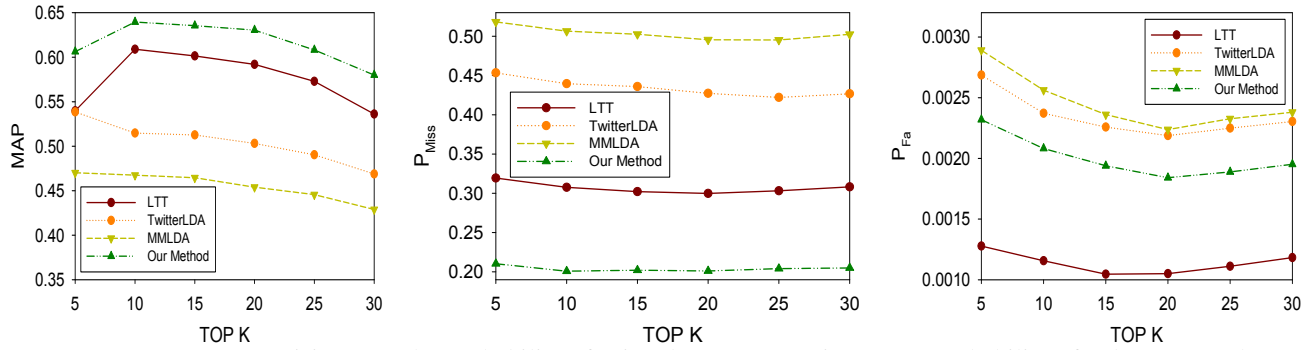


FIGURE 4.6: Effect of Images on Different Categories

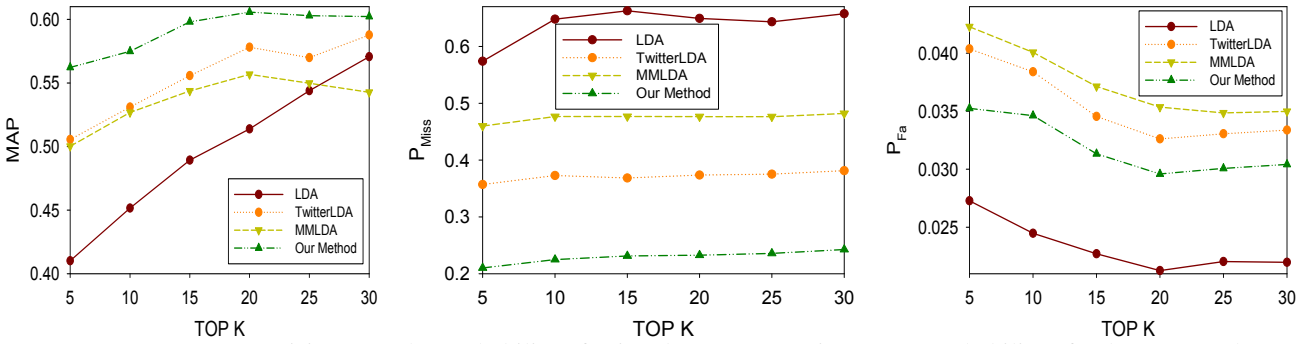
Effectiveness of Event Detection

In this part, we compared different event detection methods. Note that all of our collected tweets comprise country information, but only a small fraction of them (as shown in Table 4.2) contain latitude and longitude information. To test the algorithm LTT which requires the detailed geographical information, we construct a dataset named "Geo-labelled Data" consisting of a subset of our collected



(a) Mean Average Precision (b) Probability of Missed Tweet Detections (c) Probability of False Tweet Alarms

FIGURE 4.7: Event Detection Performance on Geo-labelled Data



(a) Mean Average Precision (b) Probability of Missed Tweet Detections (c) Probability of False Tweet Alarms

FIGURE 4.8: Event Detection Performance on All Data

tweets which contain latitude and longitude. We then conduct two series of comparisons on both the "Geo-labelled Data" (for LTT) and "All Data" (to show the consistent superiority on a large dataset). The results are shown in Fig 4.7 and Fig 4.8 respectively. We can see that our method performs the best in both MAP and P_{Miss} on both datasets. As in Fig 4.7(a), LTT is relatively 7% lower than our approach in MAP. Fig 4.7(b) shows that our method achieves remarkably smaller P_{Miss} compared to LTT (0.21 vs. 0.31 in average). The reason LTT gets a relatively lower P_{Fa} is that the events detected by LTT are much smaller than ours. This makes LTT less likely to generate false alarms since the number of tweets in each event is limited. It is the same for LDA (Figure 4.8(c)) due to the same event detection strategy as LTT. However, a big event may be separated into several small events in their methods. Moreover, comparing the performance of LTT and our method on P_{Miss}, we can observe the small events detected by LTT also causes many missed tweet detections. And as indicated by MAP, our method outperforms LTT in terms of the overall event detection performance. MMLDA is originally designed for subtopic discovery under the assumption that all the images are relevant to a certain topic. The noisy images in our collected tweet data result in the unsatisfactory event detection performance of MMLDA. LDA presents a marked increase in MAP with the increase of K. This demonstrates that when discussing the top few hottest events in Twitter, people are more focused and tend to concentrate on just one topic. However, for the less popular events, there might exist diverse discussions in one single tweet.

Date	Event Words
Aug 1, 2013	snowden 0.039, russia 0.038, **asylum** 0.016, **temporary** 0.014, moscow 0.012, **confirm** 0.010, airport 0.010, edward 0.009, paper 0.009, break 0.009
Aug 2, 2013	snowden 0.055, russia 0.032, **nsa** 0.021, asylum 0.017, edward 0.015, **read** 0.012, **leave** 0.011, **airport** 0.011, moscow 0.010, netflix 0.007
Aug 3, 2013	snowden 0.035, russia 0.033, **nsa** 0.012, moscow 0.010, edward 0.010, cbs 0.008, **russian** 0.008, **putin** 0.007, cable 0.007, **warner** 0.006

TABLE 4.3: Evolution of the Event "Snowden"

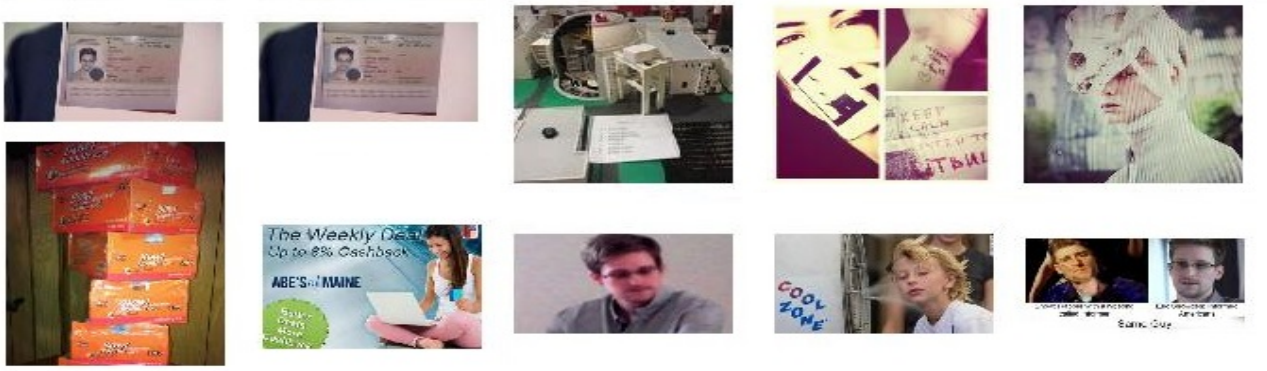
4.4.4 Event Evolution Tracking Study

We conduct a case study on the event "Snowden" to show the performance of event evolution tracking and event visualization. Table 4.3 illustrates the events related to "Snowden" detected by our framework, where the first column is the date when the event happened and the second column displays the top ten words for each event as well as their probabilities assigned to the event. This table shows the evolution of event "Snowden" in three consecutive days, from August 1, 2013 to August 3, 2013. On the first day, people were talking about Snowden's stay in the airport in Moscow and his temporary asylum issued by Russia. On the next day, the discussion shifted to that Snowden left the airport after his lawyer read the formal document. On the third day, the Russia president Putin and the warner Putin received from NSA became the foci of this event. This case study shows that the event detected from tweets can truly reflect what happens in real life. However, it is difficult to quantitatively evaluate such performance mainly due to the subjectiveness on event understanding.

4.4.5 Event Visualization Study

We compare our event visualization results for the example event "Snowden" with two baseline methods and a spectral clustering based method [10]. Observed from Fig 4.9, our method generates the most relevant and diverse representative images for the event "Snowden". As we mentioned in Section 2.2.2, there exist a certain number of images which are less relevant to the event, and existing cluster-and-rank methods (e.g., [10]) lack the strategy to deal with these images, as shown in Fig 4.9(a). Figure 4.9(b) and 4.9(c) show the results based on the criterion *visual relevance* and *visual coherence* respectively. Figure 4.9(d) demonstrates the importance of considering *distinctiveness* in addition to visual relevance and visual coherence.

(a) Spectral Clustering Based



(b) Visual Coherence Only

(c) Visual Relevance Only

(d) Our Method

FIGURE 4.9: Representative Image Selection for Event “Snowden”

4.5 Summary

In this chapter, we have presented a novel topic model named Spatio-Temporal Multimodal LDA for effective social event detection. Five different social data features are fully exploited for performance improvements. Social images are thoroughly studied and a two-filter strategy is specifically designed for eliminating the impact of noisy tweet images. Extensive experiments are conducted on a real-world tweet dataset to demonstrate the effectiveness of our method from different aspects.

Chapter 5

Community Profiling for Social Events Understanding

5.1 Introduction

In Chapter 3 and Chapter 4, we focus on detecting and monitoring the evolving events from the user-generated contents in social networks. However, we have only analysed social events from the content perspective so far. Investigating the underlying communities for each social event is another crucial perspective towards event understanding. The underlying community profiling makes the descriptions of the events more comprehensive by analysing these social events from the user perspective. Furthermore, community profiling provides the information diffusion strengths within one community and between communities. This means that we can not only have a knowledge of the events that are happening at the moment, but also further predict how will the information about the similar events diffuse in the social networks in the future. Motivated by the rich semantics about user behaviours hidden in these information, we extend the community definition as a group of users who are not only densely connected (as in the conventional community definition), but also having similar behaviours. With the richer semantics, such kind of communities can not only help to make sense of the social events from community perspective, but also benefit a lot of other applications that require community-level analysis, such as marketing research, public policy making and so on. In

this chapter, we study a new community profiling problem which detects and characterizes a community by both *content profile* (what a community is about) and *diffusion profile* (how it interacts with others). The majority of the community detection methods can be divided into three categorises: distance-based, graph-based and probabilistic model-based. The distance-based methods generally design some distance or similarity measurements between two users or two documents. Then the clustering methods are applied to group similar users into communities. The graph-based methods treat the network data as a graph where users correspond to nodes. Then the graph theories are applied which consider the graph characters such as density, in degree and out degree of nodes to group the dense and similar nodes into communities. The probabilistic model (topic model) naturally combines all the observed data (such as the contents posted by users, the links between users and links between documents) to discover the latent community assignments of each user. Compared to the distance-based methods, no similarity measurements need to be explicitly designed in topic models. Especially, when both user similarity and document similarity are considered simultaneously, how to combine the two similarity measurements will be problematic for distance-based methods. In terms of the graph-based methods, most of the existing studies only consider modularity as the criteria to group densely connected users into one community. The content similarities between the documents posted by the users are of great importance especially for social community detection. In consideration of the above limitations of both distance-based methods and graph-based methods, we design a novel topic model to solve the social community detection problem. Specifically, we propose a novel generative community profiling model (named CP-JHC) with the consideration of our three insights: joint profiling and detection, heterogeneous user interactions and comprehensive diffusion modeling. To the best of our knowledge, we are the first to holistically model user links, user generated contents and user content diffusion for community profiling. Extensive experiments have been conducted on two real-world datasets and the results demonstrate the superiority of our model over the state-of-the-art baselines by 22.65% on community detection task and 18.74% on diffusion prediction task. A case study has been conducted to further illustrate the effectiveness of our model on community profiling task for social event understanding.

The rest of this chapter is organized as follows. We first introduce the problem formulation and then present our generative model for community profiling in Section 5.2. Section 5.3 provides the

Notation	Description
$\|U\|$, $\|V\|$, $\|M\|$	The number of users, words and timestamps
$\|C\|$, $\|Z\|$	The number of communities and event topics
$\|G\|$, $\|E\|$	The number of user links and diffusion links
d_{ui}	The i-th document generated by user u
$\|D_u\|$	The number of documents generated by user u
$\|V_{ui}\|$	The number of words in document d_{ui}
t_{ui}, w_{uiv}	The timestamp and the v-th word in document d_{ui}
c_{ui}, z_{ui}	The community label and topic label assigned to d_{ui}
E^t_{ij}	A diffusion link from document i to document j at timestamp t
G_{uv}	A user link from user u to user v
π_u	Multinomial distribution over communities specific to user u
θ_c	Multinomial distribution over event topics specific to community c
ϕ_z	Multinomial distribution over words specific to topic z
$\eta_{cc'z}$	Community level information diffusion probability from community c to c' on topic z
α, β, ρ	Dirichlet priors

TABLE 5.1: Notations for Social Community Profiling

inference for our community profiling model. Experimental results are shown in Section 5.4, followed by the conclusion in Section 5.5.

5.2 Social Community Profiling

5.2.1 Problem Formulation

In this section, we first introduce some key concepts in this task and then formulate the community profiling problem we aim to address. Table 5.1 summarizes the notations used in this work.

Definition 5.1. Heterogeneous Network. A heterogeneous network is a graph $\mathcal{G} = (U, D, G, E)$, where $u \in U$ corresponds to a user, and $d \in D$ represents a user-generated document. There are two types of links in $\mathcal{G}$. $G_{uv} \in G$ stands for a *user link* from user u to v, while $E_{ij} \in E$ indicates a *diffusion link* from document i to document j. Both types of links are directed in this work.

Take the Twitter network as an example, D_u denotes the set of tweets posted by user u, $G_{u,v}$ represents that user u follows user v and $E_{i,j}$ denotes that tweet j is a retweet from tweet i.

In this work, a community is a group of users who are densely connected, share similar topics of interest and follow similar diffusion patterns. Specifically, given a heterogeneous network $\mathcal{G}$, each detected community c must meet three requirements: 1) high modularity in terms of user links G;

2) similar event topic distribution in terms of user generated contents D; 3) similar community-level diffusion behaviours in terms of diffusion links E. Next we introduce two important definitions in this work, i.e., content profile and diffusion profile for a community.

Definition 5.2. Content Profile. The content profile of a community c is a multinomial distribution θ_c over event topics, where each dimension θ_{cz} denotes the probability of c discussing event topic z. In particular, an event topic $z \in \{1, \ldots, |Z|\}$ is a $|V|$-dimensional multinomial distribution ϕ_z over words, where each dimension represents the probability that a word $w \in \{1, \ldots, |V|\}$ is related to z.

Definition 5.3. Diffusion Profile. For a community c, the diffusion profile η_c is a $|C| \times |Z|$ matrix, in which each elements $\eta_{cc'z}$ denotes the information diffusion probability from community c to c' on event topic z.

With the definition of content profile and diffusion profile, we now can introduce the community profiling problem.

The Problem. Given a heterogeneous network $\mathcal{G} = (U, D, G, E)$, the research problem we aim to address in this work is to propose a unified model to detect a set of communities $c \in \{1, \ldots, |C|\}$ and assign each user $u \in \{1, \ldots, |U|\}$ a $|C|$-dimensional community distribution π_u representing the probability that u belonging to every single community. Moreover, each community c is characterized by a content profile θ_c and a diffusion profile η_c. These two profiles describe the social events in terms of who are actively involved in the social events discussion and how does the event information diffuse among communities in social networks.

5.2.2 Community Profiling Model

In this work, we propose a novel generative model named CP-JHC to address the community profiling problem. Given the three challenges introduced in Section 1.2.3, CP-JHC is designed in a way to simultaneously detect and profile communities, with the consideration of both heterogeneous user interactions and comprehensive diffusion modeling.

Model Design

In this section, we will introduce how we realize the three insights to our model design, which are proposed to address the three challenges. The graphical model of CP-JHC is shown in Figure 5.1, in which the three components that correspond to three insights are highlighted with different colors.

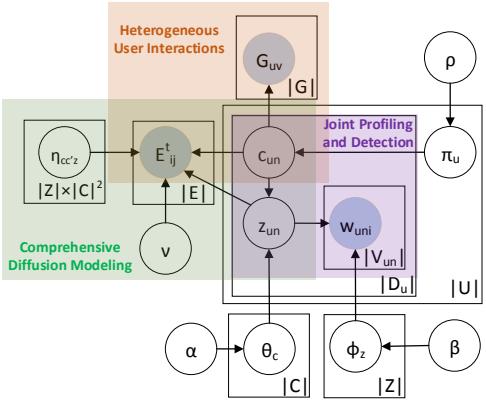


FIGURE 5.1: Graphical Model of CP-JHC

Joint profiling and detection. Jointly model community profiling and community detection can benefit the two tasks one another. In particular, the community detection mainly focuses on discovering a group of densely connected users, while the community profiling aims to provide the content profile and diffusion profile for each community. Given a heterogeneous network $\mathcal{G}$, our proposed CP-JHC jointly models the user links G, user published contents G and diffusion links E together to assign the community labels. Consequently, the community detection, content profiling and diffusion profiling tasks are completed simultaneously.

Heterogeneous user interactions. Given a heterogeneous network $\mathcal{G}$, we model user links G and diffusion links E heterogeneously in our proposed CP-JHC. These two types of links represent two user interactions. The user links stand for the interaction between two users, e.g., the following relationship in Twitter and the co-author relationship in DBLP. On the other hand, a diffusion link corresponds to the diffusion path of a document, such as the retweet in Twitter and citation in DBLP. In the conventional definition, a "good" community should have high modularity, which means the intra community links should be denser than the inter community links. In this work, user links are used to enforce the high modularity of the detected communities. The link probability between two users u and v is defined as a sigmoid function parametrized by the community distribution similarity

between u and v:

$$P_{uv} = \sigma(\hat{c}_u^{\mathrm{T}} \hat{c}_v) \tag{5.1}$$

where $\hat{c}_u$ is the community distribution of user u, which is computed by aggregating u's community assignments.

The more similar $\hat{c}_u$ and $\hat{c}_v$ are, the more likely G_{uv} exists. In other words, G_{uv} has higher probability to exist when u and v share more common community memberships. This inherently enforces that link connections within communities are denser than those across communities, leading to a high modularity score.

In contrast to user link, the intra community diffusion links are not necessarily to be denser than the inter community diffusion links. The reason is as follows. The inter community interactions are important when modeling diffusion links and they may not necessarily be "weak" [29]. Specifically, the community level diffusion probability varies across event topics. This topic-specific community level diffusion strength may break "the intra-community density" rule and should be carefully considered when we model the diffusion links. Next, we will introduce how we model the diffusion links comprehensively.

Comprehensive diffusion modeling. We characterize three important factors as the reasons for an information diffusion decision: 1) the community level diffusion preference, 2) event topic popularity, 3) individual diffusion preference. Take Twitter as an example, user u is likely to retweet v's tweet d_{vj} as her i-th tweet d_{ui} at timestamp t if: 1) the community level information diffusion strength between c_{ui} (the community u belongs to when she generates document d_{ui}) and c_{vj} on the event topic z_{vj} is strong; 2) the topic z_{vj} of d_{vj} is trending at timestamp t, hence d_{vj} is very likely to be retweeted; 3) u has individual preference on v, thus in general u tends to retweet v's tweets. Next we will explain how we incorporate the three factors in our diffusion modeling one by one.

The ***community level diffusion preference*** factor depends on two users' community memberships and the community-level diffusion strength on the specific event topic. In particular, given a user u, her underlying community topic joint probability is inferred as

$$P(c, z|u) = P(z|c)P(c|u) \propto \sum\nolimits_c \sum\nolimits_z \hat{c}_{uc} \hat{z}_{cz} \tag{5.2}$$

where $\hat{z}_c$ is the event topic distribution for community c and is computed by counting the topics assigned to the documents from c.

The community factor in determining whether v will diffuse u's document i is estimated by combining u and v's community memberships and the community-level diffusion strength on event topic z:

$$\sum_c \sum_{c'} \hat{c}_{uc} \hat{z}_{cz} \eta_{cc'z} \hat{c}_{vc'} \hat{z}_{c'z} \tag{5.3}$$

where $\eta_{cc'z}$ is the topic-specific diffusion strength from community c to c' on event topic z. For express simplicity, we set $\bar{c}_{ij} = vec([(\hat{c}_u \otimes \hat{c}_v) \circ (\hat{z}_z \otimes \hat{z}_z)])$ and $\bar{\eta} = vec(\eta)$, where $vec(A)$ is a vector concatenating the row vectors of A. Equation 5.3 is then converted to $\bar{c}_{ij} \cdot \bar{\eta}$.

The ***event topic popularity*** factor represents the popularity of an event topic at a specific timestamp. It is calculated as the count of topics z at timestamp t, denoted as $\hat{z}_z^t$.

The ***individual diffusion preference*** factor is computed as a linear function $\nu \cdot f$, where f is the vector which concatenates the individual feature vectors for user u and user v. We define the individual feature vector for user u as $< \frac{|Followers(u)|}{|Followees(u)|}, \frac{|Retweets(u)|}{|Tweets(u)|} >$ in this work, where the two terms represent the popularity and activeness of the user u. These two features are crucial when determining individual diffusion preference. Intuitively, u is more likely to diffuse information if she is very active. On the other hand, u's documents will tend to be diffused if she is popular.

In order to systematically combine the three diffusion factors, we propose to introduce a sigmoid function to define the diffusion probability from document i and j at timestamp t:

$$P_{ij}^t = \sigma(\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f) \tag{5.4}$$

where $\sigma(x) = 1/(1 + exp(-x))$. With the above sigmoid function, we can automatically learn the weight for each diffusion factor, so that the diffusion behaviours can be characterized more precisely.

Note that in this work, time is utilized to calculate the temporal popularity of a topic. It can be easily added as the observed data to model the latent community and topic labels as in [36]. We have experimentally verified the effectiveness of our model in terms of the above three insights and the results can be found in Section 5.4.2.

The Generative Process

In a heterogeneous network $\mathcal{G} = (U, D, G, E)$, given a user u, there are three possible behaviours: generating a document $d \in D$, forming a diffusion link $E_{ij} \in E$ and forming a user link $G_{uv} \in G$. Take Twitter as an example, the three behaviours for u are: posting a tweet, retweeting a tweet and following another user.

The generative process for the three behaviours is described as follows. When u posts a tweet d_{ui}, she first selects a community $c_{ui} = c$ based on her community distribution π_u. Then an event topic $z_{ui} = z$ is selected according to the community c's topic distribution θ_c. After that, every single words in d_{ui} is generated based on the word distribution ϕ_z of topic z. The link between user u and user v is formed by the similarity between the community distributions of user u and user v. The more similar communities user u and v share, the more likely u will follow v. In contrast to the user link, the generation of a diffusion link is more complicate, which depends on three factors as introduced in Section 5.2.2.

The generative process of a heterogeneous network $\mathcal{G}$ is summarized as follows.

1. For each event topic $z = 1, \ldots, |Z|$, draw $\phi_z|\beta \sim Dirichlet(\beta)$ indicating the word distribution for event topic z.

2. For each community $c = 1, \ldots, |C|$, draw $\theta_c|\alpha \sim Dirichlet(\alpha)$ denoting the topic distribution for community c.

3. For each user $u = 1, \ldots, |U|$

   (a) Draw $\pi_u|\rho \sim Dirichlet(\rho)$, denoting the community distribution for user u.

   (b) For the i-th document d_{ui} of user u

      i. Draw $c_{ui}|\pi \sim Multi(\pi_u)$, corresponding to the community assignment for d_{ui}

      ii. Draw $z_{ui}|c, \theta \sim Multi(\theta_{c_{ui}})$, denoting the topic assignment for d_{ui}

      iii. For the v-th word w_{uiv} in d_{ui}, $v = 1, \ldots, |V_{ui}|$

         A. Draw $w_{uiv}|z, \phi \sim Multi(\phi_{z_{ui}})$, a Multinomial distribution conditioned on topic z_{ui}.

(c) For each user link from user u to user v, draw $G_{u,v}|c \sim Bernolli(\sigma(\hat{c}_u^{\mathrm{T}}\hat{c}_v))$

(d) For every diffusion link $E_{i,j}^t$ user u generates from document i to document j at time t, draw $E_{i,j}^t|c, \eta, z, \nu, f \sim Bernolli(\sigma(\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f))$.

5.3 Model Inference

In our proposed CP-JHC model, there are two latent variables $\{z, c\}$ to be inferred from the observed data $\{W, G, E\}$. Collapsed Gibbs sampling algorithm is adopted to solve the probabilistic inference problem and obtain the two latent variables. The two sets of parameters, the variational parameters $\{\pi, \theta, \phi\}$ and the model parameters $\{\alpha, \beta, \rho, \nu, \eta\}$ are estimated in a variational EM procedure, in which the variation parameters are estimated in the E-step with the model parameters fixed and the model parameters are estimated in M step with others fixed. Next we will first introduce the collapsed Gibbs sampling algorithm in E-step and then illustrate the parameters estimation in the next Section.

5.3.1 The Collapsed Gibbs Sampling

To derive the formulas for Gibbs sampler, we start by computing the collapsed posterior distribution of our CP-JHC model is:

$$\begin{aligned} &p(W, G, E, C, Z, f, \nu, \eta|\rho, \alpha, \beta) \\ &= p(C|\rho)p(Z|C, \alpha)p(W|Z, \beta)p(G|c)p(E|c, \eta, z, \nu, f) \end{aligned} \tag{5.5}$$

The last two terms are the probability of the user links (G) and diffusion links (E), which are defined as $p(G) = \prod_u \prod_v P_{uv}$ and $p(E) = \prod_i \prod_j P_{ij}^t$. We follow the assumption in [14] that only the observed links will be modelled. Hence the sums in $p(G)$ and $p(E)$ range over all the observed links.

Note that $p(G)$ and $p(E)$ are two logistic likelihood functions making the Bayesian inference for our model a hard task since it is analytically inconvenient to form a Gibbs sampler for them. Recent studies [9, 17] propose a data-augmentation strategy which introduces a set of Pólya-Gamma random variables to derive an exact mixture representation of the logistic link likelihood.

In our model, we introduce two Pólya-Gamma variables λ and δ as the augmented variables for $p(G)$ and $p(E)$ respectively. These two logistic likelihood functions are now represented as:

$$p(G, \lambda) = \prod_u \prod_v e^{\frac{\hat{c}_u^{\mathrm{T}} \hat{c}_v - \lambda_{u,v} (\hat{c}_u^{\mathrm{T}} \hat{c}_v)^2}{2}} p(\lambda_{uv}|1, 0) \tag{5.6}$$

$$p(E, \delta) = \prod_i \prod_j e^{\frac{\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f - \delta_{i,j} (\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f)^2}{2}} p(\delta_{ij}|1, 0) \tag{5.7}$$

We will first introduce how to utilize the two Pólya-Gamma random variables to calculate the posterior probabilities for the Gibbs sampler of our model. A random variable X has a Pólya-Gamma distribution ($X \sim PG(a, b)$), if $X \overset{D}{=} \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{g_k}{(k-1/2)^2 + b^2/(4\pi^2)}$, where (a, b) are positive parameters and each $g_k \sim Gamma(a, 1)$ is an independent Gamma random variable.

To deal with the two logistic likelihood functions $p(G)$ and $p(E)$ in Equation 5.5, we introduce two Pólya-Gamma variables $\lambda \sim PG(1, 0)$ and $\delta \sim PG(1, 0)$ as the augmented variables. The two functions can then be represented as mixtures of Gaussians with respect to the corresponding Pólya-Gamma distribution as follows:

$$p(G_{uv}) = \frac{1}{2} e^{\frac{\hat{c}_u^{\mathrm{T}} \hat{c}_v}{2}} \int_0^{\infty} e^{-\frac{\lambda (\hat{c}_u^{\mathrm{T}} \hat{c}_v)^2}{2}} p(\lambda|1, 0) d\lambda$$
$$p(E_{ij}^t) = \frac{1}{2} e^{\frac{\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f}{2}} \int_0^{\infty} e^{-\frac{\delta (\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f)^2}{2}} p(\delta|1, 0) d\delta$$

Consequently, we derive $p(G, \lambda)$ and $p(E, \delta)$ in Equation 5.6 and 5.7.

We now have two latent variables and two augmented variables to be inferred from a given heterogeneous network, including the event topic z, community c and two augmented Pólya-Gamma variables λ and δ. At every iteration of the Gibbs sampling, we sample each of these variables one by one. In this section we will directly give the conditional distributions used in the collapsed Gibbs sampling for each variable.

In this section, we elaborate the detailed model inference formulas for Gibbs sampling. The collapsed posterior distribution of our model (Equation 5.5) augmented with two Pólya-Gamma variables λ and δ is represented as follows:

Augmented with two Pólya-Gamma variables λ and δ, collapsed posterior distribution of our

model (Equation 5.5) is represented as:

$$
\begin{aligned}
& p(W, G, E, C, Z, f, \nu, \eta, \lambda, \delta | \rho, \alpha, \beta) \\
&= p(C|\rho) p(Z|C, \alpha) p(W|Z, \beta) p(G, \lambda|c) p(E, \delta|c, \eta, z, \nu, f) \\
&= \int_\pi P(C|\pi) P(\pi|\rho) d\pi \int_\theta p(Z|C, \theta) P(\theta|\alpha) d\theta \int_\phi P(W|Z, \phi) P(\phi|\beta) d\phi \\
&\quad \times \prod_{uv} e^{\frac{\hat{c}_u^{\mathrm{T}} \hat{c}_v - \lambda_{u,v} (\hat{c}_u^{\mathrm{T}} \hat{c}_v)^2}{2}} p(\lambda_{uv}|1, 0) \prod_{ij} e^{\frac{\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f - \delta_{i,j} (\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f)^2}{2}} p(\delta_{ij}|1, 0) \\
&= \prod_{u=1}^{|U|} \frac{\Delta(n_u^{\mathbf{c}} + \rho)}{\Delta(\rho)} \prod_{c=1}^{|C|} \frac{\Delta(n_c^{\mathbf{z}} + \alpha)}{\Delta(\alpha)} \prod_{z=1}^{|Z|} \frac{\Delta(n_z^{\mathbf{w}} + \beta)}{\Delta(\beta)} p(G, \lambda) p(E, \delta)
\end{aligned}
\tag{5.8}
$$

Given Equation 5.8, the conditional distributions used in the collapsed Gibbs sampling for the event topic z and community c are derived as follows.

For Z: The probability of assigning topic z to z_{ui} is calculated as follows,

$$
\begin{aligned}
& P(z_{ui} = z | C, Z_{\neg\{ui\}}, W, G, E, f, \nu, \eta, \lambda, \delta) \\
&= P(z_{ui} = z | c_{ui} = c, C, Z_{\neg\{ui\}}, W_{\neg\{ui\}}, G, E, f, \nu, \eta, \lambda, \delta) \\
&\propto \frac{p(C, Z, W, G, E, f, \nu, \eta, \lambda, \delta | \rho, \alpha, \beta)}{p(C, Z_{\neg\{ui\}}, W_{\neg\{ui\}}, G, E, f, \nu, \eta, \lambda, \delta | \rho, \alpha, \beta)} \\
&= \prod_{c=1}^{|C|} \frac{\Delta(n_c^z + \alpha)}{\Delta(n_{c,\neg\{ui\}}^z + \alpha)} \prod_{z=1}^{|Z|} \frac{\Delta(n_z^w + \beta)}{\Delta(n_{z,\neg\{ui\}}^w + \beta)} p(G, \lambda) p(E, \delta) \\
&= \frac{n_{c,\neg\{ui\}}^z + \alpha}{n_{c,\neg\{ui\}}^{(\cdot)} + |Z| \times \alpha} \frac{\prod_{w=1}^{|W|} \prod_{i=1}^{n_{ui}^w} (n_{z,\neg\{ui\}}^w + \beta + i - 1)}{\prod_{j=1}^{n_{ui}^{(\cdot)}} (n_{z,\neg\{ui\}}^{(\cdot)} + |W| \times \beta + j - 1)} \prod_{v \in \Lambda_u} p^G(G_{uv}) \prod_{j \in \Lambda_i} p^E(E_{ij}^t)
\end{aligned}
\tag{5.9}
$$

For C: The probability of assigning community c to c_{ui} is derived as below,

$$
\begin{aligned}
& P(c_{ui} = c | C_{\neg\{ui\}}, Z, W, G, E, f, \nu, \eta, \lambda, \delta) \\
&= P(c_{ui} = c | z_{ui} = z, C_{\neg\{ui\}}, Z_{\neg\{ui\}}, W, G, E, f, \nu, \eta, \lambda, \delta) \\
&\propto \frac{p(C, Z, W, G, E, f, \nu, \eta, \lambda, \delta | \rho, \alpha, \beta)}{p(C_{\neg\{ui\}}, Z_{\neg\{ui\}}, W, G, E, f, \nu, \eta, \lambda, \delta | \rho, \alpha, \beta)} \\
&= \frac{\Delta(n_u^c + \rho)}{\Delta(n_{u,\neg\{ui\}}^c + \rho)} \frac{\Delta(n_c^k + \alpha)}{\Delta(n_{c,\neg\{ui\}}^k + \alpha)} p(G, \lambda) p(E, \delta) \\
&= \frac{n_{u,\neg\{ui\}}^c + \rho}{n_{u,\neg\{ui\}}^{(\cdot)} + |C| \times \rho} \frac{n_{c,\neg\{ui\}}^z + \alpha}{n_{c,\neg\{ui\}}^{(\cdot)} + |Z| \times \alpha} \prod_{v \in \Lambda_u} p^G(G_{uv}) \prod_{j \in \Lambda_i} p^E(E_{ij}^t)
\end{aligned}
\tag{5.10}
$$

Let $p^E(E_{ij}^t) = e^{\frac{\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f - \delta_{ij} (\bar{c}_{ij} \cdot \bar{\eta} + \hat{z}_z^t + \nu f)^2}{2}}$ and $p^G(G_{uv}) = e^{\frac{\hat{c}_u^{\mathrm{T}} \hat{c}_v - \lambda_{uv} (\hat{c}_u^{\mathrm{T}} \hat{c}_v)^2}{2}}$.

For Z: The probability of assigning event topic z to z_{ui},

$$\begin{aligned} &P(z_{ui} = z|C, Z_{\neg\{ui\}}, W, G, E, f, \nu, \eta, \lambda, \delta) \\ &= \frac{n^{z}_{c,\neg\{ui\}}+\alpha}{n^{(\cdot)}_{c,\neg\{ui\}}+|Z|\times\alpha} \frac{\prod_{w=1}^{|W|}\prod_{i=1}^{n^{w}_{ui}}(n^{w}_{z,\neg\{ui\}}+\beta+i-1)}{\prod_{j=1}^{n^{(\cdot)}_{ui}}(n^{(\cdot)}_{z,\neg\{ui\}}+|W|\times\beta+j-1)} \prod_{v\in\Lambda_u} p^{G}(G_{uv}) \prod_{j\in\Lambda_i} p^{E}(E^{t}_{ij}) \end{aligned} \tag{5.11}$$

where $\Lambda_u = \{v : (u, v) \in G||(v, u) \in G\}$ denotes the set of neighbours of user u in G. $\Lambda_i = \{j : (i, j) \in E||(j, i) \in E\}$ denotes the set of neighbours of document i in E. $n^{z}_{c,\neg\{ui\}}$ and $n^{(\cdot)}_{c,\neg\{ui\}}$ denote the number of times topic z assigned to community c and the total number of topics assigned to community c, both calculated with the current document d_{ui} excluded. Similarly, $n^{w}_{z,\neg\{ui\}}$ and $n^{(\cdot)}_{z,\neg\{ui\}}$ are the number of times word w assigned as topic z and the total number of words assigned as topic z, with d_{ui} excluded; n^{w}_{ui} and $n^{(\cdot)}_{ui}$ are the number of times word w occurs in the document d_{ui} and the total number of words in d_{ui}.

For C: The probability of assigning community c to c_{ui} is,

$$\begin{aligned} &P(c_{ui} = c|C_{\neg\{ui\}}, Z, W, G, E, f, \nu, \eta, \lambda, \delta) \\ &= \frac{n^{c}_{u,\neg\{ui\}}+\rho}{n^{(\cdot)}_{u,\neg\{ui\}}+|C|\times\rho} \frac{n^{z}_{c,\neg\{ui\}}+\alpha}{n^{(\cdot)}_{c,\neg\{ui\}}+|Z|\times\alpha} \prod_{v\in\Lambda_u} p^{G}(G_{uv}) \prod_{j\in\Lambda_i} p^{E}(E^{t}_{ij}) \end{aligned} \tag{5.12}$$

where $n^{c}_{u,\neg\{ui\}}$ and $n^{(\cdot)}_{u,\neg\{ui\}}$ are the number of documents from user u that are assigned to community c and the total number of documents from user u without counting the document d_{ui}.

For λ: The conditional distribution of the augmented variable λ is a Pólya-Gamma distribution:

$$p(\lambda_{uv}|W, G, E, C, Z, f, \nu, \eta, \delta) \propto e^{\frac{-\lambda_{uv}(\hat{c}_u^{\mathrm{T}}\hat{c}_v)^2}{2}} p(\lambda_{uv}|1, 0) = PG(1, \hat{c}_u^{\mathrm{T}}\hat{c}_v) \tag{5.13}$$

We adopt the efficient approach proposed in [72] to draw samples from the Pólya-Gamma distribution. The samples of λ are drawn from the closely related exponentially tilted Jacobi distribution.

For δ: Similarly, the posterior distribution of the other augmented variable δ is also a Pólya-Gamma distribution:

$$p(\delta_{ij}|W, G, E, C, Z, f, \nu, \eta, \lambda) \propto e^{\frac{-\delta_{ij}(\bar{c}_{ij}\cdot\bar{\eta}+\nu f+\hat{z}^{t}_{z})^2}{2}} p(\delta_{ij}|1, 0) = PG(1, \bar{c}_{ij} \cdot \bar{\eta} + \nu f + \hat{z}^{t}_{z}) \tag{5.14}$$

5.3.2 Parameters Estimation

We use the following variational EM procedure to estimate the parameters of our CP-JHC model.

1. (E-step) For each document and two types of links, use Gibbs sampling as described before to find the optimizing values of the parameters π, θ and ϕ.

2. (M-step) Maximize the joint likelihood of the model with respect to the model parameters α, β, ρ, ν and η given the parameters π, θ and ϕ fixed as computed in E-step.

In the E-step, the Gibbs sampler iteratively draws samples of Z, C, λ and δ from the poster distribution based on Equation(5.11), Equation(5.12), Equation(5.13) and Equation(5.14), respectively. The three parameters updated after each Gibbs sampling iteration are estimated as: $\pi_{uc} = \frac{n_u^c+\rho}{n_u^{(\cdot)}+|C|\times\rho}$, $\theta_{cz} = \frac{n_c^z+\alpha}{n_c^{(\cdot)}+|Z|\times\alpha}$ and $\phi_{zw} = \frac{n_z^w+\beta}{n_z^{(\cdot)}+|W|\times\beta}$.

Model parameters are estimated in the M-step. Notably, the parameters of some well-studied Dirichlet distribution are set fixed as $\alpha = 50/T$, $\beta = 0.1$, $\rho = 50/C$ for simplicity as in existing work. The remaining two parameters η and ν are estimated as below.

The community-level information diffusion strength $\eta_{cc'z}$ is calculated by aggregating the community and topic assignments for each single retweet at the last iteration of Gibbs sampling. ν is solved in the M-step by optimizing the logistic regression function with all other variable fixed as described before. To solve the logistic regression function, we randomly sample the same amount of negative diffusion links from all non-observed diffusion links as the true negative instances.

5.4 Experiments

In this section, we evaluate the performance of our proposed community profiling model on two large real-world datasets. We design the experiments to: 1) validate our three insights as just mentioned in Section 5.2.2; 2) compare our model with the state-of-the-art baselines.

Dataset	Twitter	DBLP
Number of users	137,325	916,907
Number of user links	3,589,811	3,063,186
Number of diffusion links	992,522	10,210,652
Number of words	2,316,020	330,334
Number of documents	39,952,379	4,121,213

TABLE 5.2: Dataset Statistics

5.4.1 Set Up

Datasets

We conduct the experiments on the following two publicly available benchmark datasets: Twitter [50] and DBLP [85].

Twitter: The Twitter dataset was collected in May 2011 by crawling Twitter with Twitter API. 100,000 random users were first selected as seeds to crawl users' following relationships. From the retrieved 20 million users in the followship network, 3 million users who have at least 10 relationships were selected to further crawl their user profiles. After that, 150 thousand users who have locations in their profiles were selected, and at most 500 public tweets were crawled for each of them.

DBLP: The DBLP dataset is collected from ArnetMiner [85] (from 1936 to 2010). There are 1,572,277 papers and 2,084,019 citation relationships in total. Each paper is associated with abstract, author, year, venue and title.

The text of each tweet and paper title are pre-processed by removing the stop words, stemming and obtaining nouns and verbs only using the Stanford POS Tagger [1]. We then remove the documents containing less than 2 words and accordingly remove users having no documents. Table 5.2 lists the data statistics of the two datasets after pre-processing. All the experiments are implemented on a server with Microsoft Windows Server 2012 R2 Datacenter, Intel(R) Xeon(R) E5-2690 @3.00GHz CPU, and 256GB of RAM.

We make our code public online[2].

[1] http://nlp.stanford.edu/software/tagger.shtml
[2] https://bitbucket.org/vincentzheng/communityprofiling/

Evaluation Metrics

As there is no ground truth, it is hard to directly evaluate the quality of our community profiling outputs. Therefore, we design different tasks, which rely on our community profiling output for prediction, to evaluate the quality of our community profiling. Intuitively, a good community profiling model for social event analysis should be capable of detecting communities and predicting information diffusion accurately so as to describe the social events in terms of both user and information diffusion perspectives. A case study is also conducted to verify our community profiling performance, in which both the *content profile* and the *diffusion profile* are illustrated by the outputs of our CP-JHC model.

As introduced in Section 5.2.1, given a heterogeneous network $\mathcal{G} = (U, D, G, E)$, a good community profiling model should be able to detect dense communities from $\mathcal{G}$, and accurately predict the existence probability of the two types of links: user links G and diffusion links E. In this work, we choose modularity for community detection as it is widely adopted to evaluate the density of a community. Meanwhile, AUC is adopted to evaluate both user link prediction and diffusion link prediction. The benefit is that we do not need to define the threshold for link existence since AUC is the evaluation measurement for ranking performance. The detailed introduction of the two measurements can be found below.

Modularity: Modularity is the fraction of the edges that fall within the given groups minus the expected number in an equivalent network with edges placed at random [67]. Since each user belongs to multiple communities in our work, we adopt the modularity measure of networks with overlapping communities [44]. Specifically, the modularity score of a network with $|C|$ overlapping communities is calculated as follows:

$$Modularity = \frac{1}{|C|} \sum_{c \in C} \left[\frac{\sum_{u \in c} \frac{\sum_{v \in c, u \neq v} a_{uv} - \sum_{v \notin c} a_{uv}}{d_u \cdot s_u}}{n_c} \cdot \frac{n_c^e}{\binom{n_c}{2}} \right]$$

where n_c and n_c^e is the number of users and the number of links community c contains. $a_{uv} = 1$ if user

u connected to user v, otherwise $a_{uv} = 0$. d_u is the degree of user u in the network and s_u denotes the number of communities user u belongs to.

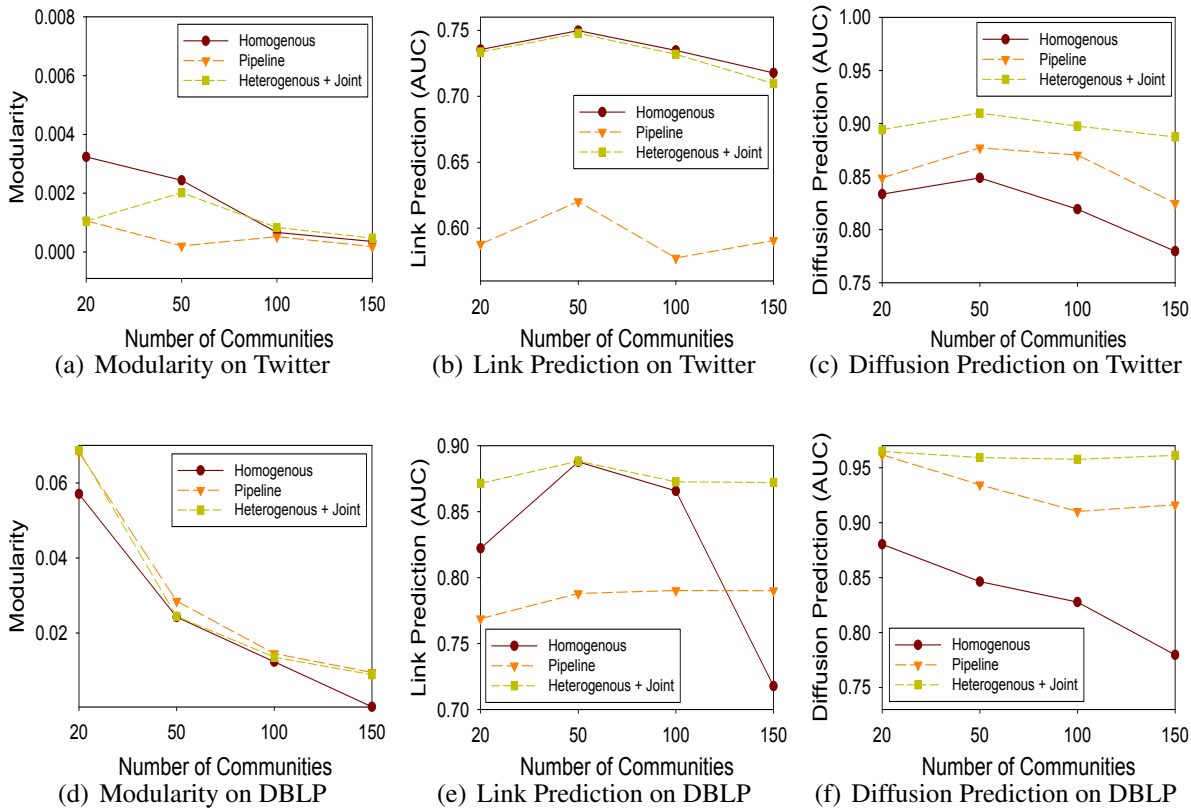
AUC: The *area under the receiver operating characteristic curve* (AUC) [3] value represents the probability that a randomly chosen true positive link is ranked above a randomly picked true negative link given a list of ranked non-observed links [36]. Specifically, the probability of a user link G_{uv} and a diffusion link E^t_{ij} is defined as Equation 5.1 and Equation 5.4.

To evaluate the community detection and diffusion prediction performance, we follow the settings in [36]. 20% user links and diffusion links are randomly selected from G and E respectively. After that, the same amount of random negative links are selected to evaluate AUC for both user links and diffusion links. More specifically, take user link prediction as an example, given a list of true neighbours who follow users u (denoted as $Nr(u)$), we randomly select the same amount of non-neighbours (denoted as $\neg Nr(u)$) who do not follow u. P_{uv} is then calculated for each user $v \in \{Nr(u) \cup \neg Nr(u)\}$. AUC is derived based on the user links ordered by the predicted link probability P_{uv}. The evaluation for diffusion links is similar, in which the true negative links are selected from the users who have not retweeted a specific tweet. Our model is trained on the remaining 80% links and all documents.

Baselines

We compare our model with four state-of-the-art baselines to evaluate the performance of our model. We summarize our differences with the baselines in Table 5.3. Both PMTLM and COLD omit user links in their model and only consider the community factor when modeling diffusion links. Their difference is that PMTLM bounds one topic to one community while COLD presents a community as a distribution of topics. We compare with each of them on both community detection and diffusion prediction tasks to verify the importance of user links and comprehensive diffusion modeling for community profiling. Similar to us, CRM utilizes user links and diffusion links when detecting communities and considers both community and individual factors for diffusion modeling. It is included to compare different link modeling approaches for community model (Multinomial link modeling vs. Bernolli Sigmoid link modeling). At last, we use WTM to compare the diffusion prediction

[3] http://en.wikipedia.org/wiki/Roc_curve



(a) Modularity on Twitter (b) Link Prediction on Twitter (c) Diffusion Prediction on Twitter

(d) Modularity on DBLP (e) Link Prediction on DBLP (f) Diffusion Prediction on DBLP

FIGURE 5.2: Validation of Joint Profiling and Detection & Heterogeneous User Interactions

performance of direct individual diffusion modeling against the comprehensive diffusion modeling.

- **Poisson Mixed-Topic Link Model (PMTLM)** [107]. PMTLM designs a generative model for both text and user links. Both texts and links are generated by the same latent variable; hence one community only discusses one topic.

- **Whom to Mention (WTM)** [89]. WTM extracts different features including user interest match, content-dependent user relationship and user influence for each user. A SVR method is adopted to rank the users based on their diffusion probabilities. Only individual factors are considered when calculating the retweet probability between two users.

- **Community Role Model (CRM)** [34]. CRM designs a generative framework to model a social network, which incorporates all the information such as links, user attributes and behaviours in a unified manner. CRM defines a community as a multinomial distribution over all links and models diffusion actions using both individual and community factors.

TABLE 5.3: Baselines

	features			diffusion factors			tasks			
methods	text	user links	diff links	indi	comm	topic	topic extr	comm detec	diff pred	comm prof
PMTLM	•		•		•		•	•	•	
WTM	•	•	•	•					•	
CRM		•	•	•	•			•	•	
COLD	•		•		•		•	•	•	
Ours	•	•	•	•	•	•	•	•	•	•

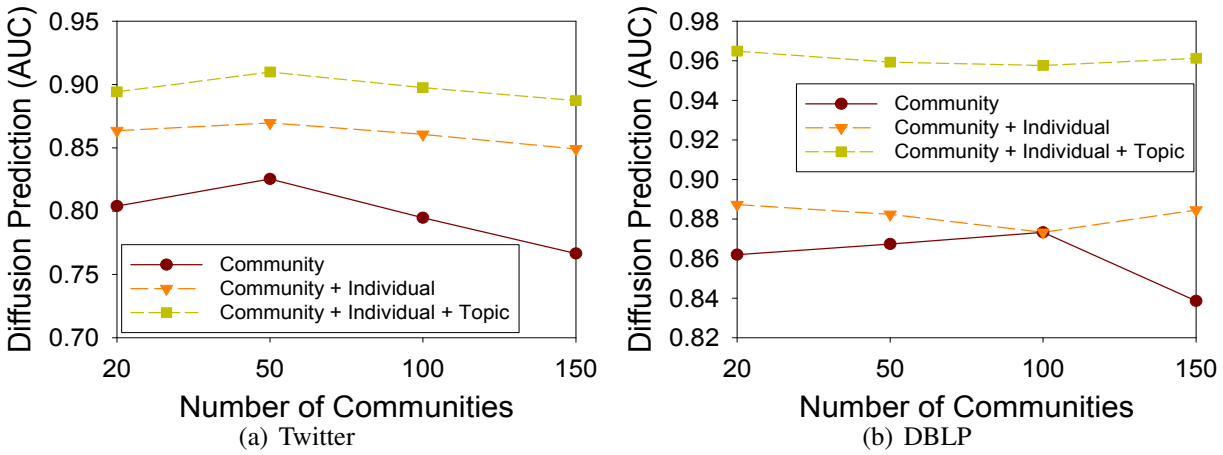
- **COmmunity Level Diffusion (COLD)** [36]. COLD proposes a generative latent topic model which simultaneously discovers the hidden topics, communities and inter-community influence by jointly modeling network, text and time. In contrast to WTM, only community level factors are utilized to model the retweet behaviour between two users.

5.4.2 Validation of Our Insights

To verify our three insights introduced in Section 5.2.2, we conduct three sets of experiments on both datasets as follows.

Joint profiling and detection. To show the advantage of joint detection and profiling, we compare with a pipeline baseline. Specifically, we first detect communities from the network based on user links only and then extract event topics and diffusion strength for each detected community. Figure 5.2 shows results for the experimental validation of joint profiling and detection and heterogeneous user interactions. The pipeline community profiling and detection leads to a significant decline in terms of Link Prediction AUC on both datasets (relatively 18.7% on Twitter and 10.5% on DBLP). This verifies the benefit of joint profiling and detection.

Heterogeneous user interactions. The importance of heterogeneous user interactions is verified by evaluating the performance of the homogeneous user interactions model which utilizes the same formula (i.e., Equation 5.1) to model both user links and diffusion links. As shown in Figure 5.2, modeling user links and diffusion links homogeneously decreases the diffusion prediction performance by around 8.6% on Twitter and 13.2% on DBLP respectively. This verifies that diffusion links are not the same as user links in modeling communities. Interestingly, the homogeneous link modeling brings slight improvements in terms of Link Prediction AUC on Twitter but the opposite effects on DBLP.



(a) Twitter (b) DBLP

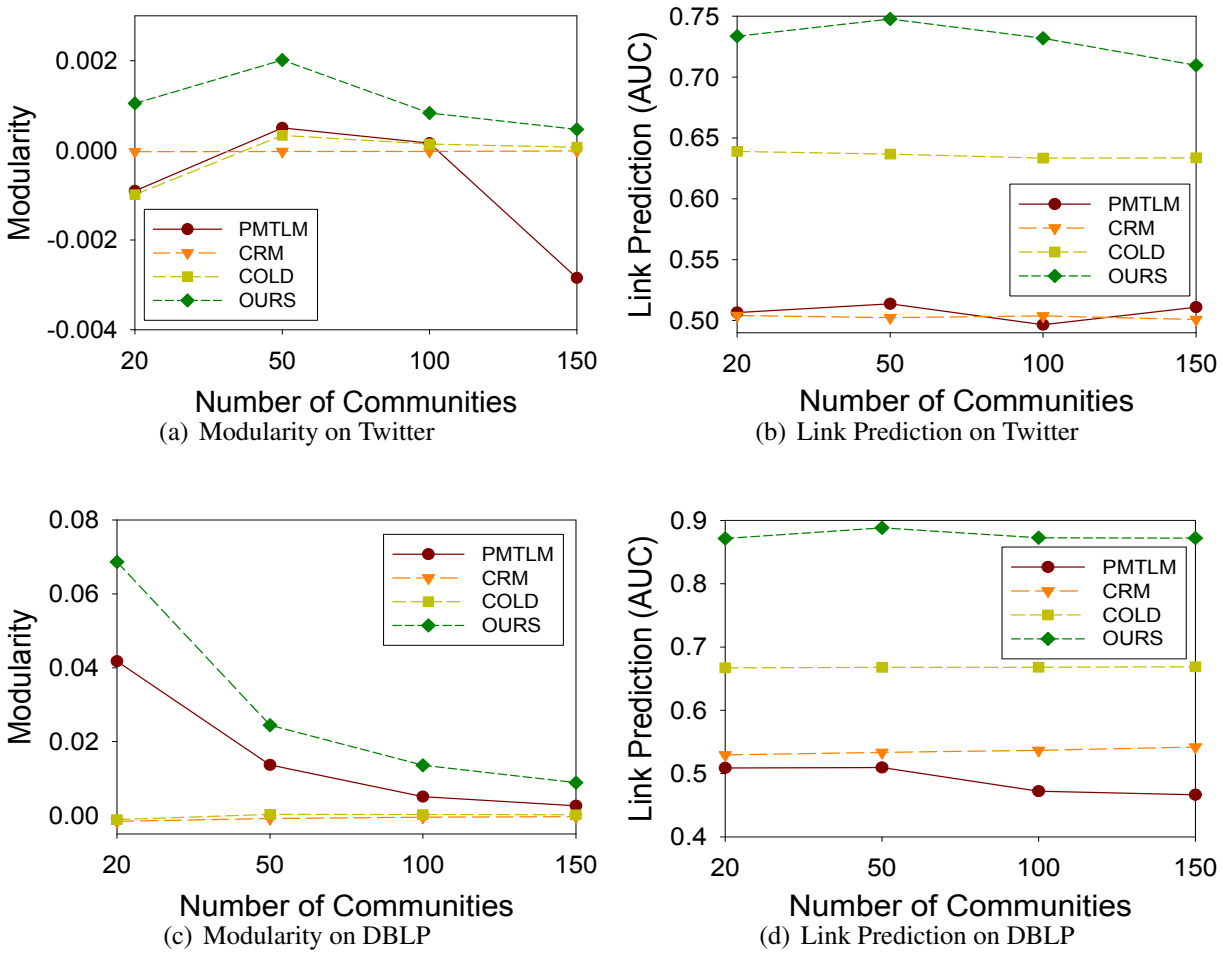
FIGURE 5.3: Validation of Comprehensive Diffusion Modeling

This is because the diffusion links only happen between users who follow each other in Twitter. Hence modeling diffusion links the same way as user links strengthens a subset of the user links generation, leading to the minor improvements when predicting the user links. However, in DBLP, the diffusion links (citation) are not limited to the users who are connected to each other (coauthor). Consequently, the user link prediction performance is affected by the diffusion links.

Comprehensive diffusion modeling. To demonstrate the benefit of comprehensive diffusion modeling, we test the effect of the three diffusion factors on the diffusion prediction performance. As in Figure 5.3, the individual preference factor brings around 8% and 2.5% improvements on the performance of the community level preference factor based diffusion prediction for Twitter and DBLP datasets respectively. With the consideration of topic popularity factor, AUC further increases around 4.2% and 9% on both datasets. Note that the combination of all the three factors consistently generates the best results, demonstrating the importance of fusing different factors comprehensively when modeling the diffusion links.

5.4.3 Results on Community Profiling

In this section, we compare our model with several state-of-the-art baselines in terms of the performance on two tasks: community detection and diffusion prediction.



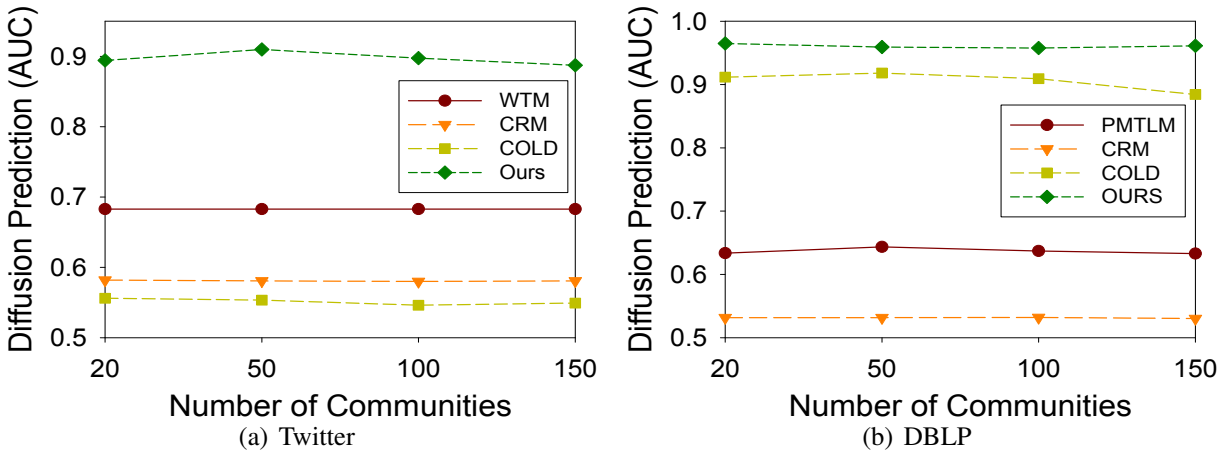
(a) Modularity on Twitter (b) Link Prediction on Twitter

(c) Modularity on DBLP (d) Link Prediction on DBLP

FIGURE 5.4: Community Detection Performance

Effectiveness of Community Detection

We first demonstrate the superiority of our model over existing approaches in terms of community detection. The results on community detection are shown in Figure 5.4. We can see that our model outperforms others in both Modularity and Link Prediction AUC. Specifically, our model achieves the best Modularity due to our user links modeling (Equation 5.1). Both CRM and COLD do not consider Modularity in their model design. Although CRM also utilizes user links, they define the community as a multinomial distribution over all user links, which cannot guarantee the modularity of the detected communities (as shown in Figure 5.4(a) and 5.4(c)). PMTLM gets relatively higher Modularity than the other two algorithms because the link generation in their methods depends on the the community distribution similarity between the two documents. However, each community only bounded with one topic in their model, which makes PMTLM hard to fit the real-world data. This is



(a) Twitter (b) DBLP

FIGURE 5.5: Diffusion Prediction Performance

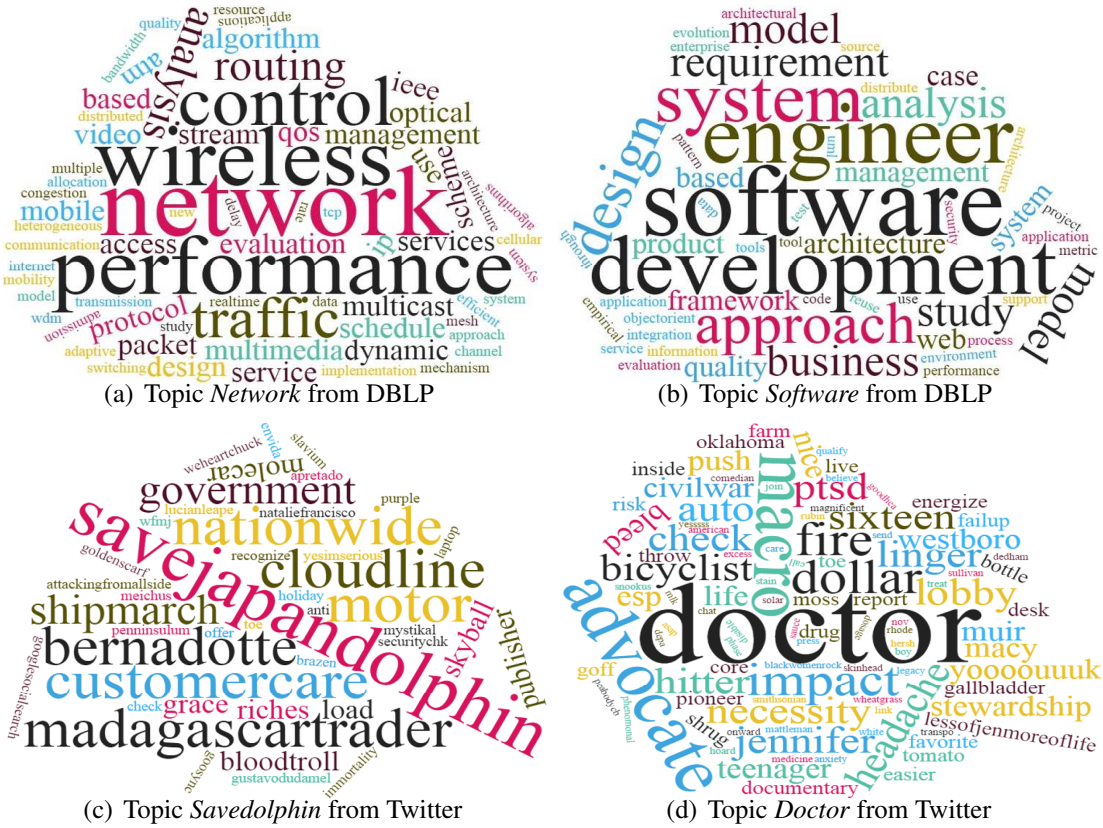
verified by their low Link Prediction AUC as shown in Figure 5.4. Our model achieves remarkably larger Link Prediction AUC compared to the state-of-the-art method COLD (0.73 vs. 0.64 in Twitter and 0.88 vs. 0.67 in DBLP, respectively). The performance of CRM shows that modeling community as multinomial distribution over links are not effective for network data.

Effectiveness of Diffusion Prediction

In this part, we compared different diffusion prediction methods on both datasets. As shown in Figure 5.5, our model achieves the highest AUC in terms of diffusion prediction on both dataset. WTM specifically designs a set of individual features for Twitter users and beats the other two algorithm with around 17.6% higher AUC in Twitter. With the consideration of community and topic factor, our model further improves the diffusion prediction performance by 31.4% in Twitter. Notably, the diffusion prediction performance of COLD is closer to our model in DBLP than in Twitter. This demonstrates that community factor contributes more to the information diffusion in DBLP than in Twitter.

5.4.4 Community Profiling Case Study

We conduct a case study to show the performance of community profiling. Specifically, we illustrate the content profiles and diffusion profiles for the communities detected by our model from DBLP dataset with setting $|C| = |Z| = 50$.



(a) Topic *Network* from DBLP (b) Topic *Software* from DBLP

(c) Topic *Savedolphin* from Twitter (d) Topic *Doctor* from Twitter

FIGURE 5.6: Word Clouds of Extracted Topics

Content Profile The content profile is an event topic distribution published by a community. We first present the word clouds of four extracted topics to better understand the topics extracted by our model. Figure 5.6 illustrates the word clouds of four event topics: *Network* and *Software* from DBLP, and *Savejapandolphin* and *doctor* from Twitter. Table 5.4 lists the content profile for five communities detected by CP-JHC model, where the first column indicates the community label and the second column displays the top five topics published by each community along with their probabilities being assigned with the corresponding community. As shown in Table 5.4, community 6, 16, 44, 45, 47 mainly publish papers in topics "neural network", "simulation analysis", "problem optimization", "motion tracking" and "knowledge analysis" respectively. Next we will show the diffusion profile of community 6, which mainly cites papers from the five communities listed in Table 5.4.

Diffusion Profile Table 5.5 lists the summarized diffusion profile for community 6. The first column presents the source community where the information is from (i.e., the community from which the papers in community 6 cite) and the second column lists the diffusion strengths from the

Community	Top 5 Topics
6	**neural network** 0.870, data classification 0.034, algorithm optimization 0.0285 knowledge analysis 0.012, image video retrieval:0.008
16	**simulation analysis** 0.837, quantum analysis 0.045, molecular dynamics study 0.028 neural network 0.010, solve equation problem 0.007
44	**solve equation problem** 0.514, simulation analysis 0.426, probebased technology 0.014 data modeling 0.011, molecular dynamics study 0.005
45	**motion tracking** 0.218, face recognition 0.086, image segmentation 0.079 imagevideo retrieval 0.071, data classification 0.063
47	**knowledge analysis** 0.981, computer science 0.010, virtual system design 0.002 web information system 0.001, face recognition 0.001

TABLE 5.4: Community *Content Profile* Examples

Source Community	Diffusion Strengths Across Topics
6	**neural network** 0.005684, synchronize technique 0.001085 simulation analysis 0.001616, web image video retrieval 3.9E-5
16	**simulation analysis** 5.8E-5, architecture design parallel 5.6E-5 neural network 2.1E-5, quantum analysis 4E-6
44	neural network 1.79E-4, **solve equation problem** 3E-5 simulation analysis 1E-5, molecular dynamics study 6E-6
45	**motion tracking** 6.25E-4, neural network 3.6E-5 graph complexity 2.7E-5, web system 1.4E-5
47	**knowledge analysis** 9.85E-4, neural network 8.95E-4 robot control 4.48E-6, synchronize technique 1.21E-6

TABLE 5.5: *Diffusion Profile* of Community 6 (neural network)

source community to community 6 on different topics. From Table 5.4 and 5.5 we can observe that, community 6 mainly cites paper on Topic "neural network" which is its major topic. But when it publishes paper in other topics, it tends to cite papers from the "expert" communities, e.g., the Topic "simulation analysis" from community 16. This proves that inter community diffusion strengths are not necessarily to be "weak" as mentioned in **Heterogeneous user interactions**.

5.4.5 Efficiency

Inspired by [66], we further implement a parallelized CP-JHC model for efficiency. Specifically, the dataset is divided into several parts based on the topics each user discusses which is derived by applying original LDA on all texts. We then assign the community and topic label to the documents from different data subsets in parallel in each Gibbs sampling iteration. The global counters such as user-community counter and community-topic counter are synchronized at the end of every Gibbs

sampling iteration. With such multi-threading implementation, we divide the Twitter and DBLP datasets into 24 and 12 subsets respectively and the training time of our model is improved by up to 25 times faster in Twitter and 6 times faster in DBLP. The links in DBLP is much denser than in Twitter, hence the improvements in the former is lower than in the latter.

5.5 Summary

In this chapter, we have presented a novel community profiling model named CP-JHC for social event analysis, which considers joint profiling and detection, heterogeneous user interactions and comprehensive diffusion modeling. Extensive experiments are conducted on two large real-world datasets to demonstrate the effectiveness of our community profiling methods from different aspects. The efficiency is guaranteed by a multi-threading implementation of CP-JHC.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we study the problem of multi-perspective analysis of social events. We propose several novel models and efficient algorithms to detect, monitor and visualize the social events from content aspect and community aspect. In the first step, we detect and monitor the evolving events from the highly dynamic social text streams with the aid of a novel event index structure as introduced in Chapter 3. In the second step, to further understand the social events, we design a novel generative model which explores various social data features to facilitate the event tracking and visualization process in Chapter 4. In the last step, we propose a novel community profiling model to analyse the underlying communities involved in the social events, which comprehensively analyses user content, user links and diffusion links in Chapter 5.

In Chapter 1, we provide the general introduction of this thesis. We first introduce the research background of the multi-perspective social event analysis problem. Afterwards, we present the three research problems included in the thesis and the corresponding challenges we are facing. At last, we summarize our proposed approaches, solutions, as well as our contributions.

In Chapter 2, we undertake a comprehensive literature review on the research topics related to our work. There are three sections, each of which summarizes the literatures related to one of our research tasks. In the first section, we review the related work about textual event detection/tracking and text indexing. In the second section, we discuss the techniques related to topic model for social event

detection and representative image selection. In the third section, we compare the social community profiling methods from three aspects: joint community profiling and detection, heterogeneous user interactions and diffusion granularity level.

In Chapter 3, we study the problem of textual event monitoring and indexing. We present a novel stream clustering method along with a multi-layer inverted event indexing structure to efficiently and effectively monitor the evolving events from social text streams. To capture the dynamics of social events over time, we design four event operations including creation, absorption, split and merge to the single pass incremental clustering algorithm. In terms of the large-scale highly dynamic social events, we propose a multi-layer event indexing structure named MIL to accelerate the event evolution monitoring process. The MIL is designed in a way that more relevant yet shorter event lists are quickly found at the lowest layer, hence searching longer event lists at the upper layers can be largely avoided by our proposed pruning strategy. We verify both the effectiveness of our stream clustering algorithm and the efficiency of our multi-layer event indexing structure by an extensive set of experiments on real-world tweet datasets.

In Chapter 4, we propose a novel topic model which exploits various social data features in addition to textual contents for effective social event tracking and visualization. We thoroughly study images in social networks, especially their effects on event detection and tracking performance. To deal with the large amount of noisy social images, we design a two-filter strategy, which first removes images that fall into the predefined stop image category, then filters the general images that frequently and uniformly appearing in many events and thus bring negative effect on event detection. Extensive experiments conducted on a real-world tweet dataset demonstrate the effectiveness of our method from different aspects.

In Chapter 5, we propose a novel social community profiling problem for social event analysis. Given a social data stream consists of user links, user-generated contents and diffusion links, we aims to detect communities from the social data stream and provide each community a content profile (the event topic distribution of a community) and a diffusion profile (a distribution of the information diffusion strength from one community to another on a certain topic). We design a novel generative model named CP-JHC which jointly detects and profiles communities from social data. Different user interactions such as user links (i.e., following relationships) and diffusion links (i.e.,

retweets) are modelled heterogeneously. Furthermore, we comprehensively consider individual factor, community factor and temporal topic popularity factor for the diffusion modeling. We implement a multi-threading CP-JHC to guarantee the efficiency of our method. The effectiveness of our method is verified by extensive experiments conducted on two large real-world datasets.

6.2 Future Directions

The investigation on the problem of multi-perspective social event analysis is still far from finished. The new applications bring new perspectives and new opportunities.

6.2.1 Personalized Event Detection

In this work, we detect and monitor the general events from the entire social data streams. In reality, people have their own topics of interest and preferences for the events that attract their attentions. How to adapt event detection to a specific user still remains open and challenging. In the third task in this thesis, we have derived the user profile from both individual and community level by comprehensively modeling user links, user generated contents and diffusion links. These user profiles can be utilized as users' interests to support personalized event recommendation. There have been some attempts for personalized event detection in the literature [20, 57]. However, the users' interests are either pre-defined context or directly derived from the data, which is not always available in real life. It will be interesting and promising to recommend the social events to users based on the automatically detected user profiles.

6.2.2 Temporal-aware Social Data Analysis

Time is one of the most important feature of social data and is of great importance for both event detection and community profiling. Since social data is updated quickly, both social events and social communities detected from social data are highly dynamic. Considering the temporal factor helps to monitor the evolving events and communities more accurately. Moreover, with the large amount of posts generated every day, it is impossible to store and manage all the data in main memory. How to

reduce the I/O cost is crucial in the social data analysis area. Take event detection task as an example, the popular events and communities are normally frequently accessed when their follow-up posts arrive. However, following the nature of the event life circle, events normally fade out after a certain period. Faded events are less likely to be accessed. Given this observation and inspired by [87], we plan to propose a temporal-aware event monitoring model which stores the recent active events in the main memory and faded events on the hard disk to reduce the number of random accesses to the hard disk. The fading probability of the events in memory are checked, and idle or faded events are moved to the hard disk.

6.2.3 Distributed Event Monitoring and Community Profiling

In view of the large amount and quickly updated social data stream, it is crucial to develop the social event analysis algorithms that are efficient and scalable. Spark is a fast in-memory cluster computing framework which supports large-scale data processing. It has been the talk of the Big Data town for a while [74]. Generally speaking, Spark extends the MapReduce model to efficiently support more types of computations such as interactive queries and stream processing and it is designed to cover a wide range of workloads that previously required separate distributed systems [32]. We intend to deploy our existing social event analysis techniques in the Spark distributed platform to support scalable and efficient multi-perspective social event analysis from big social data.

References

[1] H. Abdelhaq, C. Sengstock, and M. Gertz. Eventweet: Online localized event detection from twitter. *PVLDB*, 6(12):1326–1329, 2013.

[2] C. C. Aggarwal and K. Subbian. Event detection in social streams. In *SDM*, pages 624–635, 2012.

[3] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *SIGIR*, pages 37–45, 1998.

[4] A. Angel, N. Koudas, N. Sarkas, D. Srivastava, M. Svendsen, and S. Tirthapura. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *VLDB J.*, 23(2):175–199, 2014.

[5] F. Atefeh and W. Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 2013.

[6] A. C. Awekar and N. F. Samatova. Fast matching for all pairs similarity search. In *Web Intelligence*, pages 295–300, 2009.

[7] H. Becker, M. Naaman, and L. Gravano. Learning similarity metrics for event identification in social media. In *WSDM*, pages 291–300, 2010.

[8] H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: Real-world event identification on twitter. In *ICWSM*, 2011.

[9] B. Bi, B. Kao, C. Wan, and J. Cho. Who are experts specializing in landscape photography?: Analyzing topic-specific authority on content sharing services. In *Proc. of the 20th*

ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pages 1506–1515, 2014.

[10] J. Bian, Y. Yang, and T.-S. Chua. Multimedia summarization for trending topics in microblogs. In *CIKM*, pages 1807–1812, 2013.

[11] J. Bian, Y. Yang, and T.-S. Chua. Predicting trending messages and diffusion participants in microblogging network. In *SIGIR*, pages 537–546, 2014.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[13] M. Busch, K. Gade, B. Larson, P. Lok, S. Luckenbill, and J. Lin. Earlybird: Real-time search at twitter. In *ICDE*, pages 1360–1369, 2012.

[14] J. Chang and D. M. Blei. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, 4(1):124–150, 2010.

[15] M. Chang and C. K. Poon. Efficient phrase querying with common phrase index. In *ECIR*, pages 61–71, 2006.

[16] C. Chen, F. Li, B. C. Ooi, and S. Wu. Ti: an efficient indexing mechanism for real-time search on tweets. In *SIGMOD*, pages 649–660, 2011.

[17] N. Chen, J. Zhu, F. Xia, and B. Zhang. Generalized relational topic models with data augmentation. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1273–1279, 2013.

[18] T. Chen, D. Lu, M.-Y. Kan, and P. Cui. Understanding and classifying image tweets. In *ACM MM*, pages 781–784, 2013.

[19] Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL '15*, 2015.

[20] M. Ciglan and K. Nørvåg. Wikipop: personalized event detection system based on wikipedia page view statistics. In *CIKM '10*, pages 1931–1932, 2010.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. Ser. B*, 39:1–38, 1977.

[22] Q. Diao, J. Jiang, F. Zhu, and E.-P. Lim. Finding bursty topics from microblogs. In *ACL*, pages 536–544, 2012.

[23] N. Du, Y. Liang, M. Balcan, and L. Song. Influence function learning in information diffusion networks. In *ICML '14*, pages 2016–2024, 2014.

[24] M. Eftekhar, Y. Ganjali, and N. Koudas. Information cascade at group scale. In *KDD '13*, pages 401–409, 2013.

[25] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, pages 102–113, 2001.

[26] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 150–160, 2000.

[27] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, 1984.

[28] D. Glenn. Is the status update dead? 36% of tweets are photos [infographic]. Available: `http://www.adweek.com/socialtimes/is-the-status-update-dead-36-of-tweets-are-photos-infographic/104250?red=st`. Accessed: 2016-06-22.

[29] M. Granovetter. The strength of weak ties: a network theory revisited. *SOCIOL THEOR*, 1(1):211233.

[30] A. Gruenheid, X. L. Dong, and D. Srivastava. Incremental record linkage. *PVLDB*, 7(9):697–708, 2014.

[31] H. Gu, X. Xie, Q. Lv, Y. Ruan, and L. Shang. Etree: Effective and efficient event modeling for real-time online social media networks. In *Web Intelligence*, pages 300–307, 2011.

[32] M. Hamstra, H. Karau, M. Zaharia, A. Konwinski, and P. Wendell. *Learning Spark: Lightning-Fast Big Data Analytics*. O'Reilly, 2015.

[33] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[34] Y. Han and J. Tang. Probabilistic community and role model for social networks. In *KDD '15*, pages 407–416, 2015.

[35] W. H. Hsu, L. S. Kennedy, and S.-F. Chang. Video search reranking through random walk over document-level context graph. In *ACM MM*, pages 971–980, 2007.

[36] Z. Hu, J. Yao, B. Cui, and E. Xing. Community level diffusion extraction. In *Proc. of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1555–1569, 2015.

[37] Z. Huang, B. Hu, H. Cheng, H. T. Shen, H. Liu, and X. Zhou. Mining near-duplicate graph for cluster-based reranking of web video search results. *ACM Trans. Inf. Syst.*, 28(4):22:1–22:27, 2010.

[38] A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *KDD '06*, pages 207–216, 2006.

[39] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[40] D. Jiang, K. W. Leung, J. Vosecky, and W. Ng. Personalized query suggestion with diversity awareness. In *ICDE*, pages 400–411, 2014.

[41] Y. Jie, L. Andrew, C. Mark, R. Bella, and P. Robert. Using social media to enhance emergency situation awareness. *IEEE Intelligent Systems*, 27(6):52–59, 2012.

[42] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, May 2004.

[43] A. Khosla, B. An, J. J. Lim, and A. Torralba. Looking beyond the visible scene. In *CVPR*, pages 3710–3717, 2014.

[44] A. Lazar, D. Abel, and T. Vicsek. Modularity measure of networks with overlapping communities, 2009.

[45] P. Lee, L. V. S. Lakshmanan, and E. E. Milios. Incremental cluster evolution tracking from highly dynamic network data. In *ICDE*, pages 3–14, 2014.

[46] R. Lee and K. Sumiya. Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection. In *GIS-LBSN*, pages 1–10, 2010.

[47] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, pages 631–640, 2010.

[48] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *CIKM*, pages 155–164, 2012.

[49] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang. Tedas: A twitter-based event detection and analysis system. In *ICDE*, pages 1273–1276, 2012.

[50] R. Li, S. Wang, H. Deng, R. Wang, and K. C.-C. Chang. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *KDD '12*, pages 1023–1031, 2012.

[51] S. Lin, F. Wang, Q. Hu, and P. S. Yu. Extracting social events for learning better information diffusion models. In *KDD '13*, pages 365–373, 2013.

[52] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher. Metafac: Community discovery via relational hypergraph factorization. In *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 527–536, 2009.

[53] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang. Mining topic-level influence in heterogeneous networks. In *CIKM '10*, pages 199–208, 2010.

[54] Y. Liu, A. Niculescu-Mizil, and W. Gryc. Topic-link lda: Joint models of topic and author community. In *Proc. of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 665–672, 2009.

[55] R. Long, H. Wang, Y. Chen, O. Jin, and Y. Yu. Towards effective event detection, tracking and summarization on microblog data. In *WAIM*, pages 652–663, 2011.

[56] B. Lucier, J. Oren, and Y. Singer. Influence at scale: Distributed computation of complex contagion in networks. In *KDD '15*, pages 735–744, 2015.

[57] A. Q. Macedo, L. B. Marinho, and R. L. Santos. Context-aware event recommendation in event-based social networks. In *RecSys '15*, pages 123–130, 2015.

[58] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[59] 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 *CHI*, pages 227–236, 2011.

[60] K. Massoudi, M. Tsagkias, M. de Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR*, pages 362–367, 2011.

[61] J. J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems 25*, NIPS '12, pages 548–556, 2012.

[62] R. McCreadie, C. Macdonald, I. Ounis, M. Osborne, and S. Petrovic. Scalable distributed event detection for twitter. In *Int. Conf. on Big Data*, pages 543–549, 2013.

[63] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM*, pages 469–478, 2008.

[64] T. Minka. Estimating a Dirichlet distribution. Technical report, MIT., 2000.

[65] J. Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.*, 5(1):32–38, 1957.

[66] D. Newman, P. Smyth, and M. Steyvers. Scalable parallel topic models. *Journal of Intelligence Community Research and Development*, 2006.

[67] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci*, 103(23):8577–8582, 2006.

[68] K.-H. Nguyen, X. Tannier, O. Ferret, and R. Besançon. Generative event schema induction with entity disambiguation. In *ACL '15*, pages 188–197, 2015.

[69] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724, 2014.

[70] M. Osborne, S. Moran, R. McCreadie, A. Von Lunen, M. Sykora, E. Cano, N. Ireson, C. Macdonald, I. Ounis, Y. He, T. Jackson, F. Ciravegna, and A. O'Brien. Real-time detection, tracking, and monitoring of automatically discovered events in social media. In *ACL '14*, 2014.

[71] M. Pitts, S. Savvana, S. B. Roy, and V. Mandava. ALIAS: author disambiguation in microsoft academic search engine dataset. In *EDBT*, pages 648–651, 2014.

[72] N. G. Polson, J. G. Scott, and J. Windle. Bayesian inference for logistic models using pólya-gamma latent variables. *Journal of the American Statistical Association*, pages 1339–1349, 2013.

[73] A.-M. Popescu, M. Pennacchiotti, and D. Paranjpe. Extracting events and event descriptions from twitter. In *WWW*, 2011.

[74] Z. Qiu, B. Wu, B. Wang, C. Shi, and L. Yu. Collapsed gibbs sampling for latent dirichlet allocation on spark. In *JMLR '14*, pages 17–28, 2014.

[75] D. Ramage, S. T. Dumais, and D. J. Liebling. Characterizing microblogs with topic models. In *ICWSM*, pages 130–137, 2010.

[76] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*, 2014.

[77] S. D. Roy, T. Mei, W. Zeng, and S. Li. Socialtransfer: Cross-domain transfer learning from social streams for media applications. In *ACM MM*, pages 649–658, 2012.

[78] Y. Ruan, D. Fuhry, and S. Parthasarathy. Efficient community detection in large networks using content and links. In *Proc. of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1089–1098, 2013.

[79] M. Sachan, D. Contractor, T. A. Faruquie, and L. V. Subramaniam. Using content and interactions for discovering communities in social networks. In *Proc. of the 21st International Conference on World Wide Web*, WWW '12, pages 331–340, 2012.

[80] M. Sachan, A. Dubey, S. Srivastava, E. P. Xing, and E. Hovy. Spatial compactness meets topical consistency: Jointly modeling links and content for community detection. In *Proc. of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 503–512, 2014.

[81] T. Sakaki, M. Okazaki, and Y. Matsuo. Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Trans. Knowl. Data Eng.*, 25(4):919–931, 2013.

[82] B.-S. Seah, S. S. Bhowmick, and A. Sun. Prism: Concept-preserving social image search results summarization. In *SIGIR*, pages 737–746, 2014.

[83] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[84] Y. Sun, C. C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *Proc. VLDB Endow.*, 5(5):394–405, Jan. 2012.

[85] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD '08*, pages 990–998, 2008.

[86] L. Tang, X. Wang, and H. Liu. Uncovering groups via heterogeneous interaction analysis. In *ICDM '09*, 2009.

[87] M. Terrovitis, S. Passas, P. Vassiliadis, and T. Sellis. A combination of trie-trees and inverted files for the indexing of set-valued attributes. In *CIKM*, pages 728–737, 2006.

[88] S. Unankard, X. Li, and M. Sharaf. Emerging event detection in social networks with location sensitivity. *World Wide Web*, pages 1–25, 2014.

[89] B. Wang, C. Wang, J. Bu, C. Chen, W. V. Zhang, D. Cai, and X. He. Whom to mention: Expand the diffusion of tweets by @ recommendation on micro-blogging systems. In *WWW '13*, pages 1331–1340, 2013.

[90] X. Wang and A. McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *KDD*, pages 424–433, 2006.

[91] Z. Wang, P. Cui, L. Xie, H. Chen, W. Zhu, and S. Yang. Analyzing social media via event facets. In *ACM MM*, pages 1359–1360, 2012.

[92] J. Weng and B.-S. Lee. Event detection in twitter. In *ICWSM*, pages 401–408, 2011.

[93] C.-C. Wu, T. Mei, W. H. Hsu, and Y. Rui. Learning to personalize trending image search suggestion. In *SIGIR*, pages 727–736, 2014.

[94] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proc. of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 505–516, 2012.

[95] J. Yang, J. J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM' 13*, pages 1151–1156, 2013.

[96] Y. Yang, Z. Zha, Y. Gao, X. Zhu, and T. Chua. Corrections to "exploiting web images for semantic video indexing via robust sample-specific loss". *IEEE Trans. Multimedia*, 17(2):256, 2015.

[97] H. Yin, B. Cui, L. Chen, Z. Hu, and Z. Huang. A temporal context-aware model for user behavior modeling in social media systems. In *SIGMOD*, pages 1543–1554, 2014.

[98] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR*, pages 338–349, 2011.

[99] X. Zhou and L. Chen. Event detection over twitter social media streams. *VLDB J.*, 23(3):381–400, June 2014.

[100] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *Proc. of the VLDB Endowment*, 2(1):718–729, Aug. 2009.

[101] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen. Sparse hashing for fast multimedia search. *ACM Trans. Inf. Syst.*, 31(2):9, 2013.

[102] X. Zhu, Z. Huang, J. Cui, and H. T. Shen. Video-to-shot tag propagation by graph sparse group lasso. *IEEE Transactions on Multimedia*, 15(3):633–646, 2013.

[103] X. Zhu, Z. Huang, Y. Yang, H. T. Shen, C. Xu, and J. Luo. Self-taught dimensionality reduction on the high-dimensional small-sized data. *Pattern Recognition*, 46(1):215–229, 2013.

[104] X. Zhu, X. Li, and S. Zhang. Block-row sparse multiview multilabel learning for image classification. *IEEE T. Cybernetics*, 46(2):450–461, 2016.

[105] X. Zhu, X. Li, S. Zhang, C. Ju, and X. Wu. Robust joint graph sparse coding for unsupervised spectral feature selection. 2016.

[106] X. Zhu, L. Zhang, and Z. Huang. A sparse embedding and least variance encoding approach to hashing. *IEEE Transactions on Image Processing*, 23(9):3737–3750, 2014.

[107] Y. Zhu, X. Yan, L. Getoor, and C. Moore. Scalable text and link analysis with mixed-topic link models. In *KDD '13*, pages 473–481, 2013.

[108] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.