LOCATION ESTIMATION AND GEO-CORRELATED INFORMATION TRENDS

Zhi Liu

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2017

APPROVED:

Yan Huang, Major Professor
Eduardo Blanco, Committee Member
Wei Jin, Committee Member
Rodney D. Nielsen, Committee Member
Barrett R. Bryant, Chair of the Department of Computer Science and Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Victor Prybutok, Dean of the Toulouse Graduate School

Liu, Zhi. *Location Estimation and Geo-Correlated Information Trends*. Doctor of Philosophy (Computer Science and Engineering), December 2017, 129 pp., 15 tables, 101 numbered references.

A tremendous amount of information is being shared every day on social media sites such as Facebook, Twitter or Google+. However, only a small portion of users provide their location information, which can be helpful in targeted advertising and many other services. Current methods in location estimation using social relationships consider social friendship as a simple binary relationship. However, social closeness between users and structure of friends have strong implications on geographic distances. In the first task, we introduce new measures to evaluate the social closeness between users and structure of friends. Then we propose models that use them for location estimation. Compared with the models which take the friend relation as a binary feature, social closeness can help identify which friend of a user is more important and friend structure can help to determine significance level of locations, thus improving the accuracy of the location estimation models. A confidence iteration method is further introduced to improve estimation accuracy and overcome the problem of scarce location information. We evaluate our methods on two different datasets, Twitter and Gowalla. The results show that our model can improve the estimation accuracy by 5% - 20% compared with state-of-the-art friend-based models.

ACKNOWLEDGMENTS

First I would like to express my special appreciation and thanks to my advisor Prof. Yan Huang. During the last five years, Prof. Huang continuously supported me on the study, research, and my work. Her knowledge and methodology greatly helped me in selecting research topics, solving problems, and publishing papers. She provides lots of suggestions in every step of a research topic, how to define a problem, how to explain it clearly, how to find out the key point, and how to finish the work efficiently.

Besides my advisor, I also would like to thank my committee members: Prof. Nielsen, Prof. Jin, and Prof. Blanco, for their insightful comments and suggestions. Prof. Nielsen gave me many suggestions on my projects and research topics. He helped me have a better understanding of different related research topics. I also want to express my appreciation to Prof. Jin and Prof. Blanco for both of them would like to join my committee in the last step and spend time to help in my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

A tremendous amount of information is being shared every day on social media platforms such as Facebook, Twitter, and Google+. For example, there are 310 million average monthly active users on Twitter[1] have published 300 million tweets worldwide, and this number continues to increase at a rate of 5,700 tweets per second [81]. Oftentimes these messages include geo-information that is valuable to others, such as activities (*e.g.*, art fairs, jazz festivals, and social gatherings), natural disaster occurrences (*e.g.*, tornadoes, earthquakes), or other incidents (*e.g.*, traffic jams). Twitter's popularity carries it beyond a communication platform. With its users widely distributed, people are not only tweeting about their daily activities but can also report events happening anywhere and spread through the network. The analytics on social media content can provide most recent topic trends, real-life events, and public sentiments. Compared with traditional information source, social media are informal but can provide the most up-to-date information on location information, current events including earthquakes, sports matches, and a variety of kinds of events and news.

The goal of the first task is developing effective algorithms to estimate user location. The results will help to extract more useful information from social media which in turn can assist the assimilation of social media information of interest for application domains such as smart transportation, disaster relief and recovery, and national security. However, in the home location estimation, we face the following challenges: 1) People can share their location information more easily nowadays. Paradoxically, the problem of lacking location information still exists. Only 30% of users provide their location information to at least one social media account and 46% of teen app users have turned off the location tracking feature on their cell phone.[2]; 2) User behavior varies greatly among different social networking services. In the datasets used in this work, 27% of friend pairs on Gowalla locate within

[1] https://investor.twitterinc.com/results.cfm

[2] http://www.pewinternet.org/2013/09/12/location-based-services/

100km of each other. In Twitter, however, this ratio is only 12%. This because Twitter is not mainly a location-based social networking service and users tend to follow various media sources that are far away, and 3) Social media information is noisy and mixed with meaningless information. For example, 40% tweets are not associated with a particular subject [76]. Additionally, some users are global travelers and have many friends from many cities around the world.

Previous methods use the social network, user location, and the content information for location estimation. However, the content information is not always available on different social media tools. In this work, we focus on the analysis of social network and how the user locations affect the social connections. Our study shows that features such as friend structure of a user are important in improving the accuracy of the location estimation. This work makes the following contributions: 1) We study the geographic features of social networks. We propose measures of social closeness between friends and the tightness of the friend structure of a user. We study the relationship between the closeness/tightness and the geographic distance. Existing algorithms consider friend relation as a binary relationship. However, finer level features such as friend co-location can help determine the social closeness of two friends. Friend co-location is an index measuring how overlapped two users' friend distributions are. Statistics show that the friend co-location has a significant influence on the probability of two friends located close to each other. A user typically has a tight social structure among his friends in his home location. Local social coefficient measures the tightness of a user's friends in an area and can be a good indicator to measure if the user is located in that area. 2) We propose three user location estimation models which take social closeness and tightness of friend structure into consideration. 3) To deal with the challenge of location sparsity, we propose a confidence-based iteration model in location estimation which significantly improved the estimation accuracy. 4) We evaluate our models using two real-world datasets. Extensive experimental results show that our methods improve the estimation accuracy by 5%- 20% compared with the state-of-the-art algorithm.

Then we focus on local event detection on social media platforms. Social media has

rapidly become one of the most important platforms where people can share their thoughts, opinions, interests, and whereabouts. A typical example is Twitter, an online social networking service that enables users to send and read short 140-character messages. Twitter's popularity carries it beyond a communication platform. In this task, we try to detect local events from geotagged tweets. Monitoring local events is conducive to many real-world applications such as crime mapping, traffic monitoring, and emergency management. With location, time, and content analysis, one can explore a more comprehensive description of the events from different points of view and have a better understanding of the influence of an event. More specifically, we not only focus on detecting large events such as shows or sports matches but also try to find small-scale events. In fact, many local events in the real world are "small" events. They are not reported by news media and only a relatively small number of local people discuss the events. However, these events are important and learning such events can help in location-based services, public security, and smart transportation. Finding a system to detect these more difficult events is the focus of this work.

For detecting local events, we have the following challenges: 1) There is no standard format for Twitter users to describe an event. People discuss events on social media from different perspectives with some expressing their opinions and others stating facts in a very informal way, and most of the users will not list a clear time and location information of an event; 2) Collecting the tweets related to an event is challenging. Around 6,000 tweets are published every second [3], but only a very small percentage of them are related to an event. 40.55% tweets are pointless babbles[4]. Based on our analysis of the real dataset, only less than 5% geotagged tweets are related to an event. Users typically have different linguistic expression preference, making it difficult to filter tweets related to an event; 3) For many local events, there may be only a few users publishing a small number of related tweets. Existing methods based on outbreak detection of tweets or terms can only find large events such as sports games, shows, or disasters, which may be already on news and attract lots of

[3] http://www.internetlivestats.com/twitter-statistics/

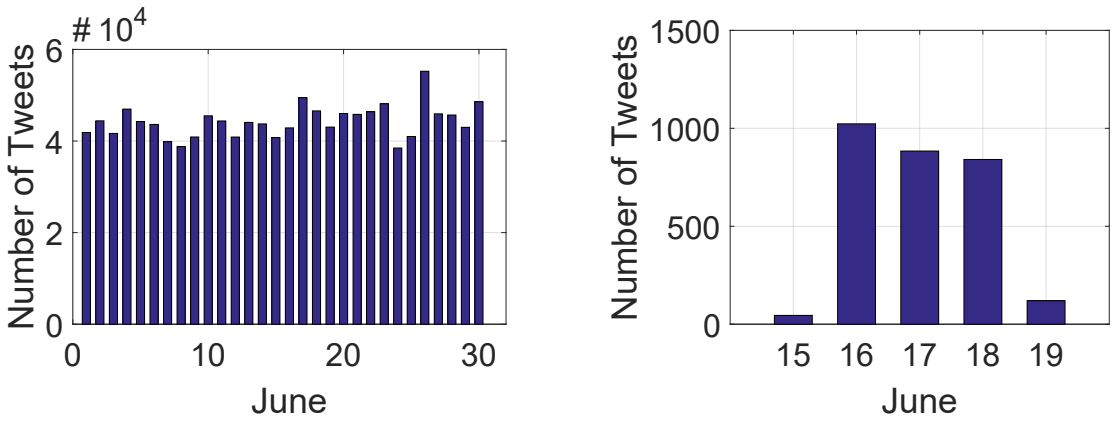
[4] http://pearanalytics.com/wp-content/uploads/2009/08/Twitter-Study-August-2009.pdf

attention. For events such as a small meeting, party, minor show, or local festivals in a small town, users discuss these locally and typically will not lead to an outbreak of the number of tweets.

To address these challenges, we propose the LEDS framework to detect **local** events. The goal is to be sensitive to all kinds of local events no matter if they are large or small. Many existing algorithms focus on larger events and detect events in limited categories, i.e. sports, shows, and disasters. Such methods use spikes of tweets or located users to detect events [53, 23, 74, 50]. However, they tend to ignore many small-scale events since most medium- or small-scale events cannot lead an obvious changing of the number of tweets in an area. For example, in Figure 1(a), we show the number of tweets we collected from Los Angeles in June 2015. The total number of tweets of each day is fairly stable and it will be difficult to detect an outbreak to extract events. In Figure 1(b), we show the number of tweets we collected from Los Angeles which contain the term of "E3" (Electronic Entertainment Expo) for the same time period. The total number of tweets from Los Angeles does not spike during E3, making it difficult to detect events from spikes of tweets. There are many events of this nature as shown in our experiments later.

Terms are more sensitive to events than tweets. During an event, people are likely to use special terms which are not commonly used in daily lives. Thus we propose an anomaly-term-driven framework. The key idea is to compare term frequencies with overall distribution. The comparison needs to be done based on spatial and time scales instead of globally. Our method can detect the abnormal distribution effectively and is not affected by factors such as the flux of online users. In this way, our proposed system can detect much broader event categories which include many local events such as parties, and personal meetings.

The goal of the third task is to detect the associations between hashtags (topics) by analyzing the time and location distributions of tweets which contain these hashtags. The analysis of Twitter data can help to predict or explain phenomena in many real-world applications [2, 8]. Topics on Twitter may potentially be associated with space and time.



(a) Number of tweets in June

(b) Number of tweets of E3

FIGURE 1.1. Detailed features of detected events.

For example, the hashtags related to the Independence Day $\#july4th$ are closely related to the hashtags $\#firework$ and $\#beer$ in the beginning of July, which directly reveals how people celebrate the Independence Day. The detection of such topic associations can help to improve online services such as news recommendation and explain the relationship among real-world events.

In this work, we define the topic set of each tweet as the hashtags it contains. So each topic can appear in a number of tweets and each tweet can contain several topics. For example, the right side of Figure 1.2 shows two tweets. The first tweet is related to the topic/hashtag $\#Amazon$ and the second one contains two topics $\#Job$ and $\#Houston$. The left side of Figure 1.2 shows three groups of hashtags represented by three colors. Using these relationships, when a user talks about certain topics, such as $\#job$, we can recommend the tweets from other related topics based on the topic associations (e.g. $\{\#job, \#hiring, \#tweetmyjobs\}$) so that the users can get more comprehensive information about what they want. Such close correlated topic associations can also reflect the relations among real-world events. For example, the mining process reveals a topic association of $\{\#nbafinals, \#cle, \#gsw\}$. As a matter of fact, hashtags $\#cle$ and $\#gsw$ are the two teams in the NBA finals 2015. The mining process reveals these three topics appearing in space and time proximity.

FIGURE 1.2. Closeness of topics and topic related tweets

We define the closeness among topics by examining the co-occurrence of tweets containing these topics. Closely related topics may tend to be talked about the similar time and region. Based on this idea, we define the participation index of topic associations. If the tweets of a set of topics tend to be found at the same time and region, they will have a higher participation index. We evaluate this concept in our experiments by comparing the measuring results of participation index with the human judgment. Based on the concept of participation index, we propose three different types of queries to help users to extract topic associations with different semantics in different time-region frames.

One of the challenges is that the number of tweets can be very large as well as the topics and their combinations. For example, in the sample we collected, there are 364,333 geo-tagged tweets that contain the hashtag $\#job$ in the U.S. in July 2015. So, how to develop an efficient algorithm to help to calculate the participation index of a set of topics will be the most important step. Here we propose a multi-layer geographic index and a time index to help us to filter the topic sets under the threshold quickly to calculate the participation index.

Another challenge in our work is the number of topics. Our dataset contains 492,492 hashtags. Any subset of these hashtags can be a potential topic association. To deal with the exponential nature of the hashtag associations, we propose two methods to optimize the mining algorithm: topic filtering and topic combination. In the topic filtering, we will remove some topics which are not affected by other topics or real-world events. These topics participate in a large number of associations and the results are meaningless. Furthermore,

we propose an algorithm to combine similar topics to further reduce the complexity.

In this dissertation, we will introduce the related work in chapter 2. The three tasks will be discussed in chapters from 3 to 5. For each task, the algorithms used to solve it and the experimental results will be discussed in detail. Finally, we summarize our work in chapter 6.

CHAPTER 2

RELATED WORK

In this section, we introduce the related work and provide a brief comparison between our work and the related research.

2.1. Community Detection

In the past decade, many algorithms have been developed to detect communities in a network. For a complete discussion of various algorithms, please refer to [32]. Aaron *et al.* provide a hierarchical clustering approach to detect communities using internal density in [26]. The internal density is the number of edges inside a community in a network. The basic idea is to increase the ratio of the edges in communities during the hierarchical clustering process using Equation 2.1:

$$Q = \frac{1}{2m} \sum_{vw} [A_{vw} - \frac{k_v k_w}{2m}] \delta(c_v, c_w) \tag{2.1}$$

where A_{vw} is the adjacency matrix of the network and k_v is the degree of node v. c_v represents the community of node v and $\delta(c_v, c_w)$ is 1 if $c_v = c_w$. m is the number of edges in the whole network G. So the value Q will be large when more edges are inside a community, which represents a good division of the work. To avoid the problem that the largest Q value 1 will only happen when all nodes belong to the same community, the authors introduce the component $k_v k_w / 2m$ in the modularity of Q. $k_v k_w / 2m$ is the probability of an edge existing between nodes v and w if edges were randomly placed. So Q will be close to zero when the network is randomly generated without community structure. Some other work is also based on modularity optimization such as [13], [70], and [52].

Another popular algorithm [63] is based on iteratively removing "unimportant" edges. The basic assumption of this method is that communities are weakly connected by a few edges. The importance of an edge, called betweenness score, is the number of shortest paths that go through that edge. The paths between different communities must go through

an edge across communities so the edges across communities will get a higher betweenness score. The edge with the highest score will be removed from the network iteratively. In [41], the authors define the similarity between nodes using their degrees and the number of the common neighborhood. The sum of the similarities of edges inside or outside a community was defined as internal or external similarity of a community.

In the last few years, some researchers have studied the geographic constraints on real-world networks. In [64], the authors build a network based on the cell phone communication records. Then they study the relationship between distance and the call/text tie probability. By dividing the network into communities[67, 32, 62, 35], the authors show that the geographic span of real-world community is much smaller than the null community especially when the community has less than 30 people. In [77], the authors define the concepts of node locality and geographic clustering coefficient. Then they show the value distribution of these two coefficients with respect to the degree of nodes. The node locality is slowly decreasing with node degree increases. Their study shows that people tend to build connections with other nearby users. Some users have social connections only with others within a close geographic distance.

The most relevant work to ours is proposed in [86]. Yves *et. al.* propose a geosocial communities detection method. The authors assign each edge with a similarity score using social relationship and the Euclidean distance between their average stop locations and then run the spectral clustering algorithm.

However, the authors only built their model on a small scale application and didn't provide evidence on how the social relation is influenced by the geographic location, which is important for using geographic information in community detection in location-tagged networks. In addition, the efficiency of the spectral clustering algorithm may be the main bottleneck when dealing with a large dataset. In this dissertation, we push the location information of nodes in networks into the community detecting algorithm and design an efficient algorithm that can scale to large networks.

2.2. Location Estimation

Recent research on location estimation in social networks follows two directions based on the data used: content-based and social-network-based. Content-based prediction models extract location information, like venue signals, from content provided by users. Cheng et al. [24] proposed and evaluated a city-level location estimation model of Twitter users purely by taking the location related words in tweet content as features and applying a classification method. Chandra et al. [22] improved the content based method by using user interactions and exploiting the relationship between different tweet message types. They also provided the estimation of the top-K probable cities for a user. Another similar method is proposed in [29]. When a user declares a place, it will be checked in gazetteer to see if it corresponds to a city name. Location information will then be applied to infer the user location by the Twitter network. Content-based methods have also been studied to solve the web page geotagging problem [6]. The authors extract the toponyms from the web page to predict its location.

Our work is closely related to the work by Backstrom et al. [9]. By modeling the relation between distance and probability of being a friend, the authors proposed a general formula to calculate the probability of a user located at a specific place. Places with the maximal probability will be estimated as the location of the user. Both [73] and [25] aimed to build a user mobility model by using the location of their friends. Cho et al. [25] used several factors in their probability model, including check-in records, social network, friends' location, and time. Sadilek et al. [73] applied a machine learning method with similar information. Li et al. [56] proposed the UDI (unified discriminative influence) framework to combine content and friends' location analysis in a unique model to profile users' home location. In [59], the authors extract features from users' tweets content, social relation, and other behaviors like geotag to infer the home location on different level. Based on the geotagged information, they also built a classifier to predict whether a user is traveling. In [46], the author applied the geometric median between users, Oja's Simplex Median, and the transitivity of the social network in the method. The goal of the algorithm is to select the

nearest neighbor as the estimation result. In [27], the authors use the number of @mentions to indicate the social ties and take it as the weight ω_{ij} between users u_i and u_j. For all the connected users, they define the total variation as $\sum_{ij} \omega_{ij} d_{ij}$ where d_{ij} is the geographic distance between them. When estimating the users' locations, they seek the solution that makes the sum as small as possible. However, all of those models take the friend relation as a binary feature, i.e. being friends or not. This premise does not allow those models to take advantage of all the information from the social network. In our model, we take the social relation as a continuous feature by introducing the concept of social closeness. Network structure and locations are taken into consideration compared with our previous work [49] to help to achieve better results. By studying and using the relation between social closeness and geographic distance, our model: 1) significantly improves the estimation accuracy, especially when people have a small number of friends, 2) overcomes the problem that only a small number of users provide their location information.

The availability of user geotagged tweets allows one to profile the geographic distribution of a term in a tweet and this distribution has been used in several existing works for geotagging. The bayesian-based method generates a language model for a location and calculates the probability of a tweet belonging to different locations based on the Bayesian model [48]. Given the local language model θ_L, the probability of a tweet T belonging to a location L can be calculated as: $P(L|T) = \frac{P(T|\theta_L)P(L)}{P(T)}$ and $P(T|\theta_L) = \prod_i P(t_i|\theta_L)$. The location with the maximal probability is assigned to a tweet as its geotag. Kullback-Leibler (KL) divergence based method measures the difference between a tweet and the local language model [48, 72]. The location with minimal KL value is assigned to the tweet. Gaussian model or Gaussian Mixture Model (GMM) fits geotags of n-grams in tweets into Gaussian distributions. The center of the Gaussian distribution [31] or a weighted sum of GMMs [68] is assigned as the geotag of tweets.

Another line of work uses location relevant words from contents as an important clue. In [14], local geo-parsing is considered to suggest locations . Geotagging is done through classification. Location indicative words (LIWs) in the content are detected mainly by three

measurements: term frequency, inverse city frequency, and information gain. These LIWs are used as features to classify tweets into cities. All of the above approaches rely solely on social media content for geotagging. However, pure content-based methods leave out the movement patterns of a user. The sequence of geotags of tweets published by a user is aligned with the location sequence of the same user. The movement of a user is affected by various factors such as home location, travel distance, recent activities, and the characteristics of cities. Using user movement patterns can potentially help improve the geotagging accuracy.

2.3. Event Detection

Event detection on social media has attracted much attention in recent years. The approaches used can be broadly classified into two categories: geographic-anomaly-detection-driven methods and content-feature-driven methods.

The basic idea of the methods in the first category is to detect events by monitoring abnormal occurrences of tweets or terms. Weng et al. [93] build signal models for individual words by applying wavelet analysis to the frequency-based raw signals of the words and capture the bursts in the words' appearance. The authors then detect events by grouping a set of words with similar patterns of burst using a community detection algorithm. Krumm et al. [50] use regression to estimate the number of tweets of an area. The features used include time of the day, the day of the week, and tweets counts from neighbors. If the actual number of tweets exceeds the threshold, a text summarization algorithm will be used on the tweets in the region to extract tweets to describe the event.

In [53], the authors detect events by monitoring the moving crowd. Both increasing of tweets and Twitter users are considered in event detection. To monitoring moving crowd, the number of users in a region, incoming users, and outgoing users are recorded. These abnormal cases will be investigated and used in event detection. Cheng et al. [23] propose a space-time scan statistics-based approach in event detection. The idea is to detect clusters on both space and time dimensions assuming people will tweet more than expected during the event. In [7], the authors propose a user-driven method to perform a geo-temporal analysis of information detect events. The DBSCAN clustering algorithm is used to identify

geo-temporal clusters of messages. In [21], the authors present a visual analytics approach to provide users with scalable and interactive social media data analysis and visualization. An analyst can extract topics from a set of messages and find unusual peaks and outliers within topic time series.

In a content-feature-driven method, text, time, location, and other information are used as features in learning algorithms or other methods. In [74], the authors detect earthquakes location from related tweets. The authors build a classifier to detect tweets which are related to the target event. A probabilistic spatiotemporal model is then built to locate the center of the event. In [99], the authors propose a graphical model to capture the content, time, and location of social messages. Each message is represented as a probability distribution over a set of topics. The event is detected by conducting efficient similarity joins over social media streams.

In [71], the authors extract four features to represent an event: name entity, event phrase, calendar date, and event type. The named entity is extracted by NLP tools and the event phrase and event types are obtained by a learning method. The calendar date is obtained from a rule-based system. Li et al. [55] propose their twitter based event detection and analysis system to detect crime and disaster Events (CDE). Related tweets are crawled by keywords and then classified by Twitter-specific features and CDE features. The location of tweets is predicted by its authors' social network. Becker et al. propose a learning method in [10] to identify whether a tweet is related to an event. The features include temporal features, social features, topical features, and Twitter-centric features.

In [91], the authors propose a local event detection system. For the collected documents, the system identifies its theme and determine whether it is related to an event. For each detected event, the system extracts the key terms to describe the details of the event. In [30], the authors propose a real-time method to detect accident and disaster events. The messages are classified by content-based features, linguistic features, and event features. Based on the content, the system also estimates the street-level location of the event. An analysis of several state-of-the-art event detection techniques is provided in [92].

2.4. Document Summarization

He et al. [38] propose a framework named Document Summarization. The framework is based on Data Reconstruction which generates a summary of documents. To model the relationship among sentences, the authors introduce linear reconstruction and nonnegative linear reconstruction.

In [45], the authors propose a summarization method for a set of tweets. Each term in a tweet is assigned a weight and the weight of the tweet will be the average weight of terms. There are also some other methods based on weight, frequency, or speech analysis of terms and sentences like [61] and [87]. In [98], the authors propose a PageRank-based method to summarize the keywords on Twitter. A probabilistic scoring function is introduced to considers both relevance and interestingness of key phrases for the ranking.

In this work, an event is detected by monitoring the abnormal distribution of terms. Different from previous work, our method does not rely on outbreaks of the number of tweets or content features since many local events may attract only a little attention and without enough number of tweets related to them. Methods based on detecting outbreaks or extracting event-related features can hardly detect such events. Thus we analyze the abnormal distribution of terms in both time and space, which can help to detect more small local events.

2.5. Mining Uncertainty Co-locations

Co-location patterns [43, 97, 60] and efficient algorithms have been studied by various researchers. An initial summary of results on general spatial co-location mining was proposed in [80]. The authors proposed the notion of user-specified neighborhoods in place of transactions to specify group of spatial items. By doing so, they can adopt traditional association rule mining algorithms, i.e., Apriori [4] to find spatial co-location rules. An extended version of their approaches was presented in [43]. Fast mining algorithms are proposed in [97]. The algorithms combine the discovery of spatial neighborhoods with the mining process. The algorithms work on a given pattern, star, clique, or generic, and calculate the participation index using an extension of a spatial join algorithm. Others have considered

complex co-location patterns including negative co-locations [60] and zonal co-locations [20] with dynamic parameters, i.e., repeated specification of zone and interest measure values according to user preferences. The problem of mining co-location patterns with rare spatial features has been studied in [42]. The authors applied a new measure which considers the maximum participation ratio of the co-location patterns.

In [3], the authors study the problem of frequent pattern mining with uncertain data. They extend the Apriori-based, hyper-structure based, and pattern growth approaches and conclude that experimental behavior of different classes of algorithms is very different in the uncertain case as compared to the deterministic case. In [88], the authors studied on the probabilistic spatially co-locations and proposed a dynamic programming algorithm which is suitable for parallel computation. They proposed the uncertainty model by introducing the concept of existential probability of an instance. The probability of a possible world can be calculated as the product of the probabilities of presence or absence of all uncertainty instances. By multiplying the probability of possible worlds and the participation index under certain case together, they can get the final participation index. This model only considers the probability of existence while our model considers the possible locations of all instances.

The lexicographic tree (enumeration tree) has been used in mining maximal or long pattern association rules. The counting [95] and pruning [18, 101, 37] methods in the search tree to generate association rules are not applicable to our problem. In [95], the authors presented the maximal clique generation method. They modified the Bierstone's algorithm [34] and used the concept of α-*related* to weaken the clique generation process to avoid the edge density is too high.

2.6. Mining Topic Relationship

In the recent years, many researchers have shown interest in topic, trend, events and their correlation on social media. Different methods have been used to analyze tweet content including machine learning algorithms, language model, feature-based algorithms, information retrieval and much more with help from spatiotemporal data mining. Petrovic

et al. [65] employ a locality-sensitive hashing (LSH) to detect the first story from a stream of tweets. The proposed approach uses the hash table to organize new similar tweets in an existing story or labels. Although LSH has been used in nearest neighbor search applications, Protrovic's work makes event detection possible on large-scale tweets dataset. This approach does not differentiate the nature of events and content such as local event, disaster or news. Sakaki et al. [74, 75] classify tweets based on features such as keywords or word number and use a probabilistic spatiotemporal model to detect earthquakes from tweet content. The event location is estimated using Kalman filtering and particle filtering. The assumption of Sakaki's approach is that only one event takes place in one region at a time and users know the event to set up keyword queries in advance.

Budak et al. [17] proposed a location-topic pair index using both tweet location and user location to detect geo-trends or trends in different geographic locations in a sliding window. Kamath et al [47] did a thorough study of a tweet's hashtags as a representation of its topic. They did an analysis of the global footprint of hashtags and an exploration of the spatial constraints on hashtag adoption. They measure the hashtag's spatial propagation properties: focus, entropy, and spread. Lee et al. [54] tried to detect events such as traffic jams by using geographical coordinates of geotagged tweets and monitoring the geographical pattern of tweets. Sugitani et al. [82] use a different approach to detect events by using spatiotemporal clustering techniques. Schulz et al. [79] are the first in using a multi-indicator approach to get an accurate geographic location of tweets and Twitter users.

Analysis of the time, location and hashtags of tweets can help to extract more meaningful information from real-world events, trends, and roles in social networks. In [16], the authors detect trends which related to locations. Hashtags are also defined as topics of tweets here as well as [83] and three different thresholds are introduced to define the correlation between topics and locations. In [84], the authors propose a linear regression based approach to predict the spread of an idea in a given time frame. The hashtags are taken as the idea of tweets. Yang et al. [94] propose measurements to analyze the main factors of how users select hashtags and a machine learning model is used to help to predict the future adoption

of hashtags of users. Kywe et al. [51] study the time distribution of hashtags, and then the authors propose a hashtag recommendation method based on collaborative filtering. In [58], the authors propose a geovisual analytics approach to leveraging Twitter in support of crisis management. The hashtags which are related to locations are used to extract more location information of tweets. By studying the tweets related to 2011 London riots, Glasgow et al. [36] analyze emergent social networks directly relating to response to crisis. Their study shows that the hashtag lifespan may relate to social behaviors and social networks coupled to crisis response. Carter et al. [19] propose a method for translating hashtags, which builds on methods from information retrieval.

CHAPTER 3

LOCATION ESTIMATION

In this section, we first analyze the relationship between social closeness and geographic distance and introduce the community detection algorithm. Then we propose the home location estimation and real-time location estimation algorithm. Finally, we will briefly demonstrate an online location estimation system.

3.1. Community Detection on Social Network

Many real-world systems or web services can be represented as a network such as social networks, transportation networks, the World Wide Web, and biological networks. Detecting communities from those networks has received considerable attention and is the main focus of many research efforts in the past decade [26, 63, 32, 28]. Generally, the goal of community detection is to find the subgraphs with tight internal connection based on node connections, labels of nodes, and the weights derived from data or network structure. Nodes in the same community are closer to each other. Therefore, in the real world, a community represents a group of nodes sharing some similar common friends or features.

However, the formation of many real-world networks is greatly influenced by the geographic locations of the nodes which have not been fully investigated by the current literature. For example, in a social network, people have a high probability to build a connection with his/her colleague or schoolmate because they know each other or in most cases, they became friends because they are geographically close. Furthermore, some network applications, such as FourSquare, are mostly location-based social networks. The geographic location will play even more important in the social network structure on these platforms. There are preliminary studies on the relationship between social network structure and geographic distance [77] and [64]. However, those studies do not push location information into community detection.

We observe that the nodes in a tightly connected community tend to be more close to each other in space as well. Location can have a different impact on social networks and the

impact can be quantified and used in community detection. Introducing locations of nodes to community detection can improve the performance of detection on real-world networks. In this dissertation, we propose community detection methods that take the locations of the nodes into consideration with the main goal of improving the quality of the detection results in terms of average internal degree, accuracy, and geographic span of detected communities. Our research is based on the following two premises: (1) Location is an important factor and can greatly influence the connection establishment in many location-tagged networks; (2) For many applications, detecting communities with constrained geographic distribution is important. For example, finding local communities will be useful for arranging meetups of communities with similar interests. Knowing geographically constrained communities with potential interests in certain concert or talk shows can help arranging and scheduling the tours.

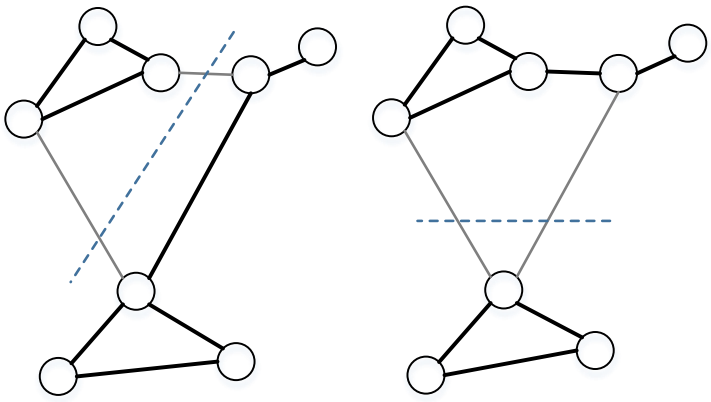


FIGURE 3.1. Two different divisions of a small location-tagged network. The left division is only based on the network structure and the right one takes the locations of the nodes into account.

We focus on finding communities with nodes distributing in a small range of area and at the same time, keeping the connection tightness of the nodes in the community. Figure 3.1 gives an example of how the geographic location of nodes can influence the detecting results. In this case, we set the number of communities to two. If we only consider the

network structure, the left one is a good result. There are only two edges coming across communities. After we introduce the location of the nodes, we will have two communities as in the right side. There are still only two connections across communities however the geographic spans of the two communities are much smaller than the left one. Unfortunately, in some networks, we may need to make a tradeoff on the structure tightness for keeping the nodes in the same community close. This dissertation presents a way to measure the locality and node similarity and gives a guidance on if a given network has locality in communities.

We denote the network as $G = (N, E, L)$, where N is the set of nodes, E is the edge set, and L is the location set of the nodes. To determine whether the locations of nodes will help in community detection, we will analyze the locality of the network first. Then we propose our locality-based method. We follow the hierarchical clustering framework combined with the location information. A good division of the network produces communities with a higher ratio of internal edges and smaller geographical scope.

3.1.1. Network Locality

As we discussed before, the formation of connections in many real-world networks are influenced by the location of nodes in the network. However, some networks are more location influenced than others. So before we provide the location-based community detection algorithm, we need to analyze the influence of the location on networks to see the degree of influence. This will be helpful in determining if location-based community detection is a suitable method. Here, we use network locality defined below to measure the relationship between location and connection in a network.

Definition[Network Locality] In a network G, we use two indexes to measure its locality: Total Variation Difference (TVD) and the Inflection Distance. Let $F(dis)$ be the cumulative distribution function (CDF) of distance between any two nodes in G and $F_c(dis)$ be the CDF of the distance between connected nodes in G, the total variation distance is defined as:

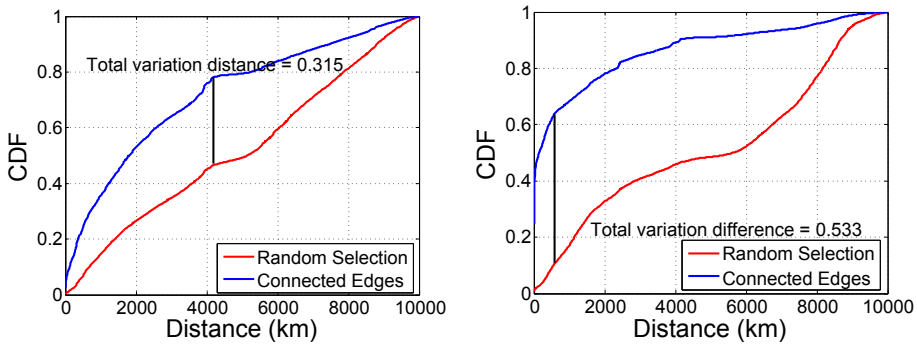
$$TVD(F, F_c) = max(F_c(dis) - F(dis)) \tag{3.1}$$

and the Inflection distance is defined as the distance where $F_c(dis) - F(dis)$ achieves the maximum value.

From the definition, we can see that a higher value of the total variation distance indicates the network is more geographically close because connected nodes in nearby locations have higher percentages. When the total variation difference is close to zero, the connection has little relationship with the locations of nodes. When the TVD is less than zero, the connection has a negative correlation with location. It is obvious that a small value of inflection distance represents a more geographic close network.

We analyze the network locality of two real datasets: Gowalla and Twitter. Gowalla is a location-based social network and users are able to check in at "spots" in their local vicinity. The Gowalla dataset [25] is a 196,591 users' friendship network. The check-in data were collected from February 2009 to October 2010 and each user has 32.8 check-in records on average. We use 99,563 of those users who have check-in records in our analysis. Since there is no user profile, we take the center of the $25km \times 25km$ area with the most number of check-ins as the user home location [78]. We also collected user profiles from Twitter, an online social networking and micro-blogging service which allows users to follow each other; post and read "tweets". The data are collected from April 14 to April 28, 2013. The social network comes from [90]. There are 660,000 distinct user IDs in total together with their social relations. We obtained locations of 148,860 users through their profiles. We define the friend relation in the same way as [44], i.e. users i and j has friend relation if they follow each other.

In Figure 3.2, we plot the cumulative distribution function of the distance between every user pairs and friend pairs. In the Twitter dataset, the total variation distance is 0.315 and the inflection distance is $4,180km$. That means that the percentage of connected edges with the distance less than $4,180km$ in all the connected edges is higher than that percentage of random user pairs by 0.315. Compared with Twitter, the Gowalla network is more close geographically since it has a higher TVD, 0.533, and a smaller inflection distance $580km$. This phenomenon illustrates that users in Gowalla tend to build friend relations with others



(a) Twitter: the total variation distance is 0.315 and the inflection distance is $4180km$

(b) Gowalla: the total variation distance is 0.533 and the inflection distance is $580km$

FIGURE 3.2. The cumulative distribution function of distance between every user pair/friend pair on Twitter and Gowalla.

who are geographically close to them compared with Twitter. In other words, the locations of nodes have a greater influence on the network structure in Gowalla. Our experiment later also show that our method can perform better on Gowalla than the Twitter network for this reason.

In practice, the total variation difference is more helpful to measure how the network structure is influenced by location. We suggest applying our method on the networks with the total variation difference larger than 0.25.

3.1.2. Connection Locality

To take location into account in community detection, first we define the concept of connection locality to qualify the graphic closeness between nodes.

Definition[Connection Locality] Let dis_{vw} be the geographic distance between nodes v and w. Let σ be the average distance between all user pairs. The connection locality can be defined as:

$$L_{vw} = exp(-dis_{vw}/\sigma) \tag{3.2}$$

So connection locality will achieve a high value when the two nodes are close. Since

our goal is to detect communities with both geographic closeness and network tightness, we measure the geographic and network closeness of the communities using the following equation:

$$C_G = \frac{1}{\sum_{vw} A_{vw} L_{vw}} \sum_{vw} A_{vw} L_{vw} \delta(c_v, c_w) \tag{3.3}$$

We can see that this method is equivalent to assign each edge in network G with the locality as weight. Inspired by the method in [26], we introduce the expected value of C_G to avoid the situation that the largest value of C_G will be achieved when all the nodes belong to the same community.

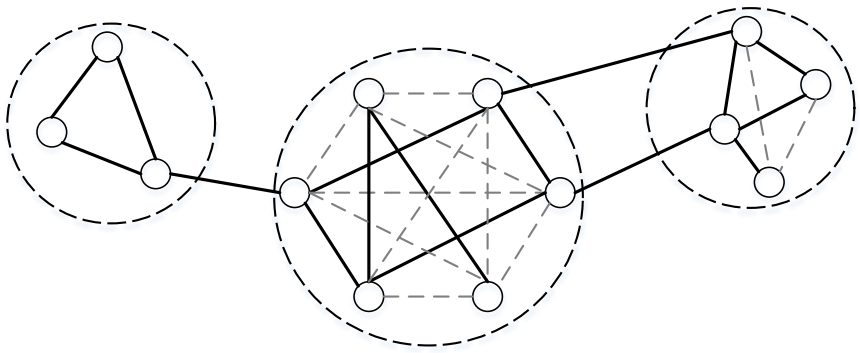


FIGURE 3.3. In this example, each dashed circle represents a community. The solid lines are connections between nodes.

The expected value of C_G is obtained from a random connection network. For a given network G, the location of the nodes and their degrees are fixed. In a random connection network, the probability of an edge existing between a node pair is $k_v k_w / 2m$. Since we already know the locations of the nodes, the community locality l_{vw} between a node pair v and w is also known. So the expect value for each edge is $l_{vw} k_v k_w / 2m$ and the expect value of C_G is the sum of the expect value of all the edges:

$$P_G = \frac{1}{\sum_{vw} A_{vw} L_{vw}} \sum_{vw} \frac{k_v k_w}{2m} L_{vw} \delta(c_v, c_w) \tag{3.4}$$

In Figure 3.3, there are three communities denoted by the dashed circles. The solid lines are the edges in the network. We use dashed lines to complement each community as a complete graph. Given the locations of all nodes, the community localities can be easily calculated. The probability of an edge existing between the bottom two nodes in the left community is $\frac{2\times 2}{13}$.

Let $\omega = \sum_{vw} A_{vw} L_{vw}$, we define the modularity Q as:

$$Q = \frac{1}{\omega} \sum_{vw} [A_{vw} L_{vw} - \frac{k_v k_w}{2m} L_{vw}] \delta(c_v, c_w) \tag{3.5}$$

The community locality between each node pair is fixed. If the network is not built based on locations, community locality will have no influence on Q and the value of the modularity Q will be close to zero. When the network has good divisions, which means the communities are close on both geographic distance and network structure, the modularity Q will achieve a higher value.

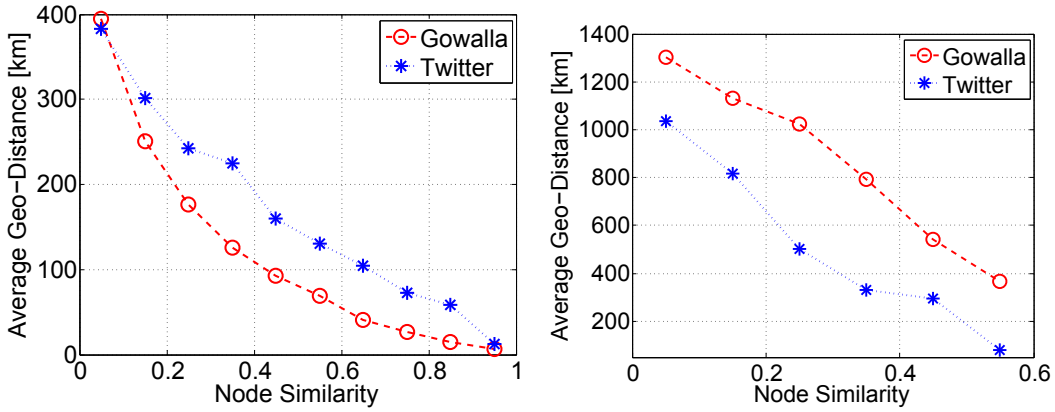
3.1.3. Node Similarity

In this section, we will discuss how to enhance the influence of network structure in the detection process. Here we define the node similarity between nodes pair by the common neighbors and their degrees:

Definition[Node Similarity] Let Γ_v be the set of neighbors of vertex v. The similarity of two nodes is calculated by their common neighbors and their degrees as:

$$S_{vw} = \frac{|\Gamma_v \cap \Gamma_w|}{\sqrt{|\Gamma_v||\Gamma_w|}} \tag{3.6}$$

First, we study the relationship between node similarity and the geographic distance between node pairs in real-world networks. From Figure 4(a) we can see that with the increasing of the value of node similarity, the average geographic distance has a significant decrease. Then we extend the investigation of node similarity to all 1- and 2-degree friend



(a) 1-degree node pairs

(b) 1- and 2-degree node pairs

FIGURE 3.4. The average geographic distance under different values of node similarity.

pairs. From Figure 4(b) we can see a similar tendency between node similarity and average geographic distance as that on 1-degree friends. But the average distances are much longer than the 1-degree friends, especially on the Gowalla data set.

To apply the node similarity in our modularity, we also need to calculate the expect value under random connection network. In the random case, when we calculate the expect value of S_{vw}, we need to know the probability of an edge existing between node v or w and any other node i. Assuming node i has k_i neighbors, the probability of node i is connected to v (w) is $k_v k_i/2m$ $(k_w k_i/2m)$. Since the probability of connection is independent, so the probability of node i connected to both v and w is $\frac{k_v k_i}{2m}\frac{k_w k_i}{2m}$. The expected value of S_{vw} is the sum of the probabilities of both v and w connected to any other node i:

$$\begin{aligned} S_{vw} &= \frac{|\Gamma_v \cap \Gamma_w|}{\sqrt{|\Gamma_v||\Gamma_w|}} \\ &= \sum_{i\neq v \& i\neq w} (k_v k_i/2m)(k_w k_i/2m)/\sqrt{k_v k_w} \\ &= \sqrt{k_v k_w} \sum_{i\neq v \& i\neq w} k_i^2/4m^2 \end{aligned} \tag{3.7}$$

In practice, we use $\tau = \sum_i k_i^2/4m^2$ instead of $\sum_{i\neq v \& i\neq w} k_i^2/4m^2$ because they have

similar value on larger networks.

We then revise ω as $\sum_{vw} A_{vw}S_{vw}L_{vw}$, and the new modularity Q^s is defined as:

$$Q^s = \frac{1}{2\omega}\sum_{vw}[A_{vw}S_{vw}L_{vw} - L(v,w)\frac{k_v k_w}{2m}\tau\sqrt{k_v k_w}]\delta(c_v, c_w) \tag{3.8}$$

In this work, we only consider the node similarity between connected nodes for the following reasons: (1) Relation of 2-degree neighbors (the node pairs which are connected but share at least one common neighbors) introduce many new connections. The number of 2-degree neighbors is much more than directly connected neighbors and will significantly increase the computation complexity. (2) The influence of 2-degree neighbors is much smaller than directly connected ones. Based on our investigation, the average distance between 2-degree neighbors is three times longer than directly connected neighbors even when they have the same node similarity.

3.1.4. The Algorithm

In this section, we will discuss how to implement the algorithm efficiently with optimization and indexing. The algorithm is based on the hierarchical clustering method with greedy strategy. At first, each node is a community. In each step, two communities whose combination increases the value of the modularity Q most are combined. In [26], the authors provide an efficient method to implement their model. They maintain and update a matrix ΔQ_{ij} which records the change of Q after combing the communities i and j. When Equation 3.5 is used as Q value, we can implement the model in a similar way. When Equation 3.8 is used as the modularity, we discuss the optimization here.

$$\begin{aligned} Q = &\frac{1}{2\omega}\sum_{vw} A_{vw}S_{vw}L_{vw}\delta(c_v, c_w) \\ &- \frac{1}{2\omega}\sum_{vw} L(v,w)\frac{k_v k_w}{2m}\tau\sqrt{k_v k_w}\delta(c_v, c_w) \end{aligned} \tag{3.9}$$

We can rewrite the modularity in Equation 3.8 into Equation 3.9. By analyzing the modularity, we can see that after we combine two communities i and j, the change of Q^s

includes two parts: 1) the connections between these two communities will increase the value of Q^s (the first part in Equation 3.9), the value equals to:

$$\Delta Q_1 = \frac{1}{2\omega} \sum_{vw} A_{vw} S_{vw} L_{vw} \delta(c_v, i) \delta(c_w, j) \tag{3.10}$$

and 2) the value generated by node pairs from communities i and j and this value equals to:

$$\Delta Q_2 = -\frac{1}{2\omega} \sum_{vw} L(v, w) \frac{k_v k_w}{2m} \tau \sqrt{k_v k_w} \delta(c_v, i) \delta(c_w, j) \tag{3.11}$$

So the ΔQ_{ij} equals to $\Delta Q_1 + \Delta Q_2$. Since we will combine two communities with the largest ΔQ_{ij}, we only need to keep values in Equations 3.10 and 3.11. Now we only need to solve two problems, how to initialize the ΔQ_{ij} and how to update it after we combine two communities.

The combination of two disconnected communities will not increase the value of Q, so we only keep the ΔQ_{ij} if there is at least one edge between them. At first, every node is a community and the ΔQ between each connected node pair is:

$$L_{ij}[\frac{S_{ij}}{2\omega} - \frac{\tau(k_i k_j)^{1.5}}{4\omega m}] \tag{3.12}$$

After we combine communities i and j, we need to update all the communities k which are connected to i or j. We use (ij) to denote the community generated by combining i and j and use $\Delta Q_{k,(ij)}$ to denote the new ΔQ value between k and (ij). If the community k is connected to both i and j, we can get the new $\Delta Q_{k,(ij)}$ by $\Delta Q_{ik} + \Delta Q_{jk}$. If k is only connected to one of them, e.g. i, we do not have Q_{jk} since they are disconnected. So we need to calculate it. We have already known that ΔQ is the sum of Equation 3.10 and 3.11. The ΔQ_1 will be zero since there is no edge between j and k. So we can have the ΔQ_{jk} as:

$$\Delta Q_{jk} = -\frac{1}{2\omega}\sum_{vw}\tau L(v,w)\frac{(k_v k_w)^{1.5}}{2m}\delta(c_v,j)\delta(c_w,k) \tag{3.13}$$

And then we can update the $\Delta Q_{k,(ij)}$ by:

$$\Delta Q_{k,(ij)} = \Delta Q_{ik} + \Delta Q_{jk} \tag{3.14}$$

Algorithm 1: Detecting communities from location-tagged network

```
Input: Network G = (N, E, l)
Output: Communities in G
Assign each node a community label from 1 to n
Initialize the ΔQ_ij as Eq.3.12
Find the maximum ΔQ_ij, maxΔQ
while maxΔQ ¿ 0 do
    Update ΔQ_k,(ij) of all the communities k connect to i or j by Eq.3.13 and Eq.3.14
    Update the community label in community i as j
end while
Return node list with the community label
```

The Algorithm 1 describes the framework of all the process. We will stop the hierarchical clustering process when the modularity Q achieve its maximum value, which means that the largest ΔQ_{ij} is less than zero.

We store each row of the ΔQ_{ij} and the node list in different communities in a balanced binary tree. When we update a $\Delta Q_{k,(ij)}$, the worst case is that we need to calculate the ΔQ_{jk} by Equation 3.13 and the complexity is $O(|j||k|logn)$, where $|j|$ represents the number of nodes in community j. For each combination of two communities, the worst case is that all the nodes connected to all the communities. Assume that the depth of the hierarchical clustering is d and the number of nodes in a community is c_n, the complexity is $O(mdc_n^2 logn)$.

3.1.5. Experimental Results

In this section, we test our method on synthetic networks and two real world social network datasets described before: Twitter and Gowalla. We use three different measurements to evaluate the results:

Definition[Geographic Span] The geographic span of a community c is defined as the average distance of the nodes in c to the centroid $(\bar{x}, \bar{y})$ of all the nodes in the community:

$$S(c) = \frac{1}{|c|} \sum_{v} \sqrt{(x_v - \bar{x})^2 + (y_v - \bar{y})^2} \delta(c_v, c) \tag{3.15}$$

Definition[Average Internal Degree] The internal degree of a node v is the number of its neighbors in the same community. The average internal degree of a community c is the average value of the internal degrees of all the nodes in c and it can be represented as:

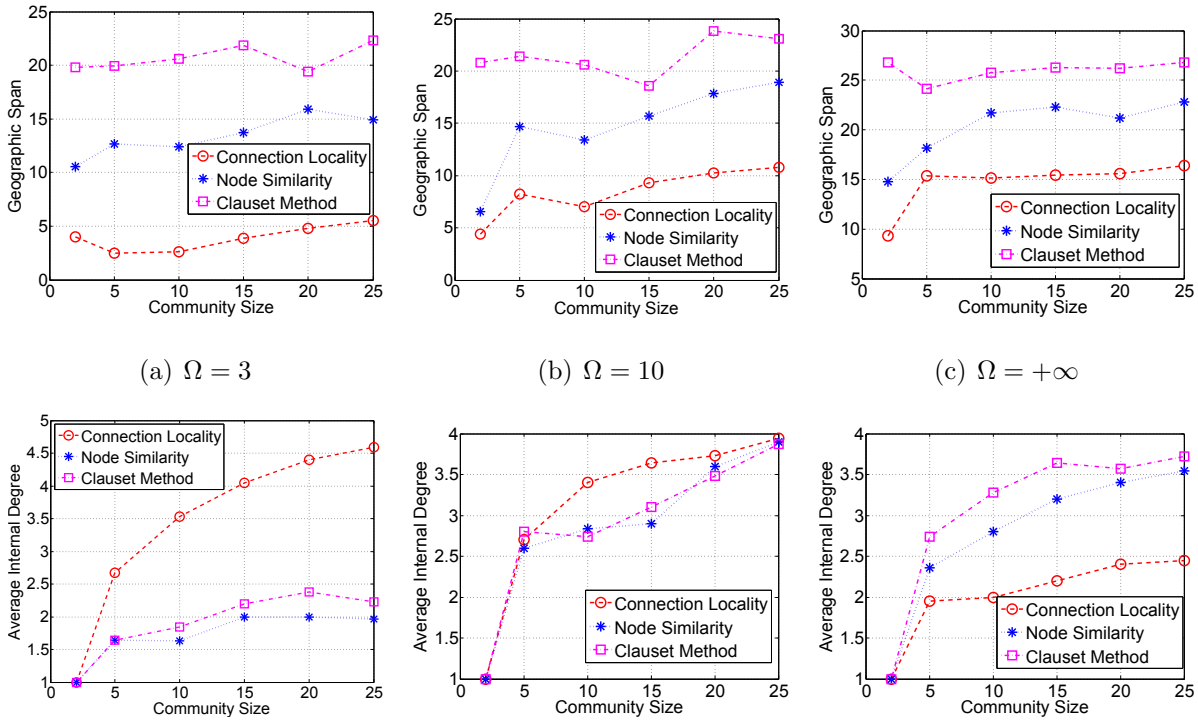
$$A(c) = \frac{1}{|c|} \sum_{vw} \delta(c_v, c)\delta(c_w, c) \tag{3.16}$$

The last measurement is the detection accuracy. Since we do not have a class label of the real world datasets, we only apply this on the synthetic networks. We implemented four community detection methods in our experiments: 1) Randomly select nodes as community (Random). 2) The method proposed in [26] (Clauset's Method). 3) The method discussed in section 4.1 using Equation 3.5 as the modularity Q (Connection Locality). 4) The method discussed in section 4.1 with Equation 3.8 as the modularity (Node Similarity).

Ω	Clauset's Method	Connection Locality	Node Similarity
1	16.24	16.63	18.38
3	16.48	22.82	24.63
5	17.72	22.40	28.77
10	22.16	25.14	26.60
30	32.84	19.76	24.42
$+\infty$	36.04	19.20	19.76

TABLE 3.1. The accuracy of different community detection methods

First we test the methods on the generated networks because a synthetic datasets allow for better parameter control. We analyze the results using the three measurements



(a) $\Omega = 3$ (b) $\Omega = 10$ (c) $\Omega = +\infty$

(d) $\Omega = 3$ (e) $\Omega = 10$ (f) $\Omega = +\infty$

FIGURE 3.5. The geographic span and average internal degree of the synthetic network under different values of Ω.

discussed above. When generating the dataset, we control the influence of geographic distance on building connection between two nodes in order to see the how the geographic feature affect the detection methods.

We generate the networks on a 50×50 grid. There are 2,500 nodes in total in the network. For each node, we randomly assign a community label to it. There are 10 different community labels in the network. We generate the probability of an edge existing between node v and w as:

$$p_e = \alpha p_c e^{-dis_{vw}/\Omega} \tag{3.17}$$

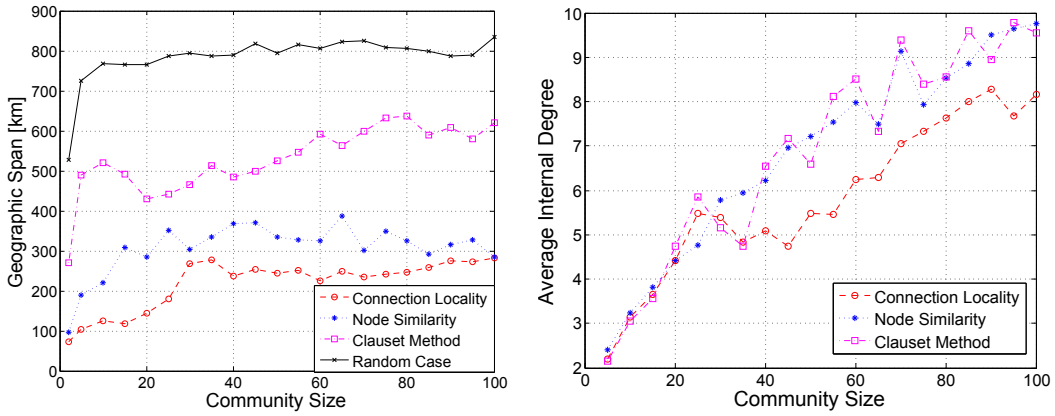
The value of p_c depends on whether the node v and w have the same community label. If their community labels are the same, p_s is set to 0.5 and if not, p_c is set to 0.1. So the edges

have a higher probability of occurrence between the nodes with the same community label. The component $e^{-dis_{vw}/\Omega}$ is used to control the influence of the locations of nodes. When we set a large enough value to Ω, the value of $e^{-dis_{vw}/\Omega}$ is close to 1 and the probability is almost not influenced by dis_{vw}. So the network structure is not influenced by the locations of nodes. On the contrary, if the value of Ω is small, the value of $e^{-dis_{vw}/\Omega}$ will be greatly influenced by the distance between v and w. In that case, only the nearby neighbors with the same community label will have a high probability of connecting. The parameter of α is used to control the average degrees of nodes in the network. In the following experiment, we make the number of average degree around 15 by adjusting the value of α.

Table 3.1 shows the accuracy of different algorithms on different generated networks. Since the largest distance in the network is only 70, when we set the Ω larger than 10, the probability of connecting is not very sensitive to the distance. We can see that when the Ω is less than 10, which means the building of connections is greatly influenced by the location of nodes, our two methods can achieve a similar or higher accuracy than the Clauset's method. With the increasing of the value of Ω, when the location has little or no influence on the network structure, the accuracy of Clauset's method performances better than our methods. So we recommend to evaluate the influence of geographic information first as described in section 3.1.1 before applying our methods on a network.

Figure 3.5 shows how the three methods perform on different synthetic networks. From Figure 5(a) to 5(c) we can see for all levels of influence (Ω) that the geographic location on network structure, the connection locality have the smallest geographic span. The geographic span of the node similarity method is smaller than the Clauset's method. Figure 5(d) to 5(f) show the average internal degree of the three methods. When Ω is 3, where the location of nodes will have the greatest influence on the network structure, the connection locality method has a higher value of internal degree. This illustrates that this method is suitable to deal with the highly geographically influenced networks. When the value of Ω increases to 10, the average internal degree of these three methods is similar. But when we set the Ω as infinity, the connection locality and node similarity method perform

worse than Clauset's method.



(a) The geographic span of different size of communities detected by different methods

(b) The average internal degree of different size of communities detected by different methods

FIGURE 3.6. Analyzing the community detection results of different methods on the Twitter Network.

In the real world, the factors which can influence the network structure can be very complex. We now test the algorithms on the networks generated by some real-world applications. The first example is the Twitter network. We have introduced the details of this network in Section 3.1.1. Since we do not have a community label for the real world dataset, we only apply the geographic span and the average internal degree of the communities to evaluate the detection results.

In Figure 6(a), we demonstrate the geographic span of different sizes (number of nodes in the community) of communities. From this figure, we can see that under the random case, the geographic span is much larger and increases quickly to 800 kilometers. The communities detected by Clauset's method has a smaller geographic span. It begins with 280 kilometers when the community size is 2 but increases quickly when the community size becomes larger. Finally, the geographic span fluctuates between 500 to 600 kilometers. The two methods proposed in this dissertation have the best performance on controlling the geographic span on communities. Although the geographic span increases quickly when the community size

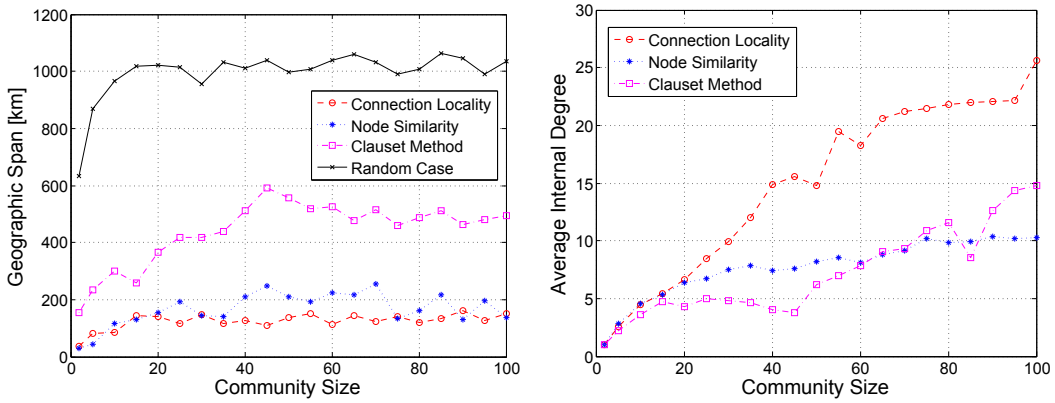
becomes larger, these two methods can keep the span much smaller than Clauset's method and the random case, especially for the method with the Equation 3.5 as the modularity. The geographic spans in different sizes of communities are only half of Clauset's method.

The Figure 6(b) shows the average internal degrees of different sizes of communities. This measurement evaluates the detection result by the network structure only. From the definition, we know that if a community has a higher internal degree, that means the connections inside the community is tighter. From the figure, we can see that with the increase of the community size, the average internal degree also becomes larger, which means nodes have more neighbors in the same community with them. The Clauset's method and one of our method, which uses Q^s as the modularity, have a similar performance. The connection locality method has a smaller average internal degree when the community size is larger than 40. The results are encouraging and showing that our methods can detect communities with a similar internal degree and a smaller community in the geographic span.

The second real-world network is Gowalla. From the analysis in section 3.1.1, we know that compared to the Twitter network, the geographic information in Gowalla has a greater influence on the network structure. So the Gowalla network is more suitable to use our community detection methods. From Figure 7(a), we can see that our methods have a strong effect on limiting the geographic span of communities. Both the two methods can keep the span around or less than 200 kilometers. Especially for the connection locality method, even when the community size is very large, it can still keep the geographic span in a small range.

Another important observation is that in the highly geographically influenced networks, our method can also improve the network tightness in the communities. Figure 7(b) shows the results of the average internal degree. The performances of these algorithms are similar to the case on the Twitter network. The difference is that in the Twitter network, the connection locality method performs worse than the other two methods. But on the Gowalla network, it performs much better. This phenomenon illustrates that on the high geographically influenced networks, our method can improve the quality of the detection

results on both geographic span and the tightness inside communities.



(a) The geographic span of different size of communities detected by different methods

(b) The average internal degree of different size of communities detected by different methods

FIGURE 3.7. Analyzing the community detection results of different methods on the Gowalla Network.

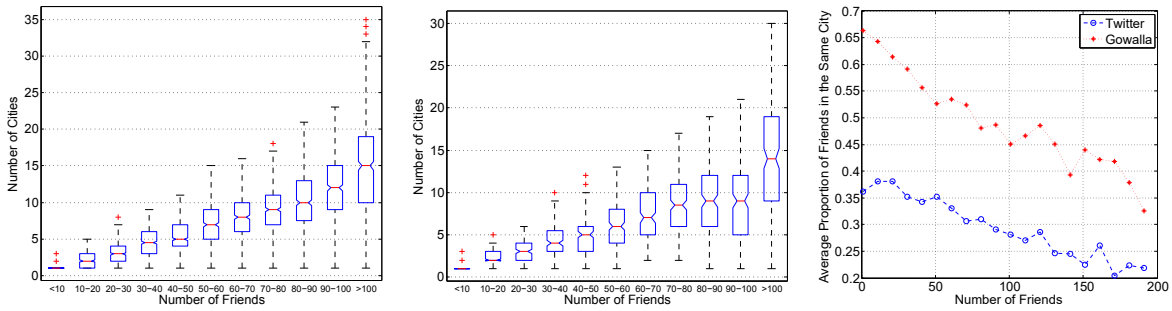
3.2. Location Estimation

In this section, we introduce our home location algorithm. First, we introduce the analysis about the relationship between social closeness and geographic distance. Then we propose our estimation model with the comfidence-based iteration method.

3.2.1. Social Closeness and Geographic Distance

We collected our data from two different social media platforms, Gowalla and Twitter. Gowalla is a location-based social network, and users are able to check in at "spots" in their local vicinity. The Gowalla dataset [25] was collected from February 2009 to October 2010, which contains 196,591 users' friendship network and 6,442,890 check-in records. We use 99,563 of those users who have at least one check-in records in our experiment. Since there is no user profile, we use the same method as that in [25] and take the center of the $25km \times 25km$ area with the most number of check-ins as the home location. We then collected user profiles from Twitter, an online social networking and microblogging service

which allows users to follow each other; as well as post and read "tweets". The user IDs and social network come from [90]. There are 660,000 distinct user IDs in total together with their social relations. We collected the profiles of these users using Twitter API [1]. We obtained locations of 148,860 users by converting the address in their profiles into geographic coordinates by the Google Maps Geocoding API [2]. The data was collected from April 14 to April 28, 2013. We define the friend relation in the same way as [44], i.e. users u_i and u_j have friend relation if they follow each other.



(a) *Twitter* (b) *Gowalla* (c) *Proportion of Friends Located in the same city*

FIGURE 3.8. The distribution of number of cities a user has friends located at and the average proportion of friends located in the same city.

Figure 8(a) and 8(b) show the city distribution of friends on Twitter and Gowalla social networks. For most users with less than 70 located friends, their friends will be located in no more than 10 cities. When a user has more than 100 located friends, his/her friends may be located in 10 to 20 cities. The rapidity and ease of modem transportation and communication present a great opportunity of meeting new friends in different places, leading to a wide distribution of friends, which challenges social network based location estimation. Figure 8(c) shows the average proportion of friends in the same city of a user with respect to the number of friends of a user. The result shows that with the increase in the number of friends, the probability of friends located in the same city decreases quickly.

[1]https://dev.twitter.com/rest/public

[2]https://developers.google.com/maps/documentation/geocoding/intro

Friend locality is the average distance to the friends of the user u_i:

$$F_{u_i} = \frac{1}{|\Gamma_i'|} \sum_{u_j \in \Gamma_i'} d_{ij} \tag{3.18}$$

Here we use Γ_i' to denote the located friends of user u_i. In Figure 3.9, it not surprising that friend locality increases with respect to the number of friends of a user. On one hand, not having enough friends will make location estimation difficult and on the other hand, having too many friends is not helping as well.

3.2.2. Friend Distribution of Connected Pairs

Friend-based methods were widely used in research literature [9, 29, 73]. By analyzing the distribution of friends' location or mobility, probability models can be built to predict locations of users. In these works, friend relation was taken as a binary feature: being a friend or not. However, in the real world, depending on the social relation and other user behaviors, friends of a user can be very different. In this section, we propose the friend co-location index ($FCoI$) to measure the "closeness" of friends on the social network.

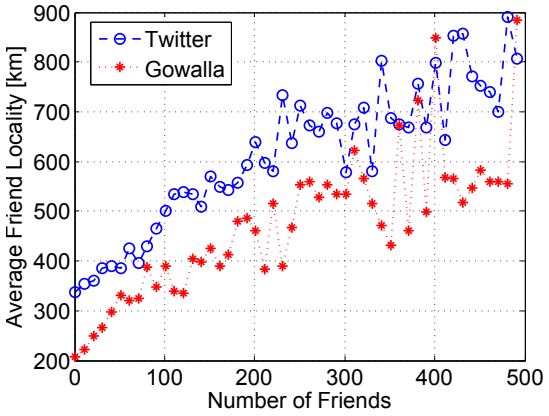


FIGURE 3.9. The relationship between average distance and number of friends.

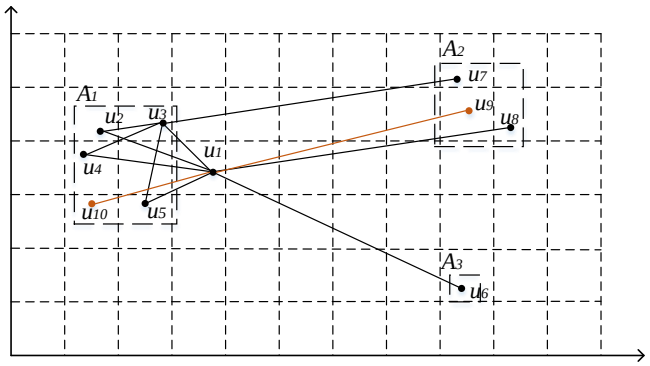


FIGURE 3.10. Estimating location of user u_1. Here $u_2, u_3, \ldots, u_8$ are located friends, and u_1, u_9, and u_{10}'s locations are unknown

We represent the social network as an undirected graph $G = (U, E)$, where U represents the user set, and edges in E exist between two users if they have a friend relation. There are two kinds of nodes in U. U' is the set of located user and U^- represents the others, so $U = U' \cup U^-$. Before performing the location estimation, we first cluster a user u_i's friends Γ_i by their location. Friends in the same city will be put in the same set and we represent those sets as $\mathscr{A} = \{A_1, A_2...\}$. The city is selected because of its natural definition of activity concentration by human geography. An example is shown in Figure 3.10, users u_1 has 7 located friends. These friends distribute in three different cities and form three sets of friends A_1, A_2 and A_3.

To measure the closeness of two users on the social network, we propose the Friend Co-location Index ($FCoI$) that takes both the social connection and the location into account. The key idea is to measure the correlation of the friends' geographic distribution of two users. For a pair of friends u_i and u_j, we firstly generate two vectors for each of them to describe the friend distributions as:

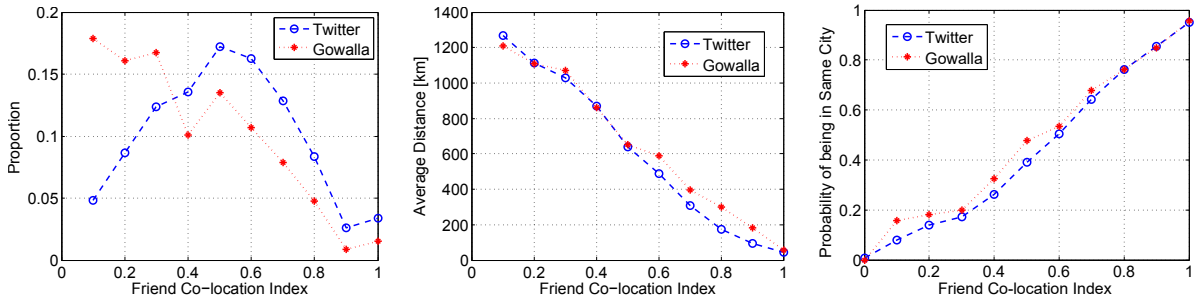
$$v_k^i = |A_k^i|/|\Gamma_i'| \tag{3.19}$$

The $|\Gamma_i'|$ is the total number of located friends of user u_i and $|A_k^i|$ is the number of friends of user u_i located in city A_k. For example, in Figure 3.10, u_3 has 4 located friends where three of them are in city A_1, one is in A_2, and none in city A_3. So the vector v^3 is [0.75 0.25 0]. Similarly, v^1 = [0.57 0.29 0.14]. After getting the distribution vectors, we can define the friend co-location index between user u_i and u_j as:

$$FCoI(u_i, u_j) = \frac{\sum_{k=1}^{m} min\{v_k^i, v_k^j\}}{\sum_{k=1}^{m} max\{v_k^i, v_k^j\}} \tag{3.20}$$

Here m is the total number of the cities. When the friends of two users have the same distribution, the distribution vectors of them will be also the same and the result of $FCoI$ is 1. On the other hand, if the distributions are completely different, e.g. all the friends of u_i are located in city A_1 and friends of u_j are located in A_2, the result will be

0. For example, in Figure 3.10, the friend co-location index between u_1 and u_3 will be: $(0.57 + 0.25 + 0)/(0.75 + 0.29 + 0.14) = 0.69$. The reason we choose the index as the sum of min over the sum over max is to give more priority to a few cities where both users have a large number of friends over many cities where both users have a small number of friends. From the observation of the dataset, we can see that many users tend to have a small number of friends in many cities but only friends located in the same city tend to have a large number of friends in a few cities.



(a) *Distribution of friend pairs under different value of friend co-location index*

(b) *The average distance between friends*

(c) *The probability of being in the same city*

FIGURE 3.11. Statistics on friend co-location index

Then we study whether the friend co-location index can reflect the geographic distance between two users effectively. We carry out these investigations on the Twitter and Gowalla users located in the North America to avoid the effect of oceans. Firstly, we calculate the proportion of friend pairs under different values of $FCoI$ in figure 11(a). The distributions of the two datasets are significantly different. More friend pairs in Gowalla have friend co-location index under 0.4, but the peak of the distribution of Twitter dataset is between 0.4 and 0.6. Another observation is that only a few friend pairs have the $FCoI$ more than 0.8 on both Twitter and Gowalla, which tells us that people have friends far away nowadays, and the social connections of each individual are quite different.

In Figure 11(b), we show the relationship between the average geographic distance and different value of friend co-location index between friend pairs. Obviously, when friends

have higher $FCoI$, they tend to be geographic close. The results of Twitter and Gowalla data sets are very similar to each other. Figure 11(c) shows the relationship between $FCoI$ and the probability of the friend pairs located in the same city. When friend pairs have friend co-location index higher than 0.9, the probability of they located in the same city can be higher than 85%. The probability increases with the increase of the value of $FCoI$. From the investigations above, we can see that $FCoI$ can be a good index to reflect the geographic relationship between friends.

We first propose the Friend Co-location Model based on the investigations above. We firstly calculate the probability $P(FCoI(u_i, u_j))$. It denotes the probability of users u_i and u_j located at the same city when the value of friend co-location index between them is $FCoI(u_i, u_j)$. We can get the value of $P(FCoI(u_i, u_j))$ based on the statistics shown in Figure 11(c). So for a user u_i in U^-, we can calculate the probability of u_i located at a city A_k as:

$$P(u_i, A_k^i) = 1 - \prod_{u_j \in A_k^i, (u_i, u_j) \in E} (1 - P(FCoI(u_i, u_j))) \tag{3.21}$$

The city A_k with the maximum value of the probability will be chosen as the estimated location of user u_i.

3.2.3. Structure of an Individual's Friends in a City

Friend co-location index can help to identify more important friends in location estimation. However, when user pairs which contain at least one user who has more than 50 friends, the value of friend co-location index will be in a small range (0 *to* 0.2) and the influence of friend co-location index on location distribution is not distinct. In this case, the effectiveness of friend co-location index in finding important friends is weakened. We introduce the concept of the local social coefficient to deal with this problem.

The new measurement, local social coefficient, is similar to modularity used in community detection [26], but incorporates location information. The local social coefficient of a user u_i in city A_k is defined as:

$$LSoC(A_k^i) = \sum_{u_p,u_q \in A_k \wedge (u_i,u_p) \in E \wedge (u_i,u_q) \in E} [a_{pq} - \frac{|\Gamma_p||\Gamma_q|}{2|E|}] \tag{3.22}$$

Here $|E|$ is the total number of edges in the social network G and $a_{pq} = 1$ if $(u_p, u_q) \in E$ and otherwise $a_{pq} = 0$. Intuitively, local social coefficient measures how tight the friends of u_i in the city A_k are compared with expected number of friend connections from a random friendship formation as measured by $\frac{|\Gamma_p||\Gamma_q|}{2|E|}$. One can choose the city with the highest value of the local social coefficient as the estimated location. The method may work well when a user has many friends and his/her friends can form structures in one or more cities. However, when a user has a small number of friends and his/her friends do not form structures in any cities, this method does not work well. Fortunately, this method performs better when the friend co-location based method fails (when a user has too many friends). In this work, we propose a method to leverage the advantages of these two approaches and achieve overall much better performance than the state-of-the-art.

Then we propose a Local Social Coefficient Model. There are nearly 28 percent of users on Twitter and 8 percent of users in Gowalla who have only less than 10 percent of friends located within 100km with them, and many of them have more friends in another city. The friend co-location based method can be very helpful when the estimated users don't have enough friends, but when the number of friends becomes larger, the effect of friend co-location index will be weakened. The local social coefficient becomes more helpful when users have more than 30 friends. In our investigation on Twitter and Gowalla data sets, the probabilities of two friends u_1 and u_2 located within 100km are 0.2 (Twitter) and 0.27 (Gowalla). However, if there exists another user u_3 who can form a size-3 clique with u_1 and u_2 in the social network, the probability of u_1 u_2 located together can increase to 0.32 (Twitter) and 0.37 (Gowalla). Moreover, if u_1 and u_2 are located within 100km, the probability of u_3 located close to them is 0.69 (Twitter) and 0.58 (Gowalla). In this method, we will calculate the local social coefficient of the groups of friends in each city as formula 3.22 and use the city which has the largest value of $LSoC$ as the estimation result.

3.2.4. Social Closeness and Social Structure-Based Model (SoSS)

We have introduced a friend co-location based model, in which we want to find more important friends in location estimation, and the local coefficient based model, which has better results when a user have many friends. Scale-free network theory[89] states that in a social network, a large portion of users has a small number of friends and the locations of these users are difficult to estimate by the location of their friends. So we propose a confidence-based iteration method to overcome the problem of the sparsity of location information. In the Social Closeness and Social Structure-Based Model ($SoSS$), we combine these two models (friend co-location based and local social coefficient based models) together to overcome the disadvantages of them. After getting the two results from the two models above, we combine these two models by following the logistic regression based method in [40] as in formula 3.23. The parameters α, β_1, β_2, and β_3 can be trained by methods based on maximum likelihood. Since the number of friends has a great impact on the location estimation models, $|\Gamma_i|$ is chosen as a feature.

$$g(u_i, A_k) = \frac{exp(\alpha + \beta_1 LSoC(A_k) + \beta_2 P(u_i, A_k) + \beta_3 |\Gamma_i|)}{1 + exp(\alpha + \beta_1 LSoC(A_k) + \beta_2 P(u_i, A_k) + \beta_3 |\Gamma_i|)} \tag{3.23}$$

3.2.5. Iteration with Confidence-Based Improvement

In location estimation, one of the main problems is the sparseness of location information. By the investigation of [9], only 6% users provide their home address in their Facebook profiles. Our experiment shows that there are two main challenges in the location estimation: 1) some users only have a few number of friends. For these users, the accuracy can be very low since we cannot get enough location information from their friends, and 2) most of the users do not provide their location information in profiles. So we try to apply an iteration method to use the estimated locations. In the iteration steps, the estimated location will be taken as a friend's real location. However, there is a problem that the incorrect estimation results may lead to the decreasing of accuracy. So we propose a confidence-based iteration method.

Confidence-based method means that when we use the estimated location, we will judge the result with a confidence value and only use the results with high reliability of being correct in the iteration. Our investigation shows that the most helpful information is the aggregation of location distribution of friends. So we use an entropy-like method to measure the friends aggregate:

$$C_{u_i} = -\sum \frac{|A_k^i|}{|\Gamma_i|} \log \frac{|A_k^i|}{|\Gamma_i|} \tag{3.24}$$

Here, $|A_k^i|$ is the number of friends of user u_i in city A_k. So each time after we finish estimating of a user, we will calculate this entropy of him/her. In our model, we only use 66% estimated locations on Twitter dataset and 74% on Gowalla dataset with lower C_{u_i} in the iteration process. We get this ratio by analyzing the estimating accuracy on the training data and with this ratio, the iteration method can achieve the best accuracy. Over filtering will decrease the estimation accuracy since it may delete many useful correct estimating locations.

Assume there are totally n users in the social network and each user has m friends on average. In the friend co-location based model, for each user u_i, we need to calculate the value of $FCoI(u_i, u_j)$ and $1 - P(FCoI(u_i, u_j))$ for every friend u_j. So the cost of this step is $O(m^2)$ and the complexity of the model is $O(nm^2)$. For the local social coefficient based model, we need to calculate the local social coefficient of each user at each city where he/she has friends located at. The worst case is that all the friends located at the same city and the complexity is $O(m^2)$. In the second step, we only need to choose the city with the highest value of the local social coefficient as the estimated result, so the complexity of this model is $O(nm^2)$. To combine the two models together, we need to repeat the calculation above at first and the complexity is $O(nm^2)$. The cost of the combination step depends on the number of cities and the cost will be less than or equal to $O(m)$. So the complexity of the $SoSS$ model is $O(nm^2)$.

3.2.6. The Dataset

We collected tweets published from May 21 to July 6, 2015 using Twitter Streaming API[3] with the spatial bounding box [24°3′N, 127°7′W, 48°8′N, 65°5′W], which covers U.S. mainland. The dataset consists of over 60 million geotagged tweets from 2,200,402 Twitter users. This dataset will be used in the experiment for real-time location estimation, event detection, and mining topic associations.

3.2.7. Tweets Geotagging: Estimating Real-Time User Locations

Geotagged tweets allow one to extract geo-information-trend [16], search local events [93], and identify natural disasters [74]. However, geotags are not consistently available due to privacy concerns and other reasons. In fact, only less than 1% of tweets contain a geotag [1] and nearly 90% of Twitter users do not have geo-location enabled[4]. Geotagging is the process of assigning a location to a tweet indicating where it is published. Accurate geotagging, the focus of this work, will provide added values to social media contents for applications such as emergency response, event search, trend detection, and smart transportation. The availability of user geotagged tweets allows one to profile the geographic distributions of terms in tweets and these distributions have been used in several existing works for geotagging. The bayesian-based method generates a language model for a location and calculates the probability of a tweet belonging to different locations based on the Bayesian model [48]. Given the local language model θ_L, the probability of a tweet T belonging to a location L can be calculated as: $P(L|T) = \frac{P(T|\theta_L)P(L)}{P(T)}$ and $P(T|\theta_L) = \prod_i P(t_i|\theta_L)$. The location with the maximal probability is assigned to a tweet as its geotag. In [24], the authors also propose a similar method to estimate the locations of tweets based on the distributions of words. Kullback-Leibler (KL) divergence based method measures the difference between a tweet and the local language model [48, 72]. The location with minimal KL value is assigned to the tweet. Gaussian model or Gaussian Mixture Model (GMM) fits geotags of n-grams in

[3]https://dev.twitter.com/streaming/overview

[4]http://www.beevolve.com/twitter-statistics/

tweets into Gaussian distributions. The center of the Gaussian distribution [31] or a weighted sum of GMMs [68] is assigned as the geotag of tweets.

Knowing a user's home helps to understand user movement patterns. Users are bound by travel distance from his home and the types of outbound cities from home affect visiting possibility. However, only 16% users register their city level locations [56] and collecting the home location from user profile is strictly limited by the rules of Twitter API. Previous work has provided different methods for estimating the home locations of users [69, 49]. In this work, we assign home location following the method used in [78, 25]. A set of tweets of each Twitter user is selected to estimate the home location (this set will not be used for building and testing geotagging models). These tweets are assigned into different cities according to their geotags. We use city boundaries from the United States Census Bureau[5] to decide which city a geotagged tweet belongs to. Then we assign the city with the most number of tweets as a user's home. In [78], the authors show that the home location obtained in this way has an 85 percent accuracy.

We select all cities with more than 5,000 users. There are 40 cities C in our dataset. We only keep the tweets: (1) twittered by users in C; (2) published in C. Figure 3.13 shows the number of users and the number of geotagged tweets in these 40 cities. The top three cities with the most number of users and tweets are Los Angeles (46,508 users;1,183,634 tweets), Chicago(30,689 users; 634,665 tweets), and New York(30,689 users; 419,130 tweets).

The problem of geotagging tweets is equivalent to estimating the locations of users where they publish the tweets. Locations of users are not only revealed by tweet content but also by a user's previous and next locations because traveling preferences have a great influence on the next locations of users. In this section, we propose a Hidden-Markov-based model to integrate user movement and tweet content in geotagging. The model also considers the home location of a user to better represent the transition probability for users of different home cities.

In our model, the states of the Hidden Markov Model are the city level locations

[5]http://www.census.gov/en.html

of users and the state observations are tweets. The state (city) is not directly visible but the observation (tweet) is visible. Therefore, the sequence of tweets generated gives some information about the sequence of cities. Each tweet corresponds to a state which represents the city where the user publishes the tweet. The transition probabilities of the HMM are observed from a large number of user movement records.

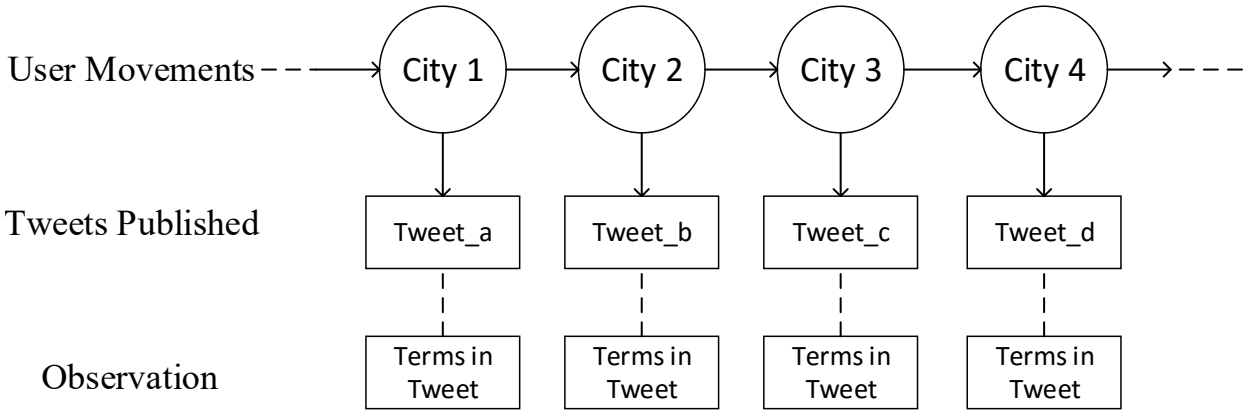
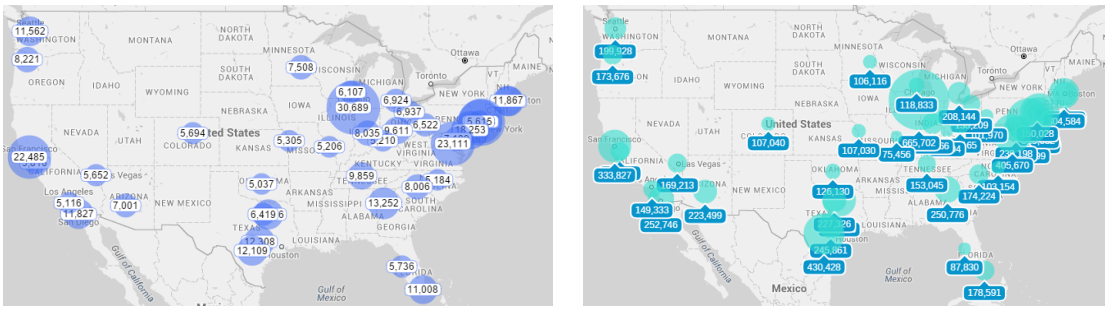


FIGURE 3.12. Estimating geotags of tweets by Hidden Markov Model

More specifically, as it is shown in Figure 3.12, for each user with N published tweets, he/she will have N corresponding states. Each state represents the location of the user where he/she publishes the tweet. The discrete states of the HMM are the set of all the cities. The goal is to predict a series of states (cities) from a series of observations (tweets) of the user.



(a) The distribution of number of users in 40 selected U.S. cities.

(b) The distribution of number of tweets in 40 selected U.S. cities.

FIGURE 3.13. The distributions of users and tweets in the dataset.

Transition Probabilities: Transition probabilities are observed from a large number of user traveling records. For each user u with a list of published tweets $T^u = \langle T_1^u, T_2^u, \dots \rangle$, we obtain the list of cities where he has visited through the geotags of his tweets: $\mathcal{C}^u = \langle \mathcal{C}_1^u, \mathcal{C}_2^u, \mathcal{C}_3^u \dots \rangle$. Given a city $\mathcal{C}_i$, the transition probability from $\mathcal{C}_i$ to city $\mathcal{C}_j$, $P(\mathcal{C}_j|\mathcal{C}_i)$, is defined as:

$$P(\mathcal{C}_j|\mathcal{C}_i) = \frac{\sum_u \mathcal{C}_i^u \to \mathcal{C}_j^u}{\sum_u \mathcal{C}_i^u \to \mathcal{C}^u} \tag{3.25}$$

where $\mathcal{C}_i^u \to \mathcal{C}_j^u$ denotes a pair of consecutive cities in the sequence $\mathcal{C}^u$. So $\sum_u \mathcal{C}_i^u \to \mathcal{C}_j^u$ is the total frequency of users traveling from city $\mathcal{C}_i$ to $\mathcal{C}_j$, and $\sum_u \mathcal{C}_i^u \to \mathcal{C}^u$ is the frequency of users traveling from city $\mathcal{C}_i$ to any other city. So this probability can be explained as: for the users who publish a tweet in the city $\mathcal{C}_i$, they have a chance of $P(\mathcal{C}_j|\mathcal{C}_i)$ to publish next tweet in the city $\mathcal{C}_j$, which is also the transition probability in our model.

The home location of a user has a great influence on the transition probability. For two users u_1 and u_2 in the same city C where u_1's home location is in C and u_2 is a visitor, the probability of u_1 to stay in C is much higher than u_2 and the probability of u_2 to return to his home city is very high as well. So, the home city greatly influences the transition probability between two cities.

To address this issue, we train transition probabilities separately based on home cities of users. We use U to denote all users in the dataset and $U_{\mathcal{C}_i}$ is the users with home city $\mathcal{C}_i$. For a user u with home city $\mathcal{C}_i$: 1) When u moves between city $\mathcal{C}_i$ and $\mathcal{C}_j$ (includes $j = i$), $P(\mathcal{C}_j|\mathcal{C}_i)$ are observed from all the users in $U_{\mathcal{C}_i}$; 2) When u moves from city $\mathcal{C}_j$ to city $\mathcal{C}_k$, $i \neq j$ & $i \neq k$, the training data are from users in $\{U - U_{\mathcal{C}_j}\}$. For the second case, we combine all the users whose home city is not $\mathcal{C}_j$ together due to the lack of enough training data for users in every city.

Observation Probabilities: The observation probability gives the likelihood of an ob-

servation from a given state. For geotagging, given a tweet T, there is an observation probability $P(T|\mathcal{C})$ for each city $\mathcal{C}$. From the geotagged tweets, we build the language model $\theta_{\mathcal{C}}$ for each city. The probability of term t from a tweet which published in city $\mathcal{C}$ can be estimated as: $p(t|\theta_{\mathcal{C}}) = tf_{(t,\mathcal{C})}/dl_{\mathcal{C}}$, where $tf_{(t,\mathcal{C})}$ is the frequency of term t in city $\mathcal{C}$ and $dl_{\mathcal{C}}$ is the total frequencies of all terms in city $\mathcal{C}$[66]. Then given the state of city C, the observation probability $P(T|\mathcal{C})$ is defined as:

$$P(T|\mathcal{C}) = \prod_{t_i \in T} P(t_i|\theta_{\mathcal{C}}) \tag{3.26}$$

In this work, we smooth the probability of $P(t_i|\theta_{\mathcal{C}})$ by using the Dirichlet smoothing [96]. The stop words[6] are removed before calculating the observation probabilities.

The probability of the initial state, $P(S_0 = \mathcal{C}_j)$, is given by the transition probability of $P(\mathcal{C}_j|\mathcal{C}_i)$, where $\mathcal{C}_i$ is the home city of the user. This is based on the observation that a user is likely to travel to any city from his home city and the first tweet observation is not necessarily from his home city. Then based on the Viterbi algorithm, we find the city sequence which can maximize the product of the observation probabilities and transition probabilities as the geotags of tweets. Assuming the total number of tweets of a user is N, the number of cities is $|\mathcal{C}|$, and each tweet contains $|T|$ terms, the time complexity of estimating the location of these N tweets is $O(N \times |\mathcal{C}|^2 + N \times |\mathcal{C}| \times |T|)$.

3.3. Experimental Results

In this section, we evaluate our user location estimation models in comparison with existing state-of-the-art methods on two data sets, i.e. the Gowalla and the Twitter datasets introduced before. We first show the estimation accuracy of the Friend Co-location Model and Local Social Coefficient Model with respect to the number of friends which illustrates why we combine these two models into our Social Closeness and Social Structure-Based Model. Then We test the effect of different parameters including the percentage of users

[6] http://xpo6.com/list-of-english-stop-words/

who provide their locations, the number of friends, and the number of iterations, and the error distance.

We first define the error distance of the estimation of user u_i as $Err(u_i)$, which represents the distance between the estimated location and the actual location of the user u_i. We consider the estimated locations with error distance less than 100 km as correct estimations. So the estimation accuracy can be represented as $\frac{|u_i|u_i \in U \wedge Err(u_i) \leq 100km|}{|U|}$.

The models we tested in the experiment are shown as follows:

- Friend-based method (FB) We take the friend-based method (FB) provided in [9] as the baseline method. In [9], the authors estimated a user location by his friend locations based on the relationship between distance and the probability of being friends. They proposed their estimation model as: $\prod_{(u_i,u_j)\in E} P(|l_i - l_j|) \prod_{(u_i,u_j)\notin E}(1 - P(|l_i - l_j|))$. Here $P(|l_i - l_j|)$ represents the probability of user u_i and u_j located with the distance of $|l_i - l_j|$ and E is the set of friend relation. Then they optimize the formula as: $\prod_{(u_i,u_j)\in E} \frac{P(|l_i - l_j|)}{1 - P(|l_i - l_j|)}$.
- Social relation based model (SR) This method is proposed in [46]. The authors apply three important methods to select the nearest friend: 1) the geometric median; 2) the minimum area formed by users and two of his friends (Oja's Simplex Median); and 3) there exists friend relationship between three users which is referred as the Triangle Heuristic.
- Friend Co-location Based Model: Our method.
- Local Social Coefficient Based Model: Our method.
- Social Closeness and Social Structure Based Model ($SoSS$): Our method.

We also test the FB, SR, and $SoSS$ methods combined with our confidence-based iteration model, which are noted as FB_I, SR_I, and $SoSS_I$. The default value for the percentage of location withheld is 25% for all experiments when not specified. At each time, the parameters used in our algorithms, like the probabilities $P(FCoI(u_i, u_j))$ in the Friend Co-location Model, will be retrained on the other 75% users. All accuracies reported is based on 3-time average on random sampling.

3.3.1. Experimental Results of Home Location Prediction

We firstly compare the Friend Co-location Model with the Local Coefficient Model. We will show the performance of friend co-location model and local coefficient model separately and explain why we combine them together. Figure 14(a) and 14(b) show the estimation accuracy of these two models on Twitter and Gowalla networks. From the figure, we can see that when users have less than 20 friends, the friend co-location based model can be much more useful than the local social coefficient model. It can help us to find out more important friends by the analysis of the relevance of friends distribution. The local social coefficient model performs worse because when a user has less than 20 friends, it is likely that there is no friend relation between his/her friends. With the increasing of the number of friends, the local social coefficient based model will perform better. The accuracy can be more than 60 percent on Twitter and 80 on Gowalla, which indicates that the analysis of the local tightness can be helpful. Since these two models perform quite differently, we combine them together to make sure that our model can work in different cases.

We then test those models under different settings of the two datasets. We randomly withhold 0%, 25%, 50%, and 75% users' location information of the two datasets and compare the performances of different models. Table 3.2 shows the results of each method.

From these results, we can see that our models can improve the estimation accuracy by 5 to 20 percent compared with the baseline methods. The $SoSS$ model combined with the confidence-based iteration achieves the best performance among them. This phenomenon demonstrates that the friend co-location and the local social coefficient can help us to detect the more important friend or group of friends effectively in the estimation.

Another observation is that the confidence-based iteration process contributes greatly in the estimation accuracy, especially in the cases when only a small portion of users have their location information known in the dataset. From the estimation results, the iteration process can improve the accuracy by 1 to 3 percent when we withhold 25% users' location information. This ratio increases to 5 to 22 percent when 75% of users' location is unknown. When more users' location information is unknown (from 25% to 75%), the estimation

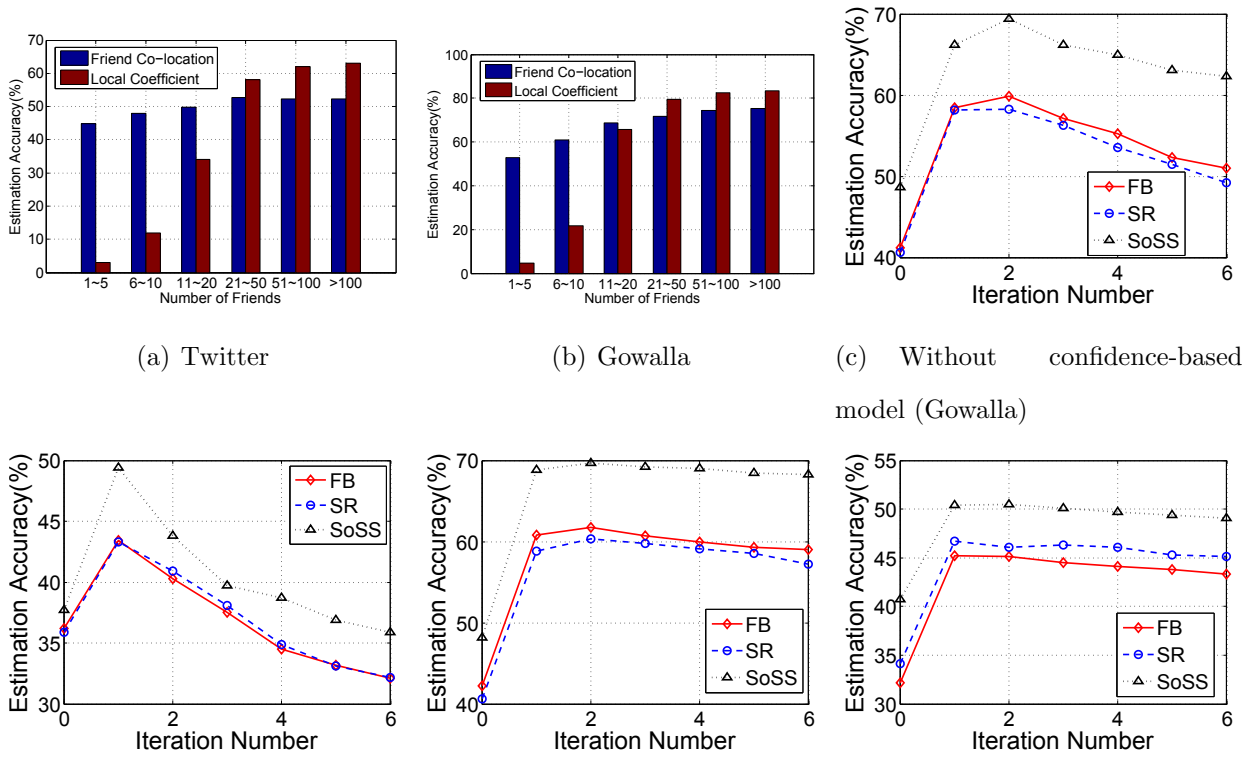
% locations withheld, by platform	FB	FB_I	SR	SR_I	$SoSS$	$SoSS_I$
Twitter 75%	35.7	43.5	35.9	43.7	40.9	48.6
Twitter 50%	41.8	46.5	43.1	48.0	47.3	52.5
Twitter 25%	48.0	48.5	48.3	50.2	54.6	55.2
Twitter 0%	48.3	–	51.8	–	55.6	–
Gowalla 75%	47.1	62.7	44.9	60.8	50.1	70.8
Gowalla 50%	64.7	69.3	61.5	69.0	68.7	77.5
Gowalla 25%	73.0	73.5	71.2	73.1	79.6	80.3
Gowalla 0%	74.2	–	72.9	–	80.6	–

TABLE 3.2. Effect of percentage of unknown locations

accuracy of those models without iteration process (FB, SR, and $SoSS$) will decrease by more than 15 percent on both Twitter and Gowalla datasets. However, at the same time, the decreasing of the accuracy of the models with iteration process is just about 7 percent on the Twitter dataset and 10 percentage on Gowalla dataset. This phenomenon tells that the confidence-based iteration model can help us to overcome the problem of the sparsity of user location.

The models work better on Gowalla dataset. We can explain this phenomenon by analyzing the difference of user behavior on Gowalla and Twitter. Users who use Gowalla tend to add friends who live close to their locations (27% within 100 km), and Twitter users will add many users which may live far away from them and only 12% friend pairs are located within 100 km.

Then we investigate how the iteration process improves the estimation accuracy and explain why we introduce the confidence-based method. Here we withhold 75% user location information and estimate their locations. Figure 14(c) and 14(d) show the estimation accuracy of the iteration method without the confidence based selection. When the iteration number is 0, it is the accuracy of the basic models. The iteration process can improve the estimation accuracy by nearly 20 percent on the Gowalla dataset and 10 percent on the Twit-



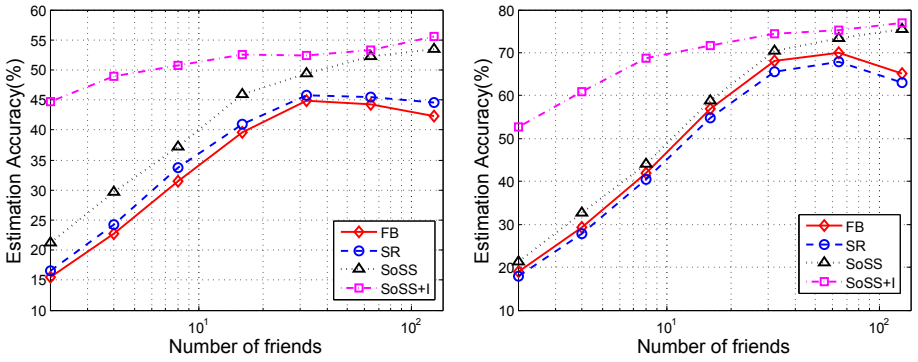
(a) Twitter (b) Gowalla (c) Without confidence-based model (Gowalla)

(d) Without confidence-based model (Twitter) (e) With confidence-based model (Gowalla) (f) With confidence-based model (Twitter)

FIGURE 3.14. The difference between the friend co-location method and the local social coefficient method and the influence of confidence-based iteration process.

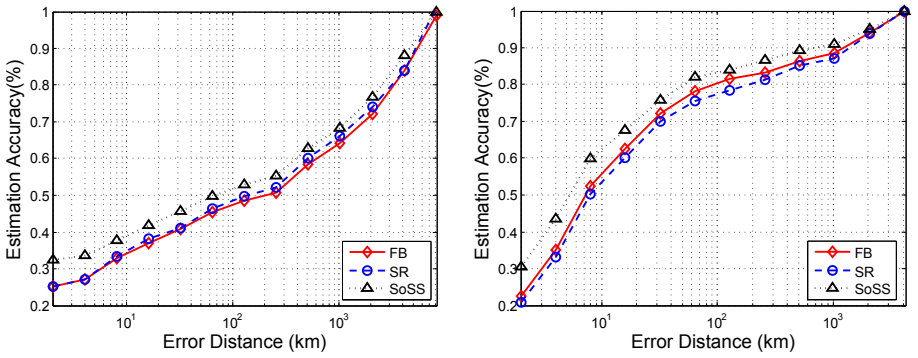
ter dataset. However, as the iteration process continues, the estimation accuracy begins to decrease. Prominently, the accuracy will decrease by nearly 10 percent in the experiment on the Twitter dataset after the sixth iteration. So we introduce the confidence-based iteration method as shown in Figure 14(e) and 14(f). The confidence-based method can prevent the estimation accuracy from decreasing and keep it to stable and higher values of accuracy. In most cases, this process can achieve the best accuracy in three times iteration.

Then we investigate the influence of the number of friends. Figure 3.15 gives the summary of the estimation accuracy of groups of users with a different number of friends. The estimation accuracies of all the models without the iteration step are very low when



(a) Twitter (b) Gowalla

FIGURE 3.15. Influence of number of friends.



(a) Twitter (b) Gowalla

FIGURE 3.16. The estimation accuracy under different error distances.

the users have few number of friends. That is because: 1) if a user only has a few friends, none of his/her friends may have their location information known. So we cannot perform the estimation by their friends' location. 2) It is difficult to decide which friend may locate closer to the user if his/her friends do not cluster. The performances of *SoSS* model are better than the FB and SR models on both the datasets. This is because our *SoSS* model distinguishes friends based on friend co-location and local coefficient indexes. Choosing socially close friends for location estimation improves the results.

On the other hand, the confidence-based iteration method can help to improve the estimation accuracy by more than 20 percent for the users who have few number of friends. With the iteration process, more users who don't have location information in the social

network will be given an estimated location, and those estimated locations can be used in estimating others' location to help to overcome the first problem above. So after we combine those models with the confidence based iteration model, many of the users who have a few friends can also be estimated correctly. When the users have more than 30 friends, the confidence based iteration model cannot help much on the accuracy.

We test the estimation accuracy on different values of error distance and show the results in Figure 3.16. The accuracy of each method will be close to 1 when the error distance is more than 3,000km. Our method performs better when we set a smaller error distance. When the error distance is larger than 200km, the accuracy of different models will get close.

3.3.2. Real-Time User Location Estimation

Knowing a user's home helps to understand user movement patterns. Users are bound by travel distance from his home and the types of outbound cities from home affect visiting possibility. However, only 16% users register their city-level home locations [56] and collecting the home location from user profile is strictly limited by the rules of Twitter API.

Previous work has provided different methods for estimating the home locations of users [57, 79, 24] based on analyzing content information or geographic distribution of friends. In this work, we assign home location following the method used in [78, 25]. A set of tweets of each Twitter user is selected to estimate the home location (this set will not be used for building and testing geotagging models). These tweets are assigned into different cities according to their geotags. We use city boundaries from the United States Census Bureau[7] to decide which city a geotagged tweet belongs to. Then we assign the city with the most number of tweets as a user's home. In [78], the authors show that the home location obtained in this way has an 85 percent accuracy.

We then select all cities with more than 5,000 users. There are 40 cities C in our dataset. We only keep the tweets: (1) tweeted by users in C; (2) published in C. Figure 3.13 shows the number of users and the number of geotagged tweets in these 40 cities. The top

[7] http://www.census.gov/en.html

three cities with the most number of users and tweets are Los Angeles (46,508 users;1,183,634 tweets), Chicago(30,689 users; 634,665 tweets), and New York(30,689 users; 419,130 tweets).

We implement four algorithms and compare their performances with different cases. To reduce the impact of spammers and robots, we only keep users with a number of geo-tagged tweets between 5 and 500 in our dataset [72]. The first quarter of tweets (sorted by published time) of users are used to estimate home cities. We then run a 10 fold cross validation on the rest of the dataset where we first divide users into 10 equal parts and then assign the tweets into 10 corresponding folds depending on where their authors belong to. So all of the tweets from the same user will be assigned to the same folder.

We use the random selection method as the baseline method and select two methods proposed by the related works to compare with our algorithm.

- Random selection (Random): Select a city as the geotag location of tweet randomly.
- Bayesian: The Bayesian-based method described in introduction [48].
- Kullback-Leibler divergence (KL divergence): The KL divergence between tweet T and city C is calculated as equation 3.27 and the geotag is based on the ranking of the KL divergence [48, 72].

$$KL(\theta_T|\theta_C) = \sum_t p(t|\theta_T) log \frac{p(t|\theta_T)}{p(t|\theta_C)} \tag{3.27}$$

- HMM-based: The method proposed in this dissertation.

A tweet is assigned to a city as its geotag. We then assign the coordinate of the city center as the predicted location of the tweet. The city center we used is listed in Wikipedia [8]. We use three measures:

- Accuracy (Acc). The percent of tweets which are predicted in the same city where they are published.
- Mean Error Distance (km): The average error distance between the predicted location and actual location.

[8] https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population

- Median Error Distance (km): The median error distance between the predicted location and actual location.

In this dissertation, we define the distance as the geographic distance.

	Overall			Non-Home City			Home City		
Methods	Mean	Med.	Acc.	Mean	Med.	Acc.	Mean	Med.	Acc.
Random	2847	2450	3.86	2252	2334	4.9	2845	2439	3.8
Bayesian	1402	1022	36.7	1245	765	31.8	1375	994	36.8
KL divergence	1318	954	38.8	1191	741	32.8	1301	944	38.9
HMM-based	52.6	1.82	95.1	997	181	34.6	52.4	1.82	96.3

TABLE 3.3. Mean error distance (km), median error distance (km), and accuracy(%) of the geotag estimation results.

Table 3.3 shows that our HMM-based methods can significantly improve the performance of geotagging compared with related works. The overall accuracy is improved by more than 50%; and the mean error distance and median error distance of our method are orders of magnitude better.

We are interested in learning if the improvement is mainly due to the reason that more than 90% of tweets are published in home cities which tend to be easier to estimate. We separate the performance for home cities and non-home cities. Since more than 90% tweets are published in home cities of users, all of these algorithms achieve similar performances in the home cities with the overall case. For the difficult case of non-home cities, the two algorithms from related work, the Bayesian-based method and the KL divergence method, perform better than they do in home cities. This illustrates that when users travel to non-home cities, they may tend to use terms with local characteristics. By contrast and when users stay in home cities, their words are more commonly used with less regional information. However, for the tweets published in these non-home cities, our HMM-based model still outperforms other two methods by nearly 2% in accuracy and the error distances

are significantly smaller at the same time. Compared with related works, the improvement on error distances demonstrates that even for the incorrectly estimated cases, our model can locate the tweet with a closer city from the real location of the tweet due to the benefits from the usage of patterns of user movements. In general, the results demonstrate that integrating home city and movement will benefit geotagging in both home and non-home cities.

3.4. Demo System for Home Location Prediction

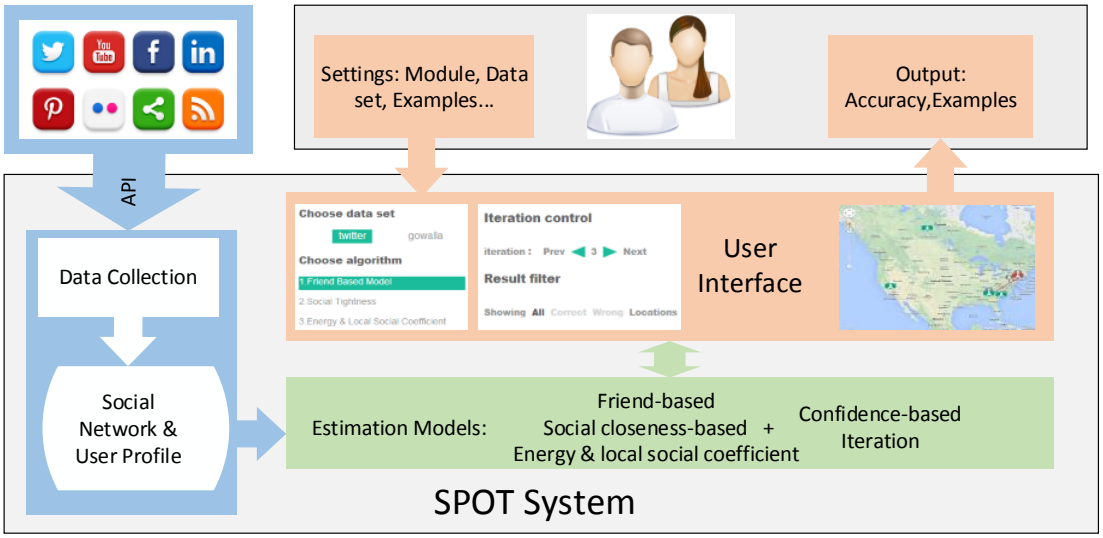


FIGURE 3.17. Architecture of the demo System: The main components includes the data collection and preprocessing module, the estimation algorithms, and the user interface.

In this demo, we show estimation models performing on large scale social media datasets. The demo system allows users to test different estimating models and view a detailed location estimation process. By selecting different estimating models, testing data sets and other settings, users can test and compare the accuracy of those algorithms under different environments. To get more information about the running process, we also provide a map view of the testing examples. The estimation cases (users on those social media platform) can be chosen from the user interface and then shown on the map. One

can check their real location, estimation locations, and the locations of their friends so that the estimation process can be visually observed. The demo website is available from http://hpproliant.cse.unt.edu/locationdemobeta.

3.4.1. Demonstration Scenario

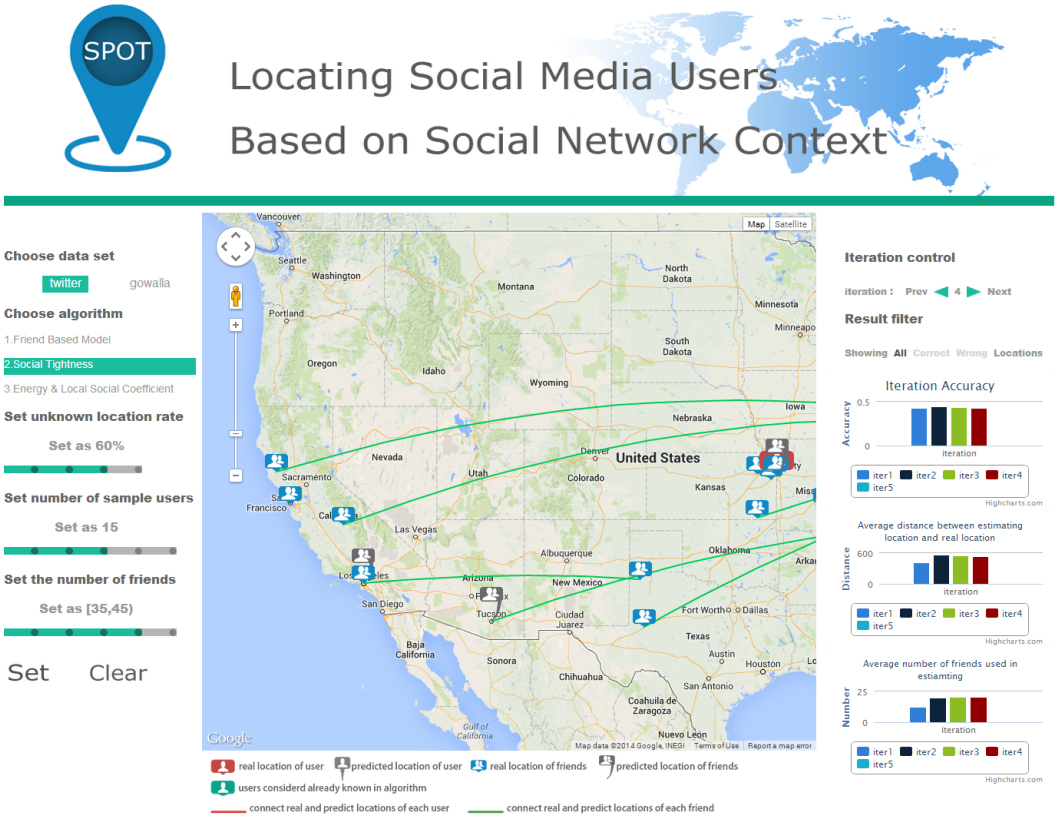


FIGURE 3.18. User interface: there are three important panels in this demo. 1) Setting Panel: The setting panel is in the left of the demo page. Users can select a dataset, algorithm, the unknown location ratio, and the number of samples to show etc. 2) Statistics Panel: We show different statistics of the estimation results in the right column of the demo page. Those statistics include the estimation accuracy, average error distance, and the average number of friends. 3) Visualization Panel: We use a map to show selected samples. Users can see real location, estimated location, and the locations of their friends.

Our data sets come from two different social network applications, Gowalla and Twitter, and both of them are publically available. Gowalla is a location-based social network and users are able to check in at "spots" in their local vicinity. The Gowalla dataset [25] contains 196,591 users[9]. We use 99,563 of those users who have check-in records in our demo and each of them has 4.8 friends on average. Since there is no user profile, we take the center of the $25km \times 25km$ area with the most number of check-ins as the user home location. We also collected user profiles from Twitter, an on-line social networking and microblogging service which enables users to follow each other and read "tweets". There are totally 660,000 users[10] and their social relation in the Twitter dataset and we collected 148,860 users' locations through Twitter API. We define the friend relationship the same way as that in [44]. Users A and B have friend relation if they follow each other. Each user has 29.4 friends on average in the Twitter dataset.

The screenshot of our demo system is shown in Figure 3.18. To begin the demonstration, users need to choose several settings include: (1) data sets selection: users can run the models on either the Twitter data set or the Gowalla data set; (2) model selection: include the friend-based model, the social tightness based model, and the energy and local social coefficient model; (3) location mask: users can hide certain percent of user location information (from 20% to 80%). In the estimation process, those users' location will not be used. In this way, we can test the tolerance of the models on the sparsity of location information; (4) estimation sample selection: users can select estimation samples by their friends number and the estimation results(correct or incorrect) and only the samples which meet the requirements will be visualized on the map.

By click the "Set" button, the demonstration will run the first time estimation. The output includes statistics of the estimation results and selected samples. The statistics results include estimation accuracy, the average distance between the estimated location and real location, and the average number of friends used in the estimation. Users can

[9]http://snap.stanford.edu/data/loc-gowalla.html

[10]http://dmml.asu.edu/users/xufei/datasets.html

click the iteration button to enter the iteration process. The system will use the estimation results in next time estimation process. The figures of those statistics will be generated and dynamically updated after each iteration. Selected estimation samples will be visualized on the map showing both their real and estimated locations. By clicking a sample, the sample's friends' estimated/real location will also be shown on the map. A user has the choice of visualizing all samples, the corrected estimated samples, or the incorrectly estimated samples as well. In this demo system, we define an estimation result as correct when the distance between the estimated location and real location is less than 100 miles. All the models can be combined with the iteration method.

CHAPTER 4

EVENT DETECTION ON SOCIAL MEDIA

In this section, we propose the event detection algorithm and the experimental results on the Twitter dataset.

4.1. The Event Detection Algorithm

We first give a brief description of the event detection algorithm. The event detection algorithm has three main steps: detecting index terms, tweets clustering, and event extraction. In the first step, detecting index terms, we try to detect terms which may be related to an event. The method is based on anomaly detection. For each term, we record the frequency of a term in tweets published every day as the term distribution. The total frequency of all terms in each day is recorded as overall distribution. Hashtags and @ mentions in tweets are also considered as terms in this work. By analyzing the correlation between term frequency and overall frequency, we can obtain a list of terms which distributed differently from overall distribution. These terms that occur abnormally will be taken as index terms.

In the second step, we will assign tweets into clusters. For each index term, the tweets which contain it are clustered by their locations and published time using a density-based clustering method. A tweet containing multiple index term may be assigned to multiple clusters. Then clusters for all terms are further clustered since an event may be related to several different index terms. To remove "noise" clusters which may be not related to an event, we propose a filtering method. About 70 percent of clusters will be removed in this step. Finally, for each cluster, a document summarization algorithm will be used to obtain a description of the event. We further extract the time and location information from the tweets to help estimate when and where the event happened.

4.1.1. Detecting Index Terms

Detecting outbreaks of tweets and terms is one of the most important existing methods to detect events. However, only global events attract a huge amount of interests. Local and

t, t^l	Tweet, published location
e, E	A term from tweets/set of all the terms
C	Cluster of tweets

TABLE 4.1. Notations and parameters

small events may only involve a few users on Twitter and these events cannot raise an outbreak of the number of tweets. So in this work, instead of looking at the outbreak of the number of tweets, we focus on detecting the abnormal distribution of terms. Term level distribution is more sensitive to events since when people discuss an event, some terms are more likely to be rare in daily conversations. As the result, even there are only a small number of people discussing a local event, we can still detect the abnormal occurrence of some terms. In this work, we define terms that distribute abnormally as index terms and in the following step, these index terms will be used to help detect tweets which may be related to an event.

The first challenge is how to detect index terms. Because the number of active users keeps changing in the different time of a day, the frequency of terms is greatly affected by time. The peak number of terms may happen just because of more users online at that time. So our idea is measuring the correlation between the term distribution and overall distribution. If a term is related to an event, it typically just appears during the event and is not affected by the total number of active users. As a result, the correlation between the distributions of this term and all terms will be lower.

Another challenge is how to detect index terms effectively. When we combine tweets from everywhere together, an event, e.g. traffic jam, may keep happening in different areas and the related terms are likely to have a high correlation coefficient with the total number of tweets. So in this work, we divide the tweets into groups based on their location. Since users are not distributed equally, we use k-d tree [11] to divide the tweets geographically. A k-d tree is a space-partitioning data structure for organizing points in a k-dimensional space. Every non-leaf node is divided into two parts that separate tweets into two equal

Algorithm 2: Event detection on Twitter

Data: Geotagged tweets

Result: Location, time, descriptions and kewords for detected event

Begin:

Divide tweets based on their locations by k-d tree;

for *Each node of the k-d tree* **do**

  for *Each term* **do**

    Calculate the correlation between term and total distribution;

    if(Correlation ¡ 0.4);

      Add term e to index terms E_I;

  end

end

for $e \in E_I$ **do**

  Cluster the tweets which contain e by DENCLUE algorithm ;

end

Merge clusters by their time, location distribution;

Filter clusters which may not be related to a local event;

for *Each cluster* **do**

  Extract time, location, keywords, and descriptions;

end

return Time, location, keywords, and descriptions of events

groups through the median of one axis. Firstly, all the tweets will be put in the root node of the k-d tree. In each step, the node will be divided into two parts by finding a line which can equally separate the tweets in the node. This process will be stopped when the number of average daily tweets in the node is less than 10 thousand.

Then for each leaf node of the k-d tree, we detect the index terms using correlation coefficient. Here we use t to represent a tweet and for each tweet, it contains a set of terms

$\{e_1, e_2, \dots\}$. We use t^p and t^l to denote the time and location when and where the tweet is published. We use E as the set of all the terms. For a term e, we count the frequency of term e in each day: $\vec{e} =< fre^1(e), fre^2(e) \cdots >$, and the overall distribution is denoted as $\vec{E} =< fre^1(E), fre^2(E) \cdots >$. We then use the Spearman correlation coefficient to measure the correlation between the two distributions as:

$$r < \vec{e}, \vec{E} >= \frac{cov(\vec{e}, \vec{E})}{\sigma_{\vec{e}} \sigma_{\vec{E}}} \tag{4.1}$$

Here $cov(\vec{e}, \vec{E})$ is the covariance of the two distributions. $\sigma_{\vec{e}}$ and $\sigma_{\vec{E}}$ are the standard deviations of the distributions.

4.1.2. Tweets Clustering

After detecting index terms, we try to use these terms to help to find the tweets related to an event. For each index term, we first collect all the tweets which contain the term. The tweets published in the k-d tree nodes where the term is not considered as index term will be removed. Then these tweets will be clustered by their time and location. Considering that during an event, there may be a quick increase of tweets related to the event and in most time, there may be only a few tweets discussing the event, we use a density-based approach to cluster the tweets [39]. The basic idea of this approach is that the influence of each data point can be modeled by an influence function and this function can be seen as a description of the impact of data points within its neighborhood. We use this method because it works well in dealing with outliers and noises and flexible for clusters with different sizes and shapes.

Definition[Influence Function] The influence function between tweets are defined as:

$$f_{t_i}(t) = e^{-\frac{d(t^l, t_i^l)^2}{2\sigma^2}} \tag{4.2}$$

Here $d(t, t_i)$ is the geographic distance between tweets t and t_i.

Definition[Density Function] With the influence function, the density function of tweet t is defined as the sum of total influence of other tweets:

$$\mathcal{I}(t) = \sum_{i}^{N} f_{t_i}(t) \tag{4.3}$$

In the algorithm, we only count the tweets published within the distance of 4σ [39] and in the past or following ψ hours.

Definition[Gradient] The gradient of $\mathcal{I}(t)$ is defined as:

$$\nabla\mathcal{I}(t) = \sum_{i}^{N} (\vec{t_i} - \vec{t}) f_{t_i}(t) \tag{4.4}$$

Here $\vec{t_i}$ and $\vec{t}$ mean a data point represented by the location (latitude, longitude) and time of the tweet in a three dimensions space.

Definition[Density-Attractor & Density-Attracted] A tweet t is a density-attractor iff $\mathcal{I}(t)$ is a local maximum of the density function. A tweet t_i is a density-attracted tweet for t when there exist a path to t that the gradient is continuously positive:

$$\vec{t^0} = \vec{t}, \vec{t^k} = \vec{t^{k-1}} + \delta \cdot \frac{\nabla\mathcal{I}(t^{k-1})}{||\nabla\mathcal{I}(t^{k-1})||} \tag{4.5}$$

Different event has different types of locations. For example, for sports matches, the location can be considered as a point. For the tornado, the location can be an area. Constrained by geographic boundaries, population distribution, and some other factors, the influence area of an event may have an arbitrary shape. So we use an arbitrary-shape clustering method here. As shown in algorithm 3, the density attractors can be merged if there is a path between them with the density function continuously exceeding ξ.

The quality of the clustering results depends on the choice of the parameters for most clustering algorithms. We choose the value of the parameters of σ, ψ, and ξ with the method provided in [39]. However, because of the particularity of the dataset, we need to take some special situations into consideration when deciding the value of σ and ψ. These two parameters can determine the influence of each point. They are considered separately because they have different influence models in their dimensions. For the parameter of ψ, which controls the influence of points on the dimension of time, we test different values.

Algorithm 3: DENCLUE algorithm on geotagged tweets

Data: Geotagged tweets

Result: Clustering results

Begin:

Uses hypercubes of with edge length 2σ and only populated cubes are saved;

Calculate the density function for each tweet;

Using hill climbing to detect density attractors;

Adding density attracted points to the cluster of density attractors;

Merge the clusters if a path exists between two density attractors with the density function continuously exceeds ξ;

return Clusters of tweets

Then for the clusters obtained from algorithm 3, we will future merge the clusters by their time, locations because an event may be related to several index terms. For each cluster C, we use normal distributions $C_{time} \sim \mathcal{N}(\mu_{Ct}, \sigma_{Ct}^2)$ to fit the distribution of published time of tweets of the cluster and the location distribution $C_{location} \sim \mathcal{N}(\mu_{Cl}, \sigma_{Cl}^2)$. If two clusters come from the same event, we can consider them as two different sampling of tweets from all the tweets related to the event. So our goal is to test the relationship between two clusters. Here we assume that cluster C is sampled from an event and the mean location and time of the event is μ_{Cl} and μ_{Ct}. Given two clusters C_1 and C_2, we can estimate whether they are sampled from the same event by using student's t-test.

With the location of all the tweets in C_1 and C_2, we can obtain the mean location of these two clusters l_{C_1} and l_{C_2} and the variance of them $\sigma_{C_{l1}}^2$ and $\sigma_{C_{l2}}^2$. With the hypothesis:

$$H_0 : \mu_{C_{l1}} - \mu_{C_{l2}} = 0 \quad H_1 : \mu_{C_{l1}} - \mu_{C_{l2}} \neq 0 \tag{4.6}$$

we need to test:

$$Z_l = \frac{(l_{C_1} - l_{C_2}) - (\mu_{C_{l1}} - \mu_{C_{l2}})}{\sqrt{\frac{\sigma^2_{C_{l1}}}{n_1} + \frac{\sigma^2_{C_{l2}}}{n_2}}} \tag{4.7}$$

Here n_1 and n_2 are the sizes of cluster C_1 and C_2. If $|Z| < |Z_{\alpha/2}|$, we will consider that C_1 and C_2 are related to the same event. We also test the time distribution of C_1 and C_2 in the same way. Assuming that the mean publishing time of tweets in C_1 and C_2 are t_{C_1} and t_{C_2} and the variance of them are $\sigma^2_{C_{t1}}$ and $\sigma^2_{C_{t2}}$, then we have:

$$Z_t = \frac{(t_{C_1} - t_{C_2}) - (\mu_{C_{t1}} - \mu_{C_{t2}})}{\sqrt{\frac{\sigma^2_{C_{t1}}}{n_1} + \frac{\sigma^2_{C_{t2}}}{n_2}}} \tag{4.8}$$

In this work, we set the value of α as 0.05.

Algorithm 4: Cluster merging algorithm

Data: Clusters of tweets

Result: Merged clusters

Begin:

Let $\mathcal{C}_{clustered} = \emptyset$ and add all the clusters to $\mathcal{C}_{clustering}$;

for *Each* $C_i \in \mathcal{C}_{clustering}$ **do**

  $C_{merge} = C_i$;

  for *Each* $C_j \in \mathcal{C}_{clustering}$ *(R-tree index)* **do**

    Calculate $|Z_t|$ and $|Z_l|$ between C_{merge} and C_j;

    if($|Z_t| < |Z_{\alpha/2}|$ and $|Z_l| < |Z_{\alpha/2}|$)

      Merge C_j to C_{merge} ;

      Remove C_j from $\mathcal{C}_{clustering}$;

  end

  Add C_{merge} to $\mathcal{C}_{clustered}$;

end

return Merged clusters of tweets

As shown in algorithm 4, for the clusters related to different index terms, they can be merged if and only if their time and location distribution meet the requirements of $|Z_t| < |Z_{\alpha/2}|$ and $|Z_l| < |Z_{\alpha/2}|$. When seeking for a cluster C_j which can be merged into C_{merge}, we use R-tree index to help to find the clusters which distribution overlapping on time and space with C_{merge}.

4.1.3. Cluster Filtering

Before extracting the event information from clusters, we firstly introduce the methods we used to filter the detected clusters. From the clusters generated by the method introduced above, we find two types of clusters that significantly affect the quality of the results of event detection.

The first type of clusters is generated by events or news which have a global influence. As we discussed above, many large events or news may have many Twitter users discussing them in different cities. Sometimes the number of tweets is large enough to form several clusters in different areas. So firstly we want to remove such clusters. There are two different cases: 1) The clusters are generated by a news. We need to remove all the clusters related to the news; and 2) The clusters are generated by a large local event with global influence. We need to remove the clusters in other places other than the place where the event happens. We filter the cluster by the following steps: horizontally detect related clusters, calculate the local weight of each cluster, detect the outlier, and then remove the clusters.

For each cluster, we detect its similar clusters in other areas by analyzing their index terms and time distribution. For two clusters in different areas, they will be considered as similar when they share more than 80% common index terms with each other and their t-test (equation 4.8) of time distributions meet the requirement of $|Z_t| < |Z_{\alpha/2}|$. The similarity is defined as a transitive relation, which means that if cluster a is similar to cluster b and b is similar to cluster c, a is also similar to c. We then define the local weight of a cluster as $C_{lw} = \frac{NumberOfTweetsIn\ C}{NumberOfLocalTweets}$. The number of local tweets is the tweets published within a certain distance (2 km in our experiment) and time (e.g. half an hour) to at least one tweet in C. Then for all the similar clusters, we apply a Gaussian distribution to fit the

distribution of the value of local weights of the clusters. If there exists a cluster located out of 3 standard deviations of the mean and larger than the mean value, we will keep this cluster and remove others. If there exists none or more than 1 clusters, we will remove all the clusters.

For the second case, we find that there may be lots of similar tweets with just one or two words difference published in a short time in an area. In most cases, they are just meaningless tweets. However, they may lead an abnormal distribution of terms and detected by our algorithm. So for each cluster, we will use a term level cosine similarity to measure the common terms between tweets. If there is a group of tweets in a cluster with: the cosine similarity between each two of them are higher than 0.5 and the number of tweets in the group is larger than half of the total number of the cluster, we will remove this cluster.

4.1.4. Event Extraction

After the clustering step, we extract the time, location, and content information from each cluster as the description of an event.

Different event has a different type of influence area and time. For example, events such as sports games or concerts have a clear location where they are happening. However, many events do not have a specific point location and time, e.g. some natural disasters. In [74], the authors use a Poisson distribution to fit the distribution of the number of tweets after an earthquake. It makes sense because users begin to discuss this only after it happened. The distribution of time of tweets can be fitted into an exponential distribution. For other events, such as procession or a football match, people may discuss the event before it happens. In this work, we estimate the time and location by the density attractors. For each cluster of tweets, we can obtain the density attractors of this cluster by the method in section 4.1.2. The average time and location of these density attractors will be used as the time and location of the event.

For extracting the keywords from tweets in the cluster, we use a PageRank-based algorithm[98]. Each term corresponds to a node in the graph. For each tweet, we add the weight of edges between two index terms in the tweet by 1. The weight of each node is

initialized as 1. Using PageRank algorithm, finally, we select top 5 terms with the highest weights as the keywords of the event.

In [45], the authors propose a summarization method for a set of tweets. Each term in a tweet is assigned a weight as equation 4.9 to 4.11:

$$W_e = tf(e) * log_2(idf(e)) \tag{4.9}$$

$$tf(e) = \frac{\#OccurencesOfTermInAllTweets}{\#TermsInAllTweets} \tag{4.10}$$

$$idf(e) = \frac{\#Tweets}{\#TweetsInWhichTermOccurs} \tag{4.11}$$

Each term is assigned a weight based on its term frequency (tf) and inverse document frequency (idf). Then the weight of a tweet is defined as the average weight of terms it contains:

$$W_t = \frac{\sum_{e \in t} W_e}{Terms\ in\ t} \tag{4.12}$$

For each cluster, we select the top 5 tweets with the highest weight to describe the event.

4.2. Experiment

In this section, we test our algorithm on the Twitter dataset. Firstly we introduce the dataset used in this work and give the definition of local events. We use precision and case study to show the results of our event detection algorithm. Then we analyze the types of events we detected to show the differences between our algorithm and previous methods. Finally, we provide more details of the detected events to help to get a better understand of our algorithm.

4.2.1. Experimental Settings

We choose the value of the parameters of σ, ψ, and ξ with the method provided in [39] for the clustering algorithm. In fact, the clustering results, the number of clusters and the average number of tweets in each cluster, have little difference when we change the value from 2 to 6 hours. Considering the time complexity, we finally choose the value of ψ as two hours. When choosing the value of σ, there is another problem that most geotags of tweets are given by a square with the side of about 10 km. So we cannot obtain the accurate location of the tweet. In this work, we use the center of the square as the location of the tweet. However, in this way, we may have more multiple density-attractors for each cluster. That is also one of the reasons why we chose the arbitrary-shape clustering method. Since most of the tweets have no accurate coordinates, we assume that they are randomly distributed in the square. So finally we use the standard deviation of this distribution, 7km, as the value of σ.

We propose a detailed description of the targeted local events. Currently, there is not a clear definition of what is a local event. An event is loosely defined as a happening which has a number of participants getting together at a specific location at a specific time. We also follow this idea in our work. However, compared with previous work, in our work we apply more strict criteria when judging whether an event is a local event:

- Time and location: A local event must have a clearly defined set of participants, location, and time. The participants can be the people who attend an event e.g. sports games or affected by an event e.g. earthquake. Only events which happen in a certain location or area will be considered as a local event. For example, national festivals, such as July 4th, is not a local event, but a firework in a park which is used to celebrate it will be considered as a local event;
- Regional requirement: A local event must belong to a certain location. For example, events such as E3 2015 or NBA finals are global events that people discuss anywhere. In large cities which are not the location where the events are happening, there may still be a large number of related tweets published. Tweets can form large clusters in

many cities with similar characteristics of being an event. However, for such cases, we only take the event extracted from the cluster at the location where the event happens as local event;

- Events are different from news. Although some news has a specific time, location, and participants, it is still news rather than a local event. For example, a soccer player transferring from a team to another is considered as news. For such news, it is difficult to provide a clear description of time and location information. In this work, we only consider these happenings as news.

4.2.2. Experimental Results

We provide a detailed demonstration of the detection results. Both the precision analysis and case study will be used to show how our algorithm works and the advantages of our algorithm.

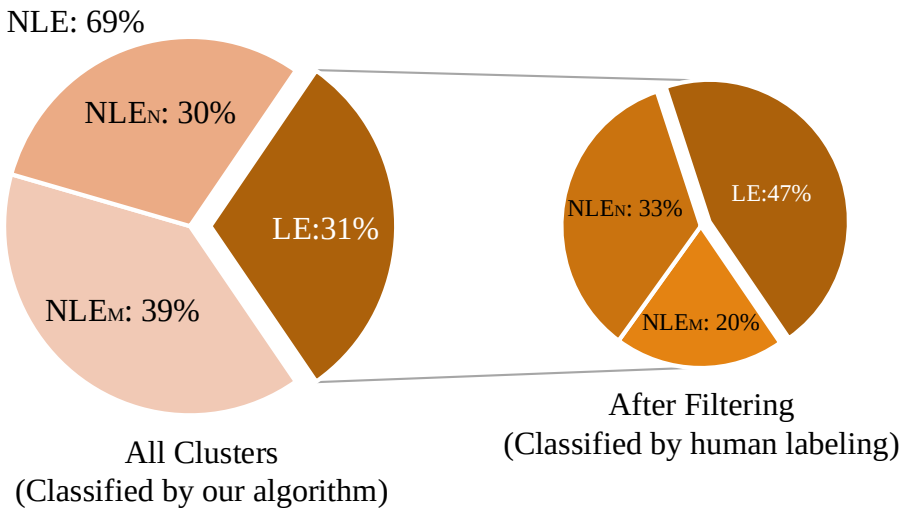


FIGURE 4.1. Proportion of different clusters.

From the Twitter dataset we collected, our LEDS system first obtains a set of candidate clusters which may be related to local events. Then we filter the clusters. We divide the clusters into two classes based on whether they are related to a local event (LE) or not (NLE). The clusters which are not related to a local event (NLE) are further divided

Event id	Date	Index Terms	Descriptions	Event Type
1	Jun 9	1	#teenchoice i nominate @5sos for #rowyso	T1
2	Jul 4	1	goonies + fireworks tonight! #hollywoodforevercemetery	T3
3	May 28	1	@sunilrawat @johnolilly i spend way too much time there.	T2
4	Jun 3	1	#teamhouse meeting at @rocknfishlalive with @tamar_house.	T3
5	Jun 3	2	happy #ax2015 everyone.	T1
6	Jun 26	2	@jaclynglenn protesting the saudi consulate in los angeles.	T2
7	Jun 1	1	caitlyn deal rey. http://t.co/y5lwsilxba.	T2
8	May 30	1	mistakes we make can paint a better picture later on.	T3
9	May 28	1	24 minutes till earthquake.	T2
10	May 30	2	saul rodriguez vs. antonio capulin.	T2
11	Jun 16	3	#e32015 adventures! http://t.co/i2hqqm4jzw	T1
12	Jun 13	1	there is a predator alien dancing at this party.	T2
13	Jun 3	1	can't wait for #animeexpo2015 !!!	T2
14	Jun 30	1	cool thunder over long beach!	T2
15	Jun 22	1	@johnny_pypes how's that golf game?? #thelastship	T3

TABLE 4.2. Selected local events in Los Angeles.

into two subclasses: a global but not local event (or news) (NLE_N) and the noise clusters (NLE_M). Please note that based on our definition of local events, when a local event with a global influence happens, we only take it as a local event at the location where it happens.

For other locations, we consider it as news rather than a local event.

Figure 4.1 shows the results of our filtering method. The left pie chart is the filtering results. 31% of the clusters detected by our algorithm are considered as related to local events and 69% of the clusters are taken as non-local event. Among the 69% clusters, 30% of them are considered as related to a news (NLE_N), and 39% of them are considered as noise clusters (NLE_M).

After the filtering step, there are 31 percent of clusters left and considered as local events by our algorithm. Then we manually label these clusters. For each cluster, we extract the time, location, keywords and top 5 tweets in the clusters to describe the event. For each detected event, we have three people to label whether the cluster is a local event (LE), news (NLE_N) or noise cluster (NLE_M) based on these information.

We randomly select 2,000 clusters from the results and manually labeled the clusters by three people and here we use the majority rule. Each result will be considered as a local event only when it has two or more positive labels. The right pie chart of Figure 4.1 demonstrates the results of the human labeling. 47% of the clusters after filtering are labeled as a local event. So the precision of our algorithm is 47%. There are also 20 percent of the clusters are labeled as noise clusters and 33 percent of clusters are labeled as news.

Then we list detected events to show the results on the real world data set. In Table 4.2, we randomly selected 15 events happened in Los Angeles from May 25 to July 8, 2015. In this table we show the date of the event, the number of index terms related to the event, and the top one tweets which are selected to describe the events. We also show the locations of the first 12 events in the Figure 4.2.

There are several large events which have global influence such as E3 2015, but more most of them are "smaller" events. They don't have a large influence and some of them even will not be mentioned in any news media. Compared with previous methods, this is an important advantage of our algorithm. For example, on July 4th, we detected a large number of tweets mentioning the keywords such as "Independent Day" or just "4th of July". However, it is not a local event and clusters containing these keywords will be filtered

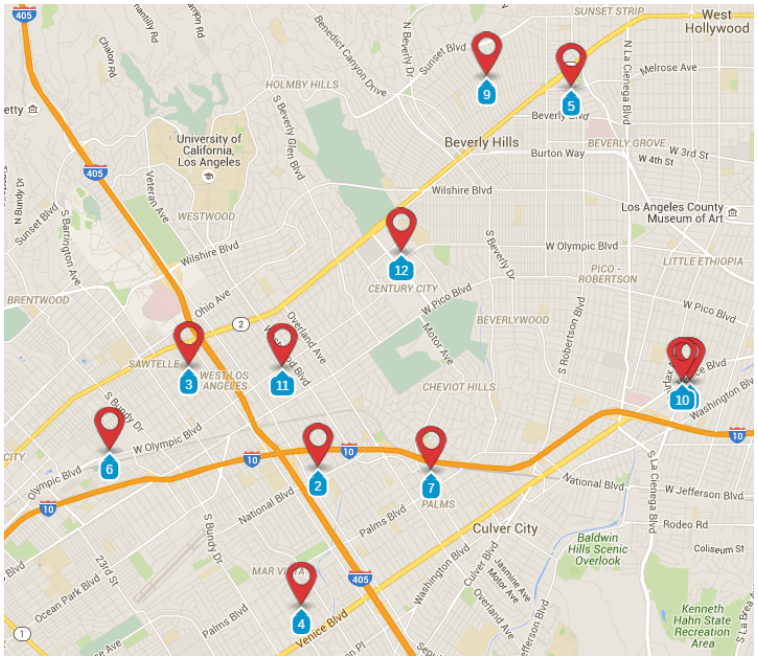


FIGURE 4.2. Location distributions of events. The id labeled to each event is consistent with Table 4.2.

by our system. However, we can detect some events such as firework for celebrating the Independent Day. Local events such as the firework, protesting, smaller earthquake, local festival, or business events, which may be mentioned by some news media, may still not have enough related tweets to be detected by the previous methods. The number of tweets in the clusters related to these events will be discussed in the next section. We also detected many events which only attract the attention from a certain group of people. For these events, only the users who attend or directly related to the event will discuss it. For example, parties, meetings, or events held by some communities like photography lovers. These events may have only tens of related tweets and are very hard to detect by the tweet spike detection based methods.

In Table 4.3, we also show some detected results which are not local events. For some news or non-local events such as national or global festivals, people may talk about them with some uncommon terms. In this way, the tweets containing such terms form very small clusters. Although there are also some big clusters related to these news or events, we may

fail to merge these small clusters to the large clusters because if the smaller clusters only contain a few number of tweets, its time and location distribution may be not that stable. So finally, they are considered as local events by our method. Based on our analysis on labeled results, more than 60 percent of detected NLE_N results belong to this case (the number of tweets in the cluster is less than 100 and there exist larger clusters related to the same event or news). Some other factors can also make the problem more complex such as people in different time zones talking about the same thing or the event happening outside of the area where we collected the data from (e.g. News id 8).

News id	Date	Index Terms	Descriptions
1	May 28	1	lebron 7, lower than jordan, bird magic!!! for king james!!!
2	Jul 26	2	beautiful day! marriage equality today! tomorrow, pass #era @eraeducation! #loveislove @potus @pattyarquette
3	Jul 4	1	happy 4th of july, from the #mets and #dodgers. l'chaim!
4	May 28	1	@ezgimdemirci international day of united nations peacekeepers!
5	May 28	1	why do they kill animals in shows i'm crying
6	Jun 17	2	zombie!! #walkingdead
7	Jun 22	1	@taylorswift13 awesome!!!
8	Jun 25	2	Apparently peru is beating bolivia because my dad keep yellin.
9	Jun 13	1	@jessjjones @ boomer #vegas june 17. hit link to perform.
10	Jun 8	1	world oceans day 8 june.

TABLE 4.3. Selected detected news (NLE_N) in Los Angeles.

4.2.3. Event Type

From the examples above, we can see that it is very difficult to give a clear classification of the local events, especially for the events which only extract local attention. Smaller local events can relate to many things. It is very difficult to find an event ontology which can

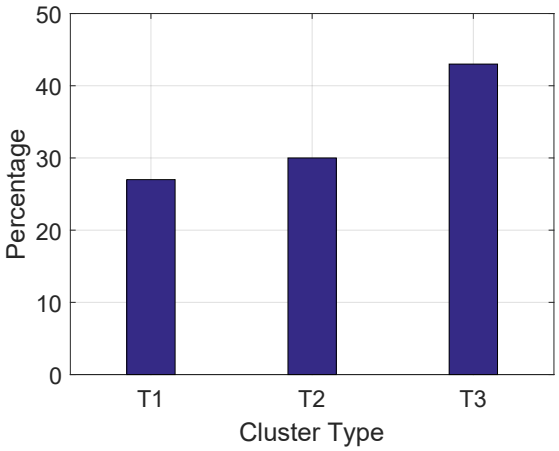
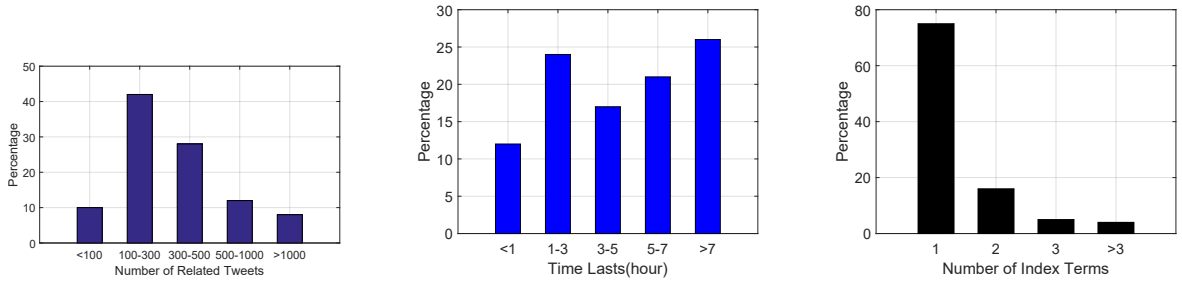


FIGURE 4.3. Proportion of different type of events.

cover all of them. To cover different cases and compare with other methods, we divide the events into three types based on their influence. The first class (T1) is the "large events", which means that the events have a global influence. People in different cities will discuss the same event when it happens, e.g. the NBA finals and E3 2015. As we discussed above, for such events, we may detect many clusters related to them in different cities. Here we only take the cluster in the location where the event happens as the local events related cluster. The second class (T2) is the event which is mentioned by some news medias but does not have a global influence. Users who attend or talk about the event mainly come from the same city, e.g. local festivals or some business events. The last class (T3) are the events that only attract the attention from the users who participate. For example, for personal meetings or parties, only the people who are attending will talk about it. We also manually labeled the types of the event. Each event will be labeled three times and we also follow the majority rule here.

In Figure 4.3, we show the ratios of these three kinds of classes. These three types of are 26%, 31%, and 43% respectively. Previous work based on detecting the burst of the number of tweets can only detect the events in the first type or some of the second type. The events related to sports, shows, or large scale disasters are the majority of the events detected by those methods. They will miss all T3 events which are 43% of all the events

LEDS detected. They will also miss some of the T2 events (31%). These events only attract the attention from a small group of users and cannot be seen on news medias. They are challenging for the previous methods.



(a) Number of Tweets (b) Time (c) Number of Index Terms

FIGURE 4.4. Detailed features of detected events.

4.2.4. More Details of the Detected Events

In Figure 4.4, we provide more detailed information of the detected events. Figure 4(a) shows the distribution of the number of tweets of each event. Here the number of tweets means the counts of tweets in the clusters. Please note that there are also non-related tweets in these clusters because in the first step, we build the original clusters just based on the index terms. There may be some non-related tweets containing the index terms in the clusters. However, the negative impact of these tweets is limited because we only need to make sure that the related tweets are the majority of the cluster. In the last step, when extracting information about the events, we can make sure that the extracted information is highly related to the events. In fact, from our labeling results, there are only 0.75 tweets of the top 5 extracted tweets not related to the event on average. Most detected events have less than 300 tweets and there are only less than 8 percent of events with more than 1,000 related tweets. For T1 events, which have a global influence, there are about 65% of them having more than 1,000 related tweets and for the T3 events, this percentage is 0.

Figure 4(b) shows the distribution of duration of the local events. Here we use the Gaussian distribution to fit the distribution of tweets of events on the timeline. The time

between the $\pm 2\sigma$ will be considered as the duration of the events. There are only 12% of events lasting less than 1 hour and 25% of the events are discussed more than 7 hours. The differences between the three types of events are not as large as the number of tweets. For T1 and T2 events, the percentage of the events lasting more than 7 hours is 54% and for T3 events, this percentage is 23%.

Finally, we provide the distribution of the number of index terms of clusters. Most events come from the clusters with only one index terms, which means that we only detect one abnormal keyword from this event. The percentage of events with more than three index terms is only about 4%. For T2 and T3 events, more than 95 percent of them contain only one index terms. T3 events contain 2.2 index terms on average.

CHAPTER 5

SPATIOTEMPORAL TOPIC ASSOCIATION DETECTION ON TWEETS

In this chapter, we first introduce the algorithm for mining uncertainty co-location. Then we will propose our topic association detection algorithm.

5.1. Mining Uncertainty Co-location

A co-location represents a subset of spatial features whose events frequently appear together in spatial proximity. In Epidemiology, incidents of different but related diseases occur in different places. These diseases may exhibit co-location patterns where some types of diseases tend to occur in spatial proximity. In Ecology, different types of animals can be observed in different locations. There exist patterns such as symbiotic relationship and predator-prey relationship. Different types of crimes committed and different types of road accidents may also exhibit co-location. Many spatial data are uncertain with approximations and errors in the real world. In Epidemiology, the occurrence of a disease may not be geo-located precisely and may be often associated with several locations, e.g. home and workplace. In Ecology, observation of spices is often imprecise. Finding co-locations under uncertainty is useful for these domains.

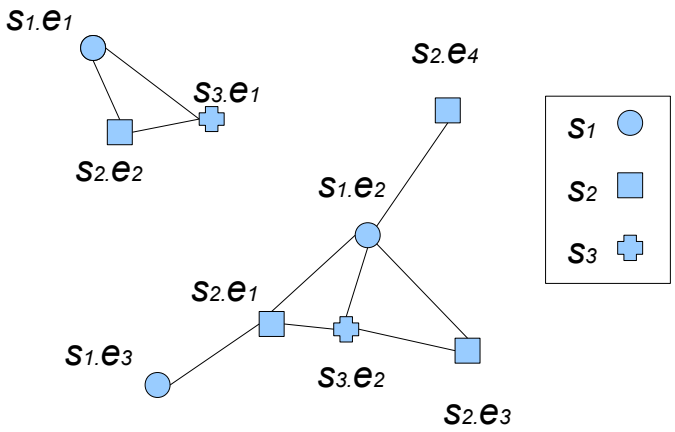


FIGURE 5.1. Example Co-location. Three spatial features: $\{s_1, s_2, s_3\}$. s_1 has three events, s_2 has four events, and s_3 has two events.

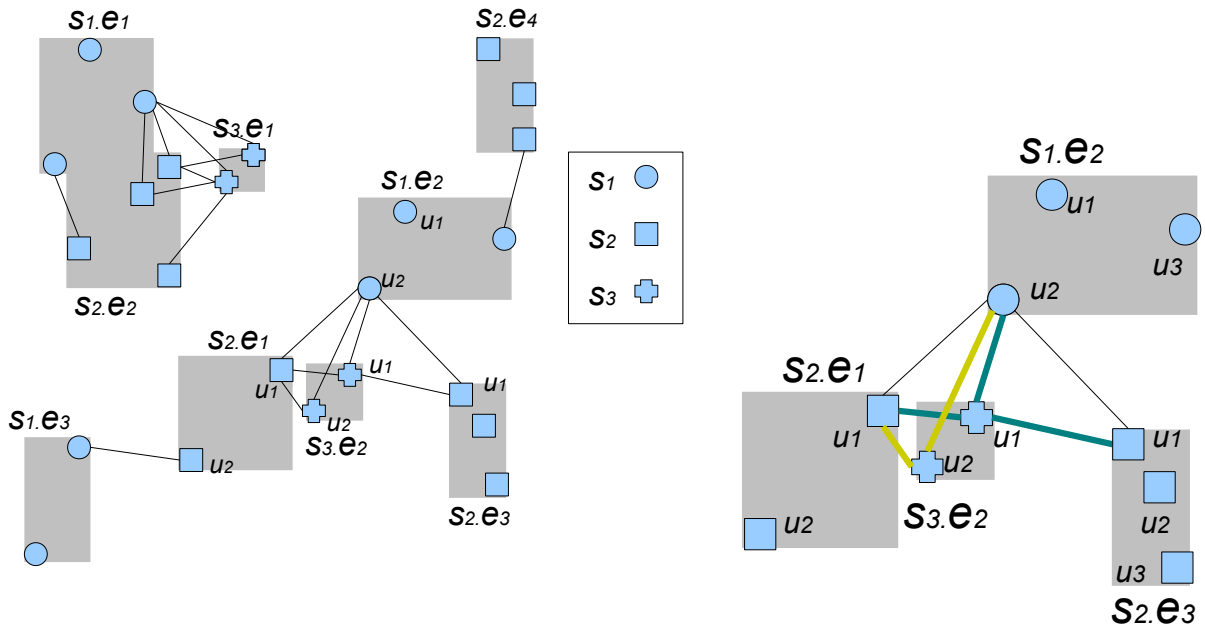
In the problem of mining certain co-locations, a set S of spatial features is given and each spatial feature s is associated with a set of events $s.E$. The spatial feature of a given event e is denoted as $s(e)$. A set of events E is supporting a subset of spatial feature $S' \subseteq S$ if: (1) E forms a clique using a user given distance threshold; (2) for any $e_1 \in E, e_2 \in E$ and $e_1 \neq e_2$, we have $s(e_1) \neq s(e_2)$; (3) $\cup_{e \in E}\{s(e)\} = S'$. The participation ratio $PR(s, S')$ of a spatial feature s in a subset of spatial feature $S' \subseteq S$ is the probability of an event of s participating in a supporting clique of S'. For example, in Figure 5.1, there are three spatial features $S = \{s_1, s_2, s_3\}$. Two events of different spatial features are connected if their distance is less than a user specified threshold. Feature s_2 has four events and three of them participate in a clique supporting $\{s_1, s_2, s_3\}$. So $PR(s_2, \{s_1, s_2, s_3\})$ is $3/4$. Then participation index $PI(S')$ is defined as $PI(S') = \min_{s \in S'}\{PR(s, S')\}$. In Figure 5.1, $PI(s_1, s_2, s_3) = \min\{2/3, 3/4, 2/2\} = 2/3$. The problem of co-location mining is to find all subsets of spatial features with participation indices above a user defined threshold.

When an event is uncertain, there are two main challenges: (1) How to define participation ratio and participation index in a probabilistic manner? (2) How to efficiently find co-locations when the number of uncertain events is large? Let us assume a simple dataset of 5 spatial features. Let each spatial feature have 4 events and each event has 3 possible locations. Then we will be $3^{45} = 3.49 \times 10^9$ possible worlds.

5.1.1. Problem Definition

We follow the commonly used uncertainty model Block-Independent Disjoint Scheme [12] to define the problem of uncertainty co-location. A probabilistic spatial feature s is given by a set of uncertain events $s = \{e_1, e_2, \ldots\}$. An uncertain event e_i is represented by a set of d-dimensional points $u_1, u_2, \ldots$ reflecting all possible instances of e_i. Each instance u_k is assigned with a probability $P(u_k)$ denoting the probability that e_i appears at u_k. Let h be a user given distance threshold. In Figure 2(a), there are three spatial features. Spatial feature s_1 has three uncertain events, spatial feature s_2 has four uncertain events, and spatial feature s_3 has two uncertain events. Each instance of an uncertain event of s_1, s_2, and s_3 is represented by a circle, a square, and a plus sign respectively. Probabilities are assigned

to instances of an uncertain event that sums up to 1 within the event. Two instances of different spatial features that are within a user given threshold h are connected by a line. In this example, s_1, s_2, s_3 may be three animal species, e.g. s_1 = Egyptian plover, s_2 = Nile crocodile, and s_3 = monitor lizard. An event of s_1 represents a particular Egyptian plover, e.g. plover Smith. And the instance of the plover Smith is a location where Smith is spotted.



(a) Example of Uncertainty Co-location

(b) Instance Centric Counting

FIGURE 5.2. In 2(a), feature s_1 has three uncertain events, i.e. $\{s_1.e_1, s_1.e_2, s_1.e_3\}$. Event $s_1.e_1$ has three uncertain instances.

A possible world $w_s = u_1^s, u_2^s, \ldots$ of a spatial feature s is a set of instances containing one instance from each event of s and occurring with a probability of $P(w_s) = \prod_k P(u_k^s)$. Let W_s be the set of all possible worlds for s, then $\sum_{w \in W_s} P(w) = 1$. We are given a set n of probabilistic spatial features $S = \{s_1, s_2, \ldots, s_n\}$, a possible world w_S is given by the combination of a possible world of each spatial feature, i.e. $w_{s_1}, w_{s_2}, \ldots, w_{s_n}$ with a probability of $P(w_S) = \prod_i P(w_{s_i})$. Obviously, let W_S be the set of all possible world for S, then $\sum_{w \in W_S} P(w) = 1$ as well.

Definition[Complete Possible World] Given a subset of probabilistic spatial features S, W_S^c includes all the instances of all the events of S and is called the complete possible world with respect to S. For any instance, we use $e(u)$ to denote the event associated with u and use $s(u)$ to denote the spatial feature of u.

All of the instances in Figure 2(a) is an example of complete possible world of $\{s_1, s_2, s_3\}$.

Definition[Supporting Clique] A set of instances c in the complete world $w_{S'}^c$ is supporting a subset of spatial feature $S' \subseteq S$ if: (1) c forms a clique using a user given distance threshold; (2) $s(u_1) \neq s(u_2)$ (which also implies $e(u_1) \neq e(u_2)$) for any $u_1 \in c, u_2 \in c$ and $u_1 \neq u_2$; (3) $\cup_{u \in U}\{s(u)\} = S'$. c is called a supporting clique of S'.

In Figure 2(a), $\{s_1, s_2, s_3\}$ has 6 supporting cliques. However, not all of them can appear together in the same possible worlds as explained later.

The participation ratio and index are used to prune the co-locations with low prevalence. Different from certain cases, the participation ratio cannot be directly calculated by counting the instances participating in a supporting clique. In an uncertain case, the participation ratio of an instance will be determined by both the probability of possible worlds and the supporting cliques it participates.

Definition[Participation Ratio of an Instance] For a subset S' of S and a user given distance threshold, the participation ratio of an instance u of spatial feature s_i in S' is defined as the sum of probabilities of all the possible worlds that instance $s_i.u$ participates in a supporting clique of S' and this can be calculated as:

$$PR(u, S') = \sum_{\{w | \exists c, u \in c, c \subseteq w, w \in W_{S'}, c \ supports \ S'\}} p(w) \tag{5.1}$$

In Figure 2(b), $PR(s_1.e_2.u_2, \{s_1, s_2, s_3\})$ is the sum of the probabilities of all the possible worlds that contain at least one supporting clique of $\{s_1, s_2, s_3\}$ with $s_1.e_2.u_2$ participating. In this example, it includes all of the worlds that contain one or more of the three supporting cliques that $s_1.e_2.u_2$ participates in the figure. Since $s_1.e_2$, $s_2.e_1$, $s_3.e_2$ and $s_2.e_3$ have 3, 2, 2, 3 possible instances respectively, the total number of possible world should be

36. Among those, 3 possible worlds have a supporting clique of $\{s_1.e_2.u_2, s_2.e_1.u_1, s_3.e_2.u_1\}$, 3 contain that of $\{s_1.e_2.u_2, s_2.e_1.u_1, s_3.e_2.u_2\}$, and 2 contain that of $\{s_1.e_2.u_2, s_2.e_3.u_1, s_3.e_2.u_1\}$. Given equal probability of instances, all of the possible worlds have the same probability of 1/36. However, one possible world contains both of the cliques $\{s_1.e_2.u_2, s_2.e_1.u_1, s_3.e_2.u_1\}$ and $\{s_1.e_2.u_2, s_2.e_3.u_1, s_3.e_2.u_1\}$. So there are totally $3 + 3 + 2 - 1 = 7$ possible worlds that $s_1.e_2.u_2$ participates in a supporting clique of $\{s_1, s_2, s_3\}$. The participation ratio of $s_1.e_2.u_2$ should be $PR(s_1.e_2.u_2) = 7 \times (1/36)$.

Definition[Probabilistic Participation Ratio] Let $s_i.U$ be all possible instances of s_i where $s_i \in S'$ and $|s_i|$ be the number of events of s_i. The probabilistic participation ratio (short as participation ratio or PR here after) of s_i is:

$$PR(s_i, S') = \frac{1}{|s_i|} \sum_{u \in s_i.U} PR(u, S') \tag{5.2}$$

Definition[Probabilistic Participation Index] For a subset $S' \subseteq S$, the probabilistic participation index (short as participation index or PI) $PI(S')$ of S' is defined as follows:

$$PI(S') = \min PR(s_i, S') \tag{5.3}$$

The problem of mining co-location under uncertainty is to find all subsets of spatial features with participation indexes above a user given threshold θ.

5.1.2. Instance Centric Counting

The naive way to calculate the participation ratio of an instance in S' is to enumerate all the possible worlds and then sum up the probabilities of the worlds where the instance participates in a supporting clique of S'. However, this method is very expensive as the number of possible worlds is very large. We propose an instance centric calculation of the participation ratio, taking supporting cliques involved into consideration. However, in this method, avoiding over-counting is a challenge. We prove a Lemma to allow instance centric counting which will enable efficient algorithms to be developed. Using this method, we can get the same result as summing up possible worlds above. We first define the relationship between a possible world and a clique.

Definition[Clique Probability] For a supporting clique c of S', the probability $P(c)$ is the sum of the probability of all of the worlds which contains the clique c. And it can be represented as:

$$P(c, S') = \sum_{\{w|w \in W_{S'}, c \in w\}} P(w) \tag{5.4}$$

Finding all the possible worlds that contain c is very expensive. However, it is easy to prove the lemma 1. The proof is similar of that of lemma 2 and is omitted due to space constraint.

Lemma 1. *For a supporting clique c of S', the probability $P(c, S')$ is equivalent to:*

$$P(c, S') = \prod_{u \in c} P(u) \tag{5.5}$$

If an instance u only participates in one supporting clique, then the participation ratio of u is the probability of the clique. However, an instance can participate in multiple cliques in the same world, resulting in over-counting if we simply sum up the probabilities of all the supporting cliques that u participates (naive PR calculation). For example, Figure 2(b) is a subset of Figure 2(a) which includes all events and cliques that relate to $s_1.e_2.u_2$. The two instances of $s_3.e_2$ cannot happen at the same time (yellow lines and green lines could not happen in the same world). So not all three cliques can happen in the same world. However, clique $\{s_1.e_2.u_2, s_2.e_1.u_1, s_3.e_2.u_1\}$ and $\{s_1.e_2.u_2, s_2.e_3.u_1, s_3.e_2.u_1\}$ can co-exist, resulting in over-counting of the worlds that contain both if we use the naive PR calculation.

Definition[Coexistence] Two supporting cliques c_1 and c_2 of S' can coexist, denoted as $\odot(c_1, c_2) = 1$, in the same possible world under the following condition: $\forall u_1 \in c_1, u_2 \in c_2$, if $e(u_1) = e(u_2)$, then $u_1 = u_2$, where $e(u)$ denotes the event associated with the instance u.

Definition[Coexistence of a Set of Supporting Cliques] A set of supporting cliques C can coexist and is denoted as $\odot(C) = 1$ if every pair of the cliques can coexist.

Definition[Probability of Coexisting Clique Set] The probability of a set of coexisting supporting cliques C of S' is the sum of the probabilities of those possible worlds in which

C happens:

$$P(C, S') = \sum_{\{w|\forall c \in C, c \subseteq w, w \in W_{S'}\}} P(w) \tag{5.6}$$

Lemma 2. *The probability of a set of coexisting supporting cliques C of S' can be calculated as:*

$$P(C, S') = \prod_{u \in I} P(u) \tag{5.7}$$

where I is the union of all instances of the events of the cliques in C.

Proof: Let E be the set of all events. For any $E' \subseteq E$, it is easy to see that the sum of the probabilities of all possible worlds $W_{E'}$ that includes an instance from each event in E' is 1. The probability of coexisting clique set is the sum of all possible worlds which contain the coexisting supporting clique set C. We can divide E into two parts: E_1 is the set of events having an instance in the coexisting cliques and E_2 is the complementary set. So a possible world that contains C can also contain any other possible worlds of E_2. So the summation can be calculated as $\prod_{u \in I} P(u) \times (\sum_{w \in W_{E_2}} P(w))$. We know that $\sum_{w \in W_{E_2}} P(w) = 1$, so the probability of a set of coexisting supporting cliques is $\prod_{u \in I} P(u)$.

From lemma 1 we know that the probability of a clique c_1 is the sum of all of the possible worlds that contain the clique. However, those possible worlds can contain both cliques c_1 and c_2. So the probabilities of the possible worlds which contain both cliques will be counted two times if they can coexist when counting the participation ratio of an instance. To summarize, when there are coexisting cliques and they contribute to the participation of the same instance, over counting will happen. We define such cliques as star supporting cliques.

Definition[Star Supporting Clique Set] All the supporting cliques of S' that an instance u participates is called the star supporting clique set of u in S' and is denoted as $\star(u, S')$.

Lemma 3. *To an instance $s_i.u_k$ in a complete possible world, the star supporting clique set $\star(s_i.u_k, S')$ of S' can be used to calculate the participation ratio of the instance as follows where $l = |\star(s_i.u_k, S')|$:*

$$
\begin{aligned}
PR(s_i.u_k, S') = & \sum_{c_j \in \star(s_i.u_k, S')} P(c_j, S') \\
& - \sum_{c_1 \in \star(s_i.u_k, S'), c_2 \in \star(s_i.u_k, S'), \odot(c_1, c_2)} P(c_1 \cup c_2, S') + \ldots \\
& + (-1)^{l+1} \sum_{c_1 \in \star(s_i.u_k, S'), .. c_l \in \star(s_i.u_k, S'), \odot(c_1, .. c_l)} P(c_1 .. \cup c_l, S')
\end{aligned} \tag{5.8}
$$

Proof: From the definition of participation ratio, $PR(u_k) = \sum_{w_i \in W_{S'}} P(w_i)$, where $W_{S'}$ is the set of possible worlds which instance u_k participates in forming a supporting cliques of S'. So we can prove lemma 3 by checking whether the probability of each possible world in $W_{S'}$ has been counted and only counted once. For each w_i in $W_{S'}$, let us assume that w_i has n coexisting cliques $\{c_1, c_2 ... c_n\}$. From the definition of the probability of coexisting clique set, we know that every time we calculate $P(C, S')$, we count the probability of w_i once. Here C represents any possible combinations of coexisting cliques in w_i. So when we calculate $P(c_1)$ as well as $P(c_1 \cup c_2)$ and any other combinations, we count $P(w_i)$ once. In lemma 3, for w_i, each of $P(c_1) ... P(c_n)$ is added to the probability of w_i once. Each of $P(c_1 \cup c_2) ... P(c_{n-1} \cup c_n)$ subtracts $P(w_i)$ once. So $P(w_i)$ has been counted $C_n^1 - C_n^2 + ... (-1)^{n+1} C_n^n = 1$ times.

Lemma 4. *(Anti-monotone Property) The participation index is anti-monotone with respect to the number of features in the co-location.*

Proof: Let l, l' be two co-locations and $l' \subseteq l$. We use W_l and $W_{l'}$ to represent the set of possible worlds which have supporting cliques of these two co-locations. For each possible world w in W_l, w contains supporting cliques of l. Because $l' \subseteq l$, w will also has supporting cliques of l'. So if $w \in W_l$, then $w_i \in W_{l'}$. Therefore, $W_l \subseteq W_{l'}$. From the definition of participation index, $PI(l) = \sum_{w \in W_l} P(w)$ and $PI(l') = \sum_{w \in W_{l'}} P(w)$, we can conclude that $PI(l') \geq PI(l)$.

5.1.3. Mining Co-location from Uncertain Data

In this section, we describe the framework of mining uncertain co-locations. While participation ratio can be calculated by counting supporting cliques [43] in the certain case, the participation ratio of uncertain co-locations is calculated by the probabilities of instances. Thus we divide the process of mining co-location in two steps, (1) finding uncertainty co-locations and their supporting cliques; (2) calculating the participation ratios. In section 5.1.3, we propose an Apriori-based algorithm to generate supporting cliques under uncertainty. Then we present the feature tree driven method combined with maximal clique method. We propose event-based pruning and clique-feature table searching to reduce the computational cost. In section 5.1.4, we will present the algorithm for calculating participation ratios and then present algebraic analysis of the computational complexity.

In the following part, we use C_k to represent a set of size k cliques, use L_k to represent a set of size k co-locations and use C_M to represent a set of maximal cliques.

Uncertain Apriori (UApriori) Co-location Miner: We first propose an Apriori like algorithm in the instance level and will present event level pruning. We process in the uncertain instance level to generate the supporting cliques for each co-location. Algorithm 5 shows the process of how to generate uncertainty co-location by Apriori algorithm. Here we use an example to illustrate. Figure 3(a) is the Apriori-gen process for the example in figure 2(b). We use plane sweep algorithm to get the neighbor relation of between the instances of S_1, S_2, S_1, S_3 and S_2, S_3 (the first three tables). They are also the supporting cliques C_2 of size-2 co-locations L_2. Assuming all of them have a participation index greater than the threshold, we can generate size-3 co-locations by them. First we generate the supporting cliques for the co-location $\{s_1, s_2, s_3\}$ from C_2 through joins. For example, we use the $size$-2 cliques $\{s_1.e_2.u_2, s_2.e_1.u_1\}$ and $\{s_1.e_2.u_2, s_3.e_2.u_1\}$ in C_2 to generate the $size$-3 supporting clique $\{s_1.e_2.u_2, s_2.e_1.u_1, s_3.e_2.u_1\}$. When generating the new supporting clique, we calculate the distance between $s_2.e_1.u_1$ and $s_3.e_2.u_1$ by their coordinates instead of searching the c_2 table to find the record of clique $\{s_2.e_1.u_1, s_3.e_2.u_1\}$.

Event Level Pruning (UApriori-E): However, in uncertainty, each event of a feature

Algorithm 5: Generating co-locations from searching table

Apply plane sweep algorithm to generate instance neighbor relation C_2

Generate size 2 co-locations L_2 by the neighbor relation

$k = 2$

while $L_k \neq \emptyset$ **do**

$k = k + 1$

Generate supporting cliques C_k from C_{k-1} for co-location L_k

Calculate the participation index of each size k co-location

Generate size k co-location set L_k

end while

Return $L_2 \cup L_3 ... \cup L_{k-1}$

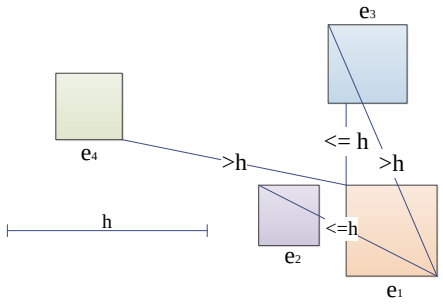
$Instance_1(s_1)$	$Instance_2(s_2)$	$Coordinate(i_2)$
$s_1.e_2.u_2$	$s_2.e_1.u_1$	(x_1, y_1)
$s_1.e_2.u_2$	$s_2.e_3.u_1$	(x_2, y_2)

$Instance_1(s_1)$	$Instance_2(s_3)$	$Coordinate(i_2)$
$s_1.e_2.u_2$	$s_3.e_2.u_1$	(x_3, y_3)
$s_1.e_2.u_2$	$s_3.e_2.u_2$	(x_4, y_4)

$Instance_1(s_2)$	$Instance_2(s_3)$	$Coordinate(i_2)$
$s_2.e_1.u_1$	$s_3.e_2.u_1$	(x_3, y_3)
$s_2.e_1.u_1$	$s_3.e_2.u_2$	(x_4, y_4)
$s_2.e_3.u_1$	$s_3.e_2.u_1$	(x_3, y_3)

$Instance_1(s_1)$	$Instance_2(s_2)$	$Instance_3(s_3)$	$Coordinate(i_3)$
$s_1.e_2.u_2$	$s_2.e_1.u_1$	$s_3.e_2.u_1$	(x_3, y_3)
$s_1.e_2.u_2$	$s_2.e_1.u_1$	$s_3.e_2.u_2$	(x_4, y_4)
$s_1.e_2.u_2$	$s_2.e_3.u_1$	$s_3.e_2.u_1$	(x_3, y_3)

(a) Process of Apriori-gen



(b) Event Level Pruning

FIGURE 5.3. Example for the process of UApriori

may have many uncertain instances. Both algebraic and experiments show that with a large number of instances, the join process to generate cliques become inefficient. To optimize this process, we propose an event level pruning approach to apply event neighbor relation checking first and only proceed to instance level join when the minimal distance between two events are within user given distance.

There are three relations between two events: the minimum distance larger than threshold h, the maximal distance less than h, and $Distance_{Max} > h \cap Distance_{Min} < h$. For the first case, such as e_1 and e_4 in figure 3(b), we can prune e_4 directly. For the second,

such as e_1 and e_2, all instances in e_2 can form neighbor relation with instances in e_1 without calculating the distances between them. We only need to calculate the distance between instances in the third case.

Uncertain Feature Tree Co-location Miner (UFTree) Feature tree has been used in mining long pattern association rules. However, the main challenge in using the same structure in co-location mining is the lack of given set of transactions. In this dissertation, we propose feature tree-based methods in mining uncertain co-locations that deal with the lack of transactions. In the first step, we use the plane sweep algorithm and the Bron-Kerbosch algorithm [15] to find all maximal cliques in the complete possible world (optimization will be discussed shortly) and add them into C_M. A naive way to generate the supporting cliques is to enumerate all sub-cliques of every maximal clique and map them into different co-locations. But without a pruning process, the cost is prohibitive especially when some large size cliques exist. Here we build a searching table to help to generate the supporting cliques and the co-location set. Algorithm 6 describes the framework of this process.

Algorithm 6: Generating co-locations from searching table

$L = \emptyset$
while $S \neq \emptyset$ **do**
  To $s_i \in S$, for each instances of s_i, get maximal cliques contain this instance from C_M
  Build searching table for s_i with other features in S
  Table searching, generating co-location and adding into L
  Remove s_i from S
end while

Figure 5.4 is an example for building searching table. $\{c_1, c_2, ..c_6\}$ is the set of maximal cliques C_M and $\{s_1, s_2, ..s_6\}$ is the set of features S. For s_1 in S, we first get the maximal cliques containing instances of s_1: $\{c_1, c_2, c_3, c_4, c_5\}$. Then those maximal cliques will be mapped into the searching table in figure 5.4. After the co-location generating step, s_1 will be removed from S and will not appear in the searching table built after this. For example, when we build the searching table for s_2, instances from s_1 will not be included in the table.

To search this table and generate co-locations, we introduce the searching strategy on a tree-based structure. The breadth-first searching with follow set pruning approach.

Maximum Cliques:

c_1: $\{s_1.u_1, s_2.u_1, s_3.u_1, s_5.u_1\}$
c_2: $\{s_1.u_1, s_2.u_2, s_4.u_1\}$
c_3: $\{s_1.u_2, s_3.u_1, s_4.u_2\}$
c_4: $\{s_1.u_3, s_2.u_3, s_4.u_2\}$
c_5: $\{s_1.u_4, s_3.u_2, s_5.u_2\}$
c_6: $\{s_2.u_1, s_3.u_2, s_4.u_3, s_6.u_1\}$

Searching Table

	c_1	c_2	c_3	c_4	c_5
s_1	$s_1.u_1$	$s_1.u_1$	$s_1.u_2$	$s_1.u_3$	$s_1.u_4$
s_2	$s_2.u_1$	$s_2.u_2$		$s_2.u_3$	
s_3	$s_3.u_1$		$s_3.u_1$		$s_3.u_2$
s_4		$s_4.u_1$	$s_4.u_2$	$s_4.u_2$	
s_5	$s_5.u_1$				$s_5.u_2$

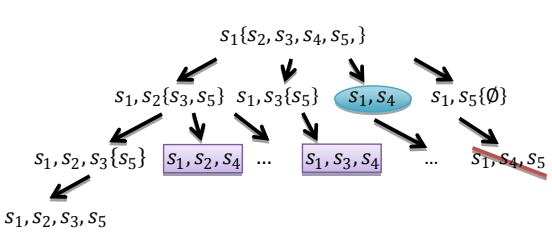
FIGURE 5.4. Searching Table

Uncertainty Feature Tree Searching with Follow Set Pruning (UFTree-FP algorithm): We introduce the following set searching strategy to generate co-locations on the table. We first present key definitions for this strategy.

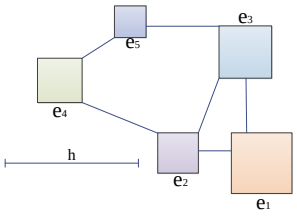
Definition[Pre-Node Set] Node $\mathcal{N}$ is a node in the feature tree. The pre-node set $Pre\{\mathcal{N}\}$ of $\mathcal{N}$ is the set of nodes which have the same father node as $\mathcal{N}$ and at the left side of $\mathcal{N}$ in the feature tree.

Definition[Follow set] The following set of node $\mathcal{N}$ is a set of features which may form co-locations with the features in node $\mathcal{N}$.

For example, for the root node of the tree in figure 5(a), the set of $\{s_2, s_3, s_4, s_5\}$ is the following set of this root node. Feature s_1 will be combined with each of them to generate new co-locations in the next level of this tree.



(a) Searching tree



(b) Finding maximal cliques at event level

FIGURE 5.5. Strategie on searching the table

Algorithm 7 describes the process of breadth-first search to generate co-locations related to s_i. When searching the table, we will generate the follow set for each node. First, we set the following set of the root node as $S_{s_i} - s_i$ and add the root node to the node list. Then we process each node in the node list. For every node, we combine the feature set of the node with each feature S_{Fi} in the following set and get a new co-location. If the participation index of the new co-location is greater than the threshold, this node will be added to the node list and the feature S_{Fi} will be added into the following sets of every pre-node of this node.

Algorithm 7: The breadth-first searching algorithm

```
Set co-location set L = ∅
Node list NL = ∅
Add root node to NL
Set the follow set of root node as S_{s_i} − s_i
i = 0
while i ≤ NL.size do
    N = NL[i]
    Combine the feature set of N with each feature s_{Fi} in the follow set and generate a new co-location l_n
    Calculate the participation index PI
    if PI ≥ threshold then
        Add N' to the node list, the feature set of N' composed by the features in l_n and the follow set is node N's
        follow set besides s_{Fi}
        Add s_{Fi} to the follow sets of each pre-node of N'
        L = L ∪ l_n
    end if
end while
Return L
```

Figure 5.5 (b) is an example for breadth-first search. During the search for the node of $\{s_1, s_3\}$, if the participation ratio is greater than the threshold, s_3 will be added into the follow set of node $\{s_1, s_2\}$. The participation ratio of $\{s_1, s_4\}$ is less than the threshold, so the sub-tree of this node $\{s_1, s_4, s_5\}$ will be pruned and s_4 will not be added into the follow sets of the pre-nodes of $\{s_1, s_4\}$. So the node of $\{s_1, s_2, s_4\}$ and $\{s_1, s_3, s_4\}$ will also be pruned.

The process of finding maximal cliques can be computational expensive with a high

density of instances. To avoid this, we generated maximal cliques in the event level at first and used this as an index to generate instance level cliques used in the searching table. For example, in the figure 5(b), there are four maximal event cliques: $\{e_1, e_2, e_3\}$, $\{e_2, e_4\}$, $\{e_3, e_5\}$ and $\{e_4, e_5\}$.

5.1.4. Calculating the Participation Index

We have proposed lemma 3 to calculate the participation ratio of each possible instance. If there was no coexisting cliques, lemma 3 can be represented as: $PR(s_i.u_k, S') = \sum_{c_j \in \star(s_i.u_k, S')} P(c_j, S')$. So we can calculate all the participation ratios of possible instances by calculating the probability of each supporting clique and add the probability to the ratio of the instances belonging to this clique. If c_i and c_j are two coexisting cliques and $\{u_1, u_2 ... u_n\}$ are their common possible instances, to these instances, their participation ratios $PR(u)$ will be deducted by $P(c_i \cup c_j)$ according to the lemma 3. So we need to calculate the probabilities of different combinations of coexisting cliques and add or subtract the results to the ratios of instances shared by those cliques according to lemma 3. After getting all of the participation ratios, we can calculate the participation indexes of co-locations by the method described in the definition of probabilistic participation ratio and participation index.

Table 5.1 summarizes the parameters used in complexity analysis as well as in the experiment section. In the UApriori algorithm, when generating $size(k+1)$ co-locations from $size\ k$, the worst case is: for each supporting clique, we have to see if they have $k-1$ instances in common with other cliques. If we define $N_{C(k)}$ as the number of supporting cliques of size k co-locations, the cost of generating supporting cliques is: $O(\sum_{k=2}^{w-1}(k-1) \mid N_{C(k)} \mid)$. For the uncertain feature tree algorithm, the cost of generating step can be divided into three parts: finding out all maximal cliques, building a searching table, and generating supporting cliques. Finding maximal cliques is a NP problem [15]. Thus we propose to do this step on event level in our approach to avoid high computational cost. For each feature, we use N_{C_L} as the average number of supporting cliques of each co-location and N_{F_i} as the number of cliques related to each feature. So the cost of building the searching table is $\mid F \mid N_{F_i} N_{C_L}$. To generate supporting cliques for each co-location set, we need to search a

line of the searching table and the worst case is $O(N_{F_i})$. So the cost of UFTree algorithm is $O(\mid F \mid N_{F_i} N_{C_L} + N_{F_i} \sum_{k=2}^{w} \mid C(k) \mid)$.

N_s	Number of Features
O_L	The overlapping ratio of co-locations
$\overline{I}$	Average number of instances of an event
N_L	Number of co-locations
$\overline{L}$	Average length of co-locations
$\overline{E}$	Average number of events of each feature
$\overline{C}$	Average number of supporting cliques of a co-location in event level

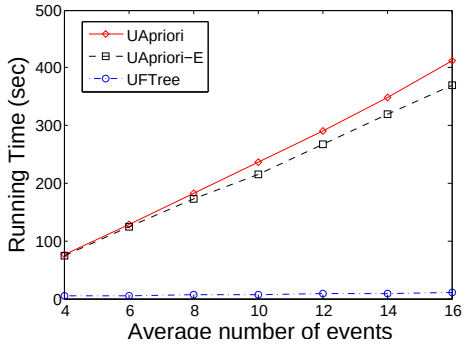
TABLE 5.1. Parameters used in generating synthetic data

5.1.5. Experimental Results

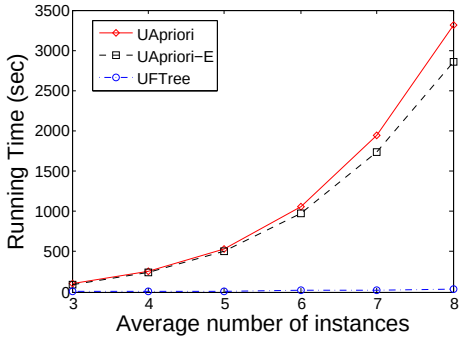
In this section, we will first compare the algorithms on different synthetic datasets to show the influence of parameters. Since there were no algorithms solving the same problem with us, here we only compared our own method and analysis the results. Then we will use the Shanghai taxi trip dataset to illustrate the application of our algorithms in real world.

Synthetic Data Generation: Our model of the synthetic data is similar of that of paper [43]. Parameters in table 5.1 are used to generate synthetic data. We generated uncertainty datasets with a broad range of values for the chosen parameters to evaluate the performance of those algorithms. The length of co-location, the average number of supporting cliques, and the average number of instances in an event will be picked from three Poisson distributions $P(\overline{L})$, $P(\overline{C})$ and $P(\overline{I})$. The generating step will be divided into three steps: 1. Generating a co-location; 2. Generating event level supporting cliques for this co-location; 3. Transforming each event into possible instances using a Poisson distribution in an area of given size. We generated the datasets by setting different values for $\overline{L}$ (from 2 to 11), $\overline{E}$ (from 3 to 16), $\overline{I}$ (from 2 to 8), and setting 5 values for $O_L : 0, 20\%, 40\%, 60\%, 80\%$.

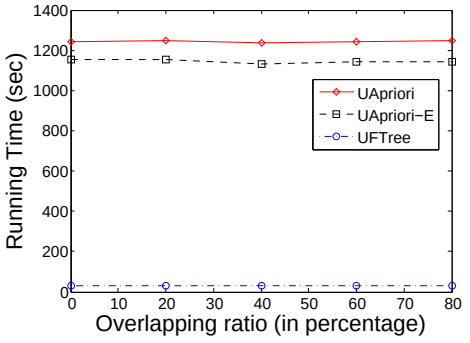
Figure 5.6 presents the running time of the algorithms under different values of those



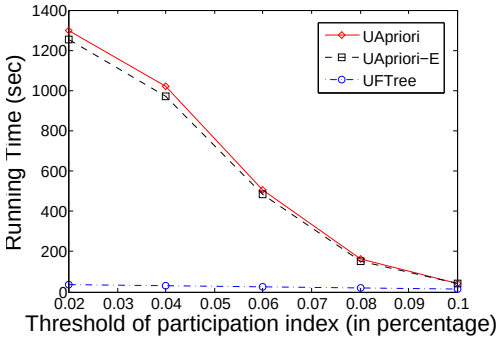
(a) $N_L = 10k, \overline{L} = 4, \overline{E} : 4 \sim 16, \overline{I} = 4$



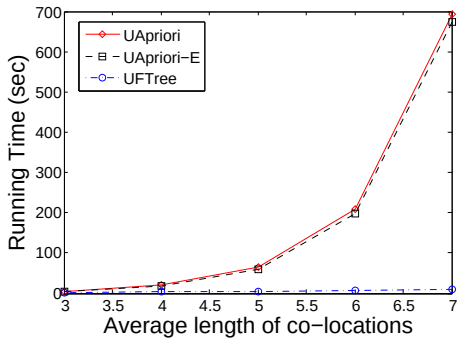
(b) $N_L = 20k, \overline{L} = 4, \overline{E} = 5, \overline{I} : 3 \sim 8$



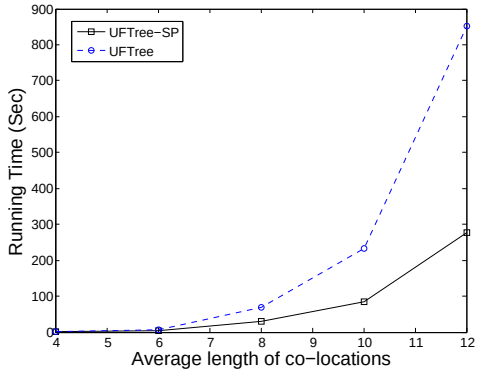
(c) $N_L = 20k, \overline{L} = 5, \overline{E} = 3, \overline{I} = 3$, overlapping ratio(%) : $0 \sim 80$



(d) $N_L = 20k, \overline{L} = 6, \overline{E} = 3, \overline{I} = 3$,PI threshold(%) : $0.02 \sim 0.1$



(e) $N_L = 20k, \overline{L} : 3 \sim 7, \overline{E} = 3, \overline{I} = 2$



(f) $N_L = 20k, \overline{L} : 4 \sim 12, \overline{E} = 3, \overline{I} = 3$

FIGURE 5.6. Experimental results on synthetic data.

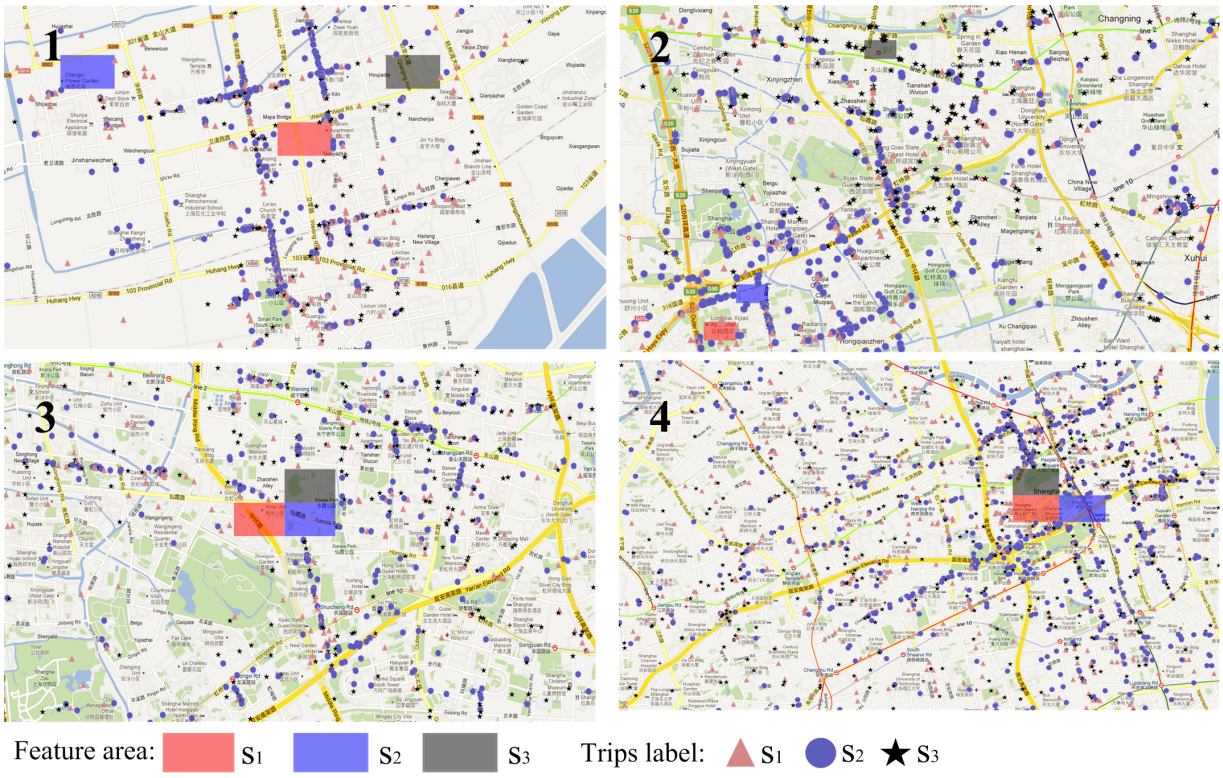


FIGURE 5.7. Four example co-locations from top 10 co-locations on the taxi trip dataset. Rectangle areas represent the spatial features. Their trip ends are represented by a triangle, circle and pentagram respectively.

parameters. We first compare those algorithms by changing the value of $\overline{E}$ and $\overline{I}$. The results are shown in figure 6(a) and 6(b). Figure 6(a) shows the influence of $\overline{E}$. $\overline{E}$ has little influence on the UFTree algorithm. However, with the increasing value of $\overline{E}$, the advantage of event pruning process becomes significant between UApriori and UApriori-E algorithms. In figure 6(b), we can see that with the increase of the average number of instance, the running times of both UApriori-E and UApriori algorithm increase quickly. But the UFTree algorithm still uses a very short time. This phenomenon illustrates that generating cliques in the searching table performs much better than the Apriori-gen process. From figure 6(c), we can conclude that the overlapping ratio has litter influence on those algorithms. In figure 6(d), we tested the influence of the threshold of participation ratio on running time.

The UFTree algorithm performs much better due to its fast matching process and

searching strategy. To illustrate this phenomenon, we can simplify the expression of the computational complexity and just concentrate on the generating process to analyze the computational cost. For UApriori, UApriori-E and UFTree algorithms, the worst case of generating the supporting cliques can be represented by $O((k-1)+(\overline{E}\times\overline{I})^2)$, $O((k-1)+\overline{E}^2+\overline{E}\times(\overline{I})^2)$ and $O((\overline{E}\times\overline{I})^2)$ in generating the supporting cliques of a size k co-locations. But with different values of parameters, the costs of UApriori and UFTree algorithm are typically far less than that. For example, if event from one feature can only form one event pair with another event and there is only one instance pair between them, the cost of UApriori and UFTree will become $O((k-1)+\overline{E}^2+\overline{E}\times\overline{I}^2)$ and $O(\overline{E})$. So with the increase of $\overline{L}$, the UApriori Algorithm becomes much slower and the UFTree algorithms provide a much better performance. We can see that in the results of figure 6(e).

Tree-based algorithm has been used in long pattern association rule mining. A significant advantage of this algorithm is its subset pruning strategy. For example, when we get $\{S_1, S_2, S_3\}$, we need not to test $\{S_2, S_3\}$ due to the anti-monotone property. In our experiment mentioned above (figure 6(a) to 6(e)), we didn't use this strategy. In figure 6(f), we compared the UFTree algorithm with and without the subset pruning: UFTree-Subset Pruning and UFTree algorithm. The results show that subset pruning can improve the performance especially in mining co-locations of large sizes. The reason is larger sized co-locations contain more subsets (equal to the number of power set). From the results, we can see that when the size of co-locations is larger than 8, the UTFree-Subset Pruning method made a significant improvement.

Data for the framework is based on trips of 17,000 Shanghai taxis for one day (May 29, 2009); the dataset contains 432,327 trips. Each trip includes the starting and destination coordinates and the start/end time. Our real data experiment is based on the premise that if people from two or more areas often take taxi to some other common areas, these two or more areas may have some potential characteristics in common. For example, they may be all university areas or residential areas. Of course, people from adjacent areas also tend to have the same destinations because adjacent areas tend to have similar characteristics. But

the taxi trips start and end coordinates can be different even when they have the same start and destination areas. For example, a university may have several entrances and people can get on or off taxis at any entrance. So it becomes an uncertainty co-location problem. The university area is an event and each entrance can be a possible instance in our uncertainty co-location model.

We generate the dataset according to our uncertainty model from the real world taxi trips. Firstly, we divided the region of Shanghai into $500 \times 300m$ rectangles. Top 680 areas with more than 300 trip starting points are chosen as feature areas. Each area is a spatial feature. For each feature s, an area e with a trip starting in s and ending in e is an event of s. The trip ending points in an event e of s is then clustered into small groups, each representing an instance with probability as the ratio of the number of trip ending points in the small group over that in e.

We ran the mining algorithms on the taxi trips dataset with instance distance threshold as 100 meters. Figure 5.7 shows 4 example results from the top 10 co-locations. Table 5.2 describes the feature areas of the co-locations in figure 5.7 and gives the possible reasons for the co-locations. Feature areas in example 1 and 2 have the same characteristics (residences, school zones). Example 3 and 4 show three adjacent features. Especially in example 4, People's Square and People's Park are famous attractions in Shanghai. People often visit them together.

5.2. Spatiotemporal Topic Association Detection on Tweets

detection of topic associations has several challenges. First, we need an appropriate measurement to evaluate the closeness between two or more topics. The second challenge is that each topic may have a large number of related tweets. We need algorithms scalable to a large number of tweets and topics. More importantly, the closeness of topic associations can be different in different time and region. So we must take time-region frame into consideration to provide a more accurate measurement, which can make the calculation more complex and expensive. Finally, the detection may result in a large number of topic associations. So it is necessary to filter out the less interesting topic associations.

Co-location Samples with Top Rank Participation Index
1. {Chengxi Flower Garden }, { Shanxin Apartment}, {Zongtongwan Garden}
Explanation: All of those three areas are large and up-scale neighborhoods with many community facilities and amenities.
2. {Shanghai Longbai Middle School, Longbai Rainforest Preschool}, {Longbai No.1 Elementary School, Xijiao Apartment}, {Shanghai Tianshan Middle School}
Explanation: These three areas largely overlap with a school zone.
3. {Hongxian Unit, Xinle Unit}, {Hongxian Unit}, {Xianxia Villa}
Explanation: Three adjacent residential areas.
4. {People's Square, Shanghai Grand Theatre},{People's Square}, {People's Park}
Explanation: Three famous adjacent entertainment and tourism areas in Shanghai.

TABLE 5.2. Examples of real data co-location

First we define the closeness among topics by examining the co-occurrence of tweets containing these topics. Closely related topics may tend to be talked in the similar time and region. Based on this idea, we define the participation index of topic associations. If the tweets of a set of topics tend to be found in the same time and region, they will have a higher participation index. We evaluate this concept in our experiments by comparing the measuring results of participation index with the human judgment. Based on the concept of participation index, we propose three different types of queries to help users to extract topic associations with different semantics in different time-region frames.

One of the challenges is that the number of tweets can be very large as well as the topics and their combinations. For example, in the sample we collected, there are 364,333 geo-tagged tweets that contain the hashtag $\#job$ in the U.S. in July 2015. So how to develop

an efficient algorithm to help to calculate the participation index of a set of topics will be the most important step. Here we propose a multi-layer geographic index and a time index to help us to filter the topic sets under the threshold quickly to calculate the participation index.

Another challenge in our work is the number of topics. Our dataset contains 492,492 hashtags. Any subset of these hashtags can be a potential topic association. To deal with the exponential nature of the hashtag associations, we propose two methods to optimize the mining algorithm: topic filtering and topic combination. In the topic filtering, we will remove some topics which are not affected by other topics or real-world events. These topics participate in a large number of associations and the resulting associations are not interesting. Furthermore, we propose an algorithm to combine similar topics to further reduce the complexity.

This work makes the following contributions:

- We define the participation index of a set of topics by measuring the time and location distributions of tweets containing the topics; We define three types of queries to answer questions related to topic associations;
- By introducing the time and location indexes, we propose an efficient algorithm to calculate the participation index;
- Topic filtering and topic combination methods are developed to optimize the querying results;
- We test our algorithm on a large Twitter dataset which contains 27,956,257 tweets and 492,492 hashtags. The results demonstrate that our method is effective and efficient.

5.2.1. Problem Definition

In this section, we introduce the definition of participation index of topic association and three kinds of topic association queries. Table 5.3 gives the notations used in this work.

A tweet v_i is represented by a triple $(t_i, l_i, \mathcal{H}_i)$, where t_i is the time when it is shared, l_i is the location where it is shared, and $\mathcal{H}_i$ is a set of hashtags contained in the tweet. For

v_i	Tweet i
$t_i, l_i, \mathcal{H}_i$	The time, region and hashtags of tweet v_i
w, r	Time and region windows
W, R	Set of time and region windows
$h, \mathcal{H}$	Topic h and a set of topics $\mathcal{H}$
t, d, θ, k	Thresholds in query

TABLE 5.3. Notations and parameters

example, in the Figure 1.2, the triple for the first tweet v_1:$(t_1, l_1, \mathcal{H}_1)$ has the time information *September* 24, location information l_1, and the topics list $\{\#amazon\}$.

Definition[Topic] The topics of each tweet are defined as the hashtags it contains. One tweet can relate to several topics.

Let w and r denote time and region windows, and we define the pair of $\langle w, r\rangle$ as a *time-region frame*. Each frame corresponds to a set of tweets $V_{\langle w,r\rangle} = \{v_i | t_i \in w \wedge l_i \in r\}$.

Definition[Related Tweets] For each topic, its related tweets are the tweets which contain this topic. So the set of related tweets of topic h in a time-region frame $\langle w, r\rangle$ can be denoted as: $V_{\langle w,r\rangle}(h) = \{v_i | h \in \mathcal{H}_i \wedge t_i \in w \wedge l_i \in r\}$.

Definition[Topic Set] A topic set is a set of hashtags and denoted as: $\mathcal{H} = \{h_1, h_2, \ldots\}$.

5.2.2. Calculating Participation Index

Definition[Neighbor Relationship] Two tweets, v_i and v_j, are neighbors when: 1) $|t_i - t_j| \leqslant \varphi$ and $|l_i - l_j| \leqslant \delta$ where φ and δ are two user given thresholds on time and distance.

Definition[Supporting Tweet] A set $V = \{v_1, v_2, \ldots, v_k\}$ of k tweets supports a *size-k* topic set $\mathcal{H} = \{h_1, h_2, \ldots, h_k\}$ if tweets in V are pairwise neighbors and $h_i \in \mathcal{H}_i$ for $i = 1, 2, \ldots, k$. Each tweet v_i in the set V is called supporting $\mathcal{H}$ with h_i and the predicate $s(v_i, h_i, \mathcal{H})$ is true.

Definition[Participation Ratio] The participation ratio of topic h in the topic set $\mathcal{H}$

in the time-region frame $\langle w, r\rangle$ is defined as:

$$pr_{\langle w,r\rangle}(h, \mathcal{H}) = \frac{|\{v_i | s(v_i, h, \mathcal{H}) \wedge v_i \in V\langle w, r\rangle\}|}{|V_{\langle w,r\rangle}(h)|} \tag{5.9}$$

Definition[Participation Index] The participation index of topic set $\mathcal{H}$ in time-region frame $\langle w, r\rangle$ is defined as:

$$pi_{\langle w,r\rangle}(\mathcal{H}) = min_{h\in\mathcal{H}} pr_{\langle w,r\rangle}(h, \mathcal{H}) \tag{5.10}$$

Definition[Topic Association] A topic set $\mathcal{H}$ is a topic association if its participation index meets a given threshold, and such topic set will be denoted as $\mathcal{H}_c$.

5.2.3. Definitions of Queries

Each query $q(W_q, R_q, \varphi_q, \delta_q, \theta, k)$ contains six constraints: 1) A set of time windows $W_q = \{w_1, w_2, \ldots\}$; 2) A set of region windows $R_q = \{r_1, r_2, \ldots\}$; 3) Time threshold φ_q; 4) Distance threshold δ_q and 5) A user given participation index threshold θ; 6) a rank threshold k. Based on the return of queries, we define two kinds of queries: vertical query and horizontal query. Then we replace k with a topic set to define super topic association query which allows a user to find other topics associated with a topic set that he is interested in.

Vertical Query: A vertical query $q_v(W_q, R_q, \varphi_q, \delta_q, \theta, k)$ includes five constraints and a ranking threshold k. The Cartesian product of W_q and R_q defines a set of time-region frames, which are all the possible combinations of time and region windows, $\{\langle w, r\rangle | w \in W_q \wedge r \in R_q\}$. Each frame corresponds to a set of tweets $V_{\langle w,r\rangle}$. For each time-region frame $\langle w, r\rangle$, we will calculate the participation index of topic sets $\mathcal{H}$ using related tweets in $V\langle w, r\rangle$, and query q_v will return the sets of topic associations with the participation indexes larger than θ and ranking in the top-k among all the topic sets.

Horizontal Query: Similarly, the horizontal query $q_h(W_q, R_q, \varphi_q, \delta_q, \theta, k)$ also has a ranking threshold k. For each topic set $\mathcal{H}$, we calculate its participation indexes $pi_{\langle w,r\rangle}(\mathcal{H})$ in different time-region frames $\langle w, r\rangle$ and then return the time-region frames with top-k

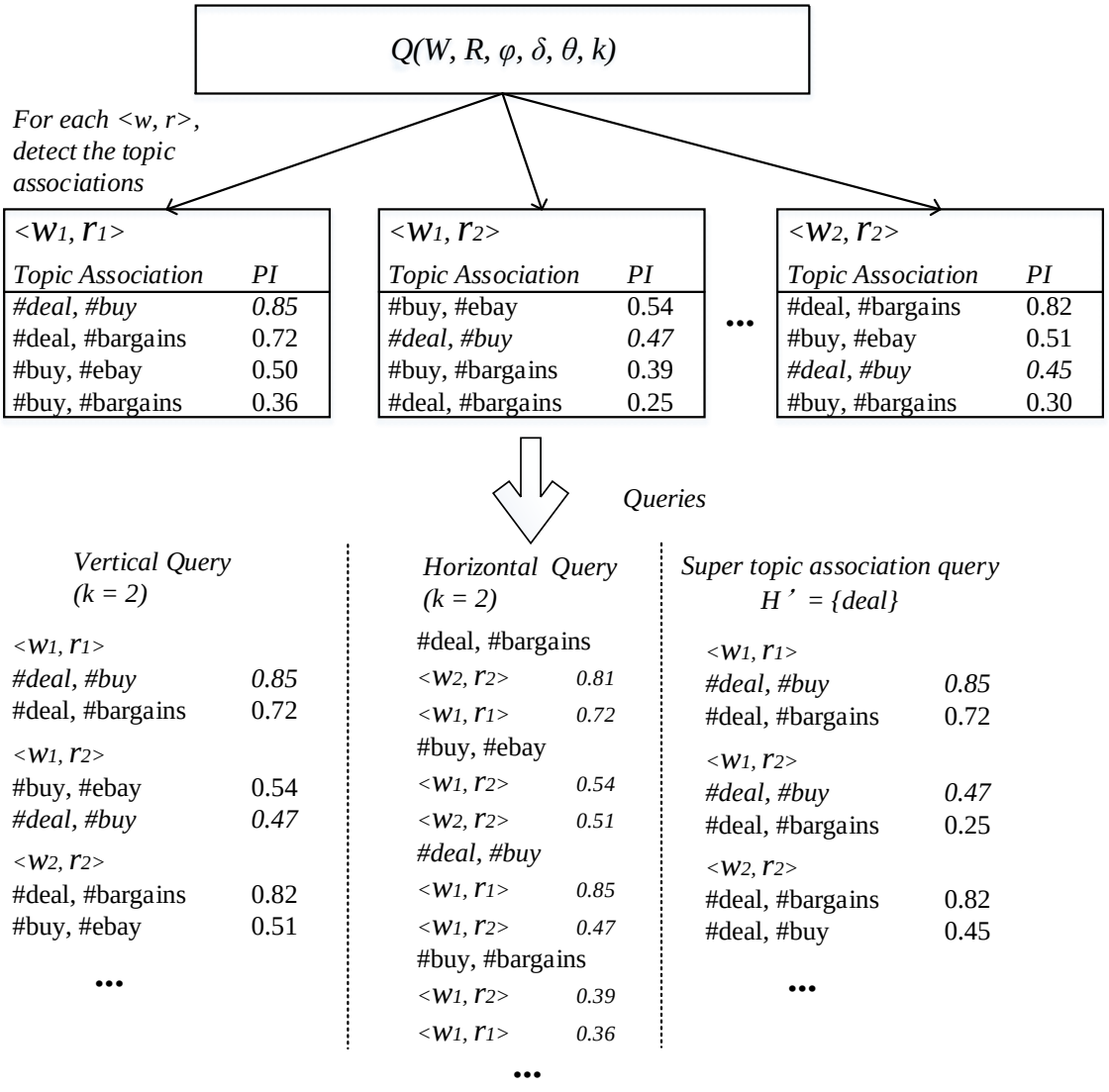


FIGURE 5.8. Vertical query, horizontal query, and super topic association query

highest values of $pi_{\langle w,r\rangle}(\mathcal{H})$. The number of returned frames may be less than k if there are less than k frames having participation index higher than the given threshold θ.

Super Topic Association Query: The super topic association query, $q_h(W_q, R_q, \varphi_q, \delta_q, \theta, \mathcal{H}')$, replaces k with a set of topics/hashtags $\mathcal{H}'$. For each time-region frames $\langle w, r\rangle$ in $W_q \times R_q$, it returns topic associations $\mathcal{H}$ where $\mathcal{H}' \subsetneq \mathcal{H}$ and $pi_{\langle w,r\rangle}(\mathcal{H})$ larger than θ.

For example, Figure 5.8 shows the results of the vertical query and horizontal query. W contains two time windows w_1 and w_2, and R contains two regions r_1 and r_2. Assuming

$\theta = 0.25$, there are several hashtags and they can form topic associations in different time-regions frames. The participation index of each topic association is shown in the figure. For the vertical query, assuming $k = 2$, it will return the top-2 topic associations in each frame. For example in $\langle w_1, r_1 \rangle$, it returns $\{\#deal, \#buy\}$ and $\{\#deal, \#bargains\}$ since their participation indexes are larger than the threshold and ranked in the top-2. For the horizontal query, it returns the top related time-region frames. For instance, the topic association $\{\#deal, \#bargians\}$ achieves highest participation index in $\langle w_2, r_2 \rangle$ and $\langle w_1, r_1 \rangle$. So when $k = 2$, it will return these two time-region frames for this topic association. For the super topic association query, assuming that the set of hashtags inputted is $\{\#deal\}$, it will return the associations in different time-region frames which contain $\#deal$.

5.2.4. Mining Topic Association

Given the definition of the participation index, we now propose algorithms for the queries of mining topic association.

Calculation of the Participation Index: When considering the neighbor relationship between tweets v_i and v_j, we redefine it as $|t_i - t_j| \leq \varphi_q$ and v_i and v_j are published in the same city because real-world events are greatly affected by the geographic boundary. People may be interested in completely different topics in different cities even the distance between the two cities is short. Another reason is that sometimes we do not have an accurate location of the tweet. We may only have the city level location from the tweets or from the profiles of authors. Finally, users may have a city level active area and they may publish tweets anywhere in the city they live.

From the definition of the participation ratio and index, we can see the most important step in detecting topic association is counting the number of supporting tweets of each topic h in $\mathcal{H}$ and calculating the participation index. The Algorithm 8 gives the process of this step.

Previous work has developed R-tree based [100] or grid-based [85] index for spatial related search engines. In this problem, the tweets $V_{\langle w,r \rangle}(h)$ of each topic h are distributed in a three dimensions space, latitude, longitude, and time. Based on our definition of neigh-

Algorithm 8: Calculation of Participation Index

Data: Topic geographic index $\mathcal{I}_T$ and Time index $\mathcal{I}_c$, set of topics $\mathcal{H}$, time-region frame $\langle w, r \rangle$

Result: Participation index of hashtag set $\mathcal{H}$ in $\langle w, r \rangle$

Begin:

$Supporting[h] = 0$;

Find cities containing topics in $\mathcal{H}$ by index $\mathcal{I}_T$, limited by region window r:

$R(\mathcal{H}) = B(h_1) \cap B(h_2) \cap \ldots, h \in \mathcal{H}$;

for *Each* c *in* $R(\mathcal{H})$ **do**

Get the common effective time window by time index $\mathcal{I}_c$, limited by time window w: $W_{common}(\mathcal{H}) = W_{\langle w,c \rangle}(h_1) \cap W_{\langle w,c \rangle}(h_2) \cap \ldots, h \in \mathcal{H}$;

for $h \in \mathcal{H}$ **do**

Count the number n of tweets of topic h in $W_{common}(\mathcal{H})$ at c by the time index $\mathcal{I}_c$;

$Supporting[h] = Supporting[h] + n$;

end

end

$pr_{\langle w,r \rangle}(h, \mathcal{H}) = \frac{Supporting[h]}{|V_{\langle w,r \rangle}(h)|}$ (Equation 5.9) ;

$pi_{\langle w,r \rangle}(\mathcal{H}) = min_{h \in \mathcal{H}} pr_{\langle w,r \rangle}(h, \mathcal{H})$ (Equation 5.10) ;

return $pi_{\langle w,r \rangle}(\mathcal{H})$

bor relationship, we use the bitmap index as the Topic Geographic Index $\mathcal{I}_T$ as shown in Figure 5.9. For each topic h, $\mathcal{I}_T$ contains the index $\mathcal{I}_T(h)$ of h which records the geographic distribution of tweets among cities related to topic h. If we divide the region window r into city level as: $r = \{c_1, c_2, \ldots\}$, where c represents a city level region window, we can denote $V_{\langle w,r \rangle}(h)$ as $\{V_{\langle w,c_1 \rangle}(h) \cup V_{\langle w,c_2 \rangle}(h) \cup \ldots\}$. Each time index in a city c corresponds to a set of tweets $V_{\langle w,c \rangle}(h)$ if it is not empty (topic h has at least one tweet in region c during the time windows in w), and we denote the set of these cities c as $B(h)$.

Then we apply an Time index $\mathcal{I}_c(h)$ on the time dimension on each set of tweets

$V_{\langle w,c\rangle}(h)$ as it is shown in Figure 5.9. Every tweet v_i of topic h at city c has an " effective period" $[t_i - \varphi_q, t_i + \varphi_q]$ defined by the user given time threshold φ_q in the query. The union of the effective periods of tweets in $V_{\langle w,c\rangle}(h)$ is the total effective periods of topic h in the city c and denoted as $W_{\langle w,c\rangle}(h)$. So based on the definition of neighbor relationship, for a tweet v_j of another topic, if v_j was published at city c and during the period of $W_{\langle W_q,c\rangle}(h)$, we can find at least one tweet v_i of topic h which has neighbor relation with v_j.

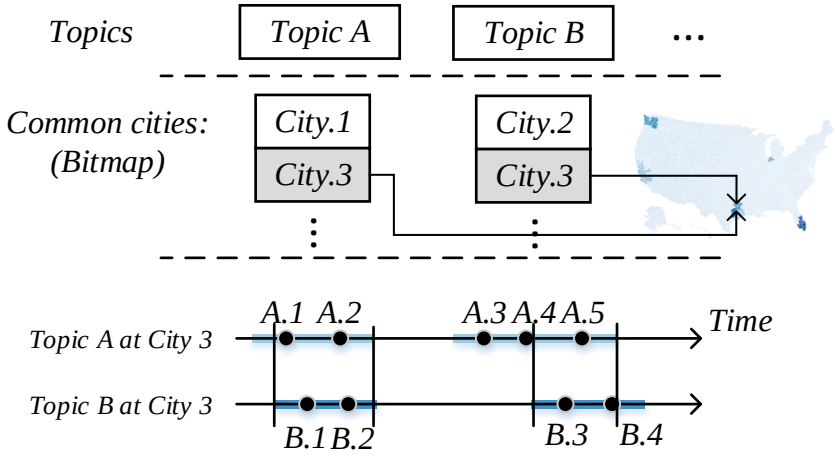


FIGURE 5.9. Calculation of participation index of topic sets $\mathcal{H}$: There are two topics in this example, $\#Topic\ A$ and B. $\#Topic\ A$ has related tweets distributed in two cities. We use the bitmap index $\mathcal{I}_T$ to record cities which has tweets related to $\#Topic\ A$. For the tweets in each city, e.g. the tweets related to $\#Topic\ A$ in $City.3$, we will use the Time Index $\mathcal{I}_c$ to record the begin and end time of the effective period and the tweets published during these periods.

Algorithm 8 provides the process of the calculation of participation index. Given a set of topics $\mathcal{H}$ and the time-region frame $\langle w, r\rangle$, we first find the cities which contain tweets of every topic h in $\mathcal{H}$. For each topic h, we have the topic geographic index $\mathcal{I}_T(h)$ to help to get the set of cities which have related tweets of h and we denote the set of cities as $B(h)$. So the intersection of all $B(h)$, $h \in \mathcal{H}$, is the set $R(\mathcal{H})$ of cities containing related tweets of

every topic in $\mathcal{H}$.

Based on our definition of neighbor relation, for a tweet $v_i(t_i, l_i, \{h_i\})$, $l_i \in c$, we can find a neighbor v_j related to another topic topic h_j only when h_j has tweets published in city c and $t_i \in W_{\langle w,c\rangle}(h_j)$. For a set of topics, we can calculate the effective period $W_{\langle w,c\rangle}(h)$ of each topic in a city c. The common effective period of the topics is denoted as $W_{common}(\mathcal{H})$. For each topic h in $\mathcal{H}$, if it has a related tweet v published in $W_{common}(\mathcal{H})$ in the city c, we can find at least one neighbor tweet from other topics in $\mathcal{H}$. So v will be a supporting tweet for $\mathcal{H}$. In Algorithm 8, for each city c in $R(\mathcal{H})$, we use the time index of tweets in city c of topic h, $\mathcal{I}_c(h)$, which records the begin and end time of each effective period and the tweets published during the period, to achieve the common period $W_{common}(\mathcal{H})$. Tweets of each topic in $W_{common}(\mathcal{H})$ will be counted as supporting tweet for $\mathcal{H}$. After scanning all the cities in $R(\mathcal{H})$, we can calculate the participation ratio of h by the number of supporting tweets and get the participation index of $\mathcal{H}$.

5.2.5. The Algorithms of Queries

For the horizontal query, we need to generate all the topic associations in each time-region frame. Algorithm 9 provides this process. For each $\langle w, r\rangle$ frame in $W_q \times R_q$, we firstly put every topic in the set of a size-1 topic association $\{\mathcal{H}_c\}_1^{\langle w,r\rangle}$. In each loop, we use the apriori-gen algorithm [5] to generate size-n topic association candidates $\{\mathcal{H}\}_n$ by size-$(n-1)$ topic association $\{\mathcal{H}_c\}_{n-1}^{\langle w,r\rangle}$. Then we use the Algorithm 8 to calculate the participation index of every candidate and get the size-n topic association. After generating topic associations in every $\langle w, r\rangle$ frames, we will return every topic association $\mathcal{H} \in \{\mathcal{H}_c\}$ and the top-k frames which $\mathcal{H}$ achieves the highest participation index.

As shown in Algorithm 10, for each time-region frame $\langle w, r\rangle$, after we generate size-n topic association, we will prune the topic association by removing the association ranking out of k in $\{\mathcal{H}_c\}^{\langle w,r\rangle}$ because such topic association cannot be a subset of a topic association which has a higher participation index.

The algorithm of super topic association query is based on the method of the vertical query. The difference is that the process will begin at generating size-$(|\mathcal{H}'| + 1)$ topic

Algorithm 9: Horizontal Query

Data: Tweets $vt, l, \mathcal{H}$, topic geographic index $\mathcal{I}_T$, time index $\mathcal{I}_c$, set of topics $\mathcal{H}$, time-region frame $\langle w, r\rangle$, query $q_h(W_q, R_q, \varphi_q, \delta_q, \theta, k)$

Result: Topic association $\{\mathcal{H}_c\}$ and top-k $\langle w, r\rangle$ frames with highest participation index of each topic association $\mathcal{H}_c$

initialization;

$\{\mathcal{H}_c\} = \phi$;

for $\langle w, r\rangle \in W_q \times R_q$ **do**

  $\{\mathcal{H}_c\}^{\langle w,r\rangle} = \phi$;

  Add all topics to $\{\mathcal{H}_c\}_1^{\langle w,r\rangle}$;

  for $n = 2; \{\mathcal{H}_c\}_{n-1}^{\langle w,r\rangle} \neq \phi; n++$ **do**

    $\{\mathcal{H}\}_n = apriori - gen(\{\mathcal{H}_c\}_{n-1}^{\langle w,r\rangle})$ //Generate new topic association candidates ;

    for $\forall \mathcal{H} \in \{\mathcal{H}\}_n$ **do**

      Calculate the participation index of $\mathcal{H}$ by Algorithm 8 ;

    end

    $\{\mathcal{H}_c\}_n^{\langle w,r\rangle} = \{\mathcal{H} | pi_{\langle w,r\rangle}(\mathcal{H}) \geq \theta \wedge \mathcal{H} \in \{\mathcal{H}\}_n\}$;

  end

  $\{\mathcal{H}_c\}^{\langle w,r\rangle} = \cup_n \{\mathcal{H}_c\}_n^{\langle w,r\rangle}$;

  $\{\mathcal{H}_c\} = \{\mathcal{H}_c\} \cup \{\mathcal{H}_c\}^{\langle w,r\rangle}$;

end

for $\mathcal{H} \in \{\mathcal{H}_c\}$ **do**

  Get the top-k $\langle w, r\rangle$ frames with highest $pi_{\langle w,r\rangle}(\mathcal{H})$;

end

return $\{\mathcal{H}_c\}$ and top-k $\langle w, r\rangle$ frames for $\mathcal{H} \in \{\mathcal{H}_c\}$

association candidates and these candidates must contain all the hashtags in $\mathcal{H}'$.

Algorithm 10: Vertical Query

Data: Tweets $vt, l, \mathcal{H}$, topic geographic index $\mathcal{I}_T$, time index $\mathcal{I}_c$, set of topics $\mathcal{H}$, time-region frame $\langle w, r\rangle$, query $q_v(W_q, R_q, \varphi_q, \delta_q, \theta, k)$

Result: Top-k topic association with highest participation index in every time-region frames

initialization;

for $\langle w, r\rangle \in W_q \times R_q$ **do**

  $\{\mathcal{H}_c\}^{\langle w,r\rangle} = \phi$;

  Add all topics to $\{\mathcal{H}_c\}_1^{\langle w,r\rangle}$;

  for $n = 2; \{\mathcal{H}_c\}_{n-1}^{\langle w,r\rangle} \neq \phi; n++$ **do**

    $\{\mathcal{H}\}_n = apriori - gen(\{\mathcal{H}_c\}_{n-1}^{\langle w,r\rangle})$ //Generate new topic association candidates ;

    for $\forall \mathcal{H} \in \{\mathcal{H}\}_n$ **do**

      Calculate the participation index of $\mathcal{H}$ by Algorithm 8 ;

    end

    $\{\mathcal{H}_c\}_n^{\langle w,r\rangle} = \{\mathcal{H} | pi_{\langle w,r\rangle}(\mathcal{H}) \geq \theta \wedge \mathcal{H} \in \{\mathcal{H}\}_n\}$;

    $\{\mathcal{H}_c\}^{\langle w,r\rangle} = \cup\{\mathcal{H}_c\}_n^{\langle w,r\rangle}$;

    Prune topic association in $\{\mathcal{H}_c\}^{\langle w,r\rangle}$ and $\{\mathcal{H}_c\}_n^{\langle w,r\rangle}$ by only keeping the topic association with the participation index ranking in the top-k in $\{\mathcal{H}_c\}^{\langle w,r\rangle}$;

  end

end

return $\{\{\mathcal{H}_c\}^{\langle w,r\rangle} | \langle w, r\rangle \in W_q \times R_q\}$

5.2.6. Optimization of Querying Results

Topic Combination: Many topic pairs or groups refer to the same thing or same event. These topics may share similar words and be talked in the same area and the same time (e.g. *#amazonwishlist* and *#amazoncart*). Some official Twitter accounts even list several hashtags and tell users that all these hashtags are related to the same thing, for example, *#choicetvchemistry* and *#choicetvliplock* in Table 5.4. If such a hashtag group contains n

hashtags, it may generate 2^n topic associations. It increases computation time and leads to a huge number of redundant associations. So before we apply our algorithm, we will combine some hashtags which are related to the same thing together to make the algorithm more efficient, and less number of detected results can help to extract useful topic associations more easily.

We combine the hashtags in two steps: First, we build a hashtag similarity graph. Each hashtag is a node in this graph and there is an edge between every hashtag pair. We assign each edge a weight as:

$$\begin{aligned} &Weight(hashtag_1, hashtag_2) \\ &= \frac{|Total\ longest\ common\ substring|}{max\{|hashtag_1|, |hashtag_2|\}} \end{aligned} \tag{5.11}$$

We follow the method in [33] to get the longest substring. We remove the longest common substring of the two hashtags iteratively until there is no common substring longer than 1. For example, the longest substring of $\#amazonwishlist$ and $\#amazoncart$ is "amazon" and the weight of the edge is 0.43.

Then we run the hierarchical clustering algorithm on this graph. The distance between two clusters is defined as the average weight of edges between hashtags in these two clusters. However, we have two other problems: 1) when we stop the hierarchical clustering, some hashtags share long common substrings but actually, they do not refer to the same thing (e.g. $\#braves$ and $\#travel$). Table 5.4 lists more examples. All these examples are similar strings but their participation index is 0.

To solve these problems, we introduce the participation index here. Before we combine two clusters together, we will calculate the participation index of the new cluster. If it is less than the threshold, we won't combine them and we will go to next two clusters with the highest average weight of edges. If all the top-N candidates cannot meet the requirement, we will stop the hierarchical clustering. Table 5.4 shows that the use of participation index in topic clustering can filter out hashtags "similar in appearance but dissimilar in spirit".

Topic Pairs	Wt	PI
tamuc, *tamuk*	0.8	0
longlivela, *longlivejr*	0.8	0
rays, *raya*	0.75	0
live, *livy*	0.75	0
freedom, *freedex*	0.71	0
amazoncart, *amazonwishlist*	0.42	0.95
phillies, *philly*	0.625	0.81
travelnursing, *travelhealthcare*	0.375	0.68
nationaldonutday, *nationaldoughnutday*	0.84	0.48
choicetvchemistry, *choicetvliplock*	0.47	0.43

TABLE 5.4. Apply participation index in topic combination

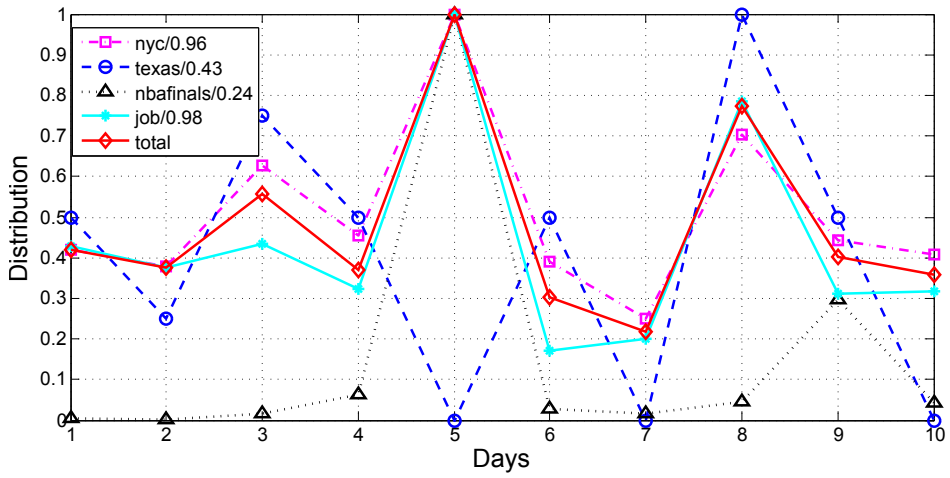
Topic Filtering: Many topics follows a rhythm of a typical day. For example, they happen more when there are more people using Twitter such as lunchtime. These topics such as job advertisement are white noises and do not relate to other topics. Here we propose a method to filter out these topics to prevent them from participating in most topics associations.

In a time-region frame $\langle w, r\rangle$, we calculate the correlation coefficient between topic h and all the topics by their distributions of number of tweets in each city c in r during the time window w as:

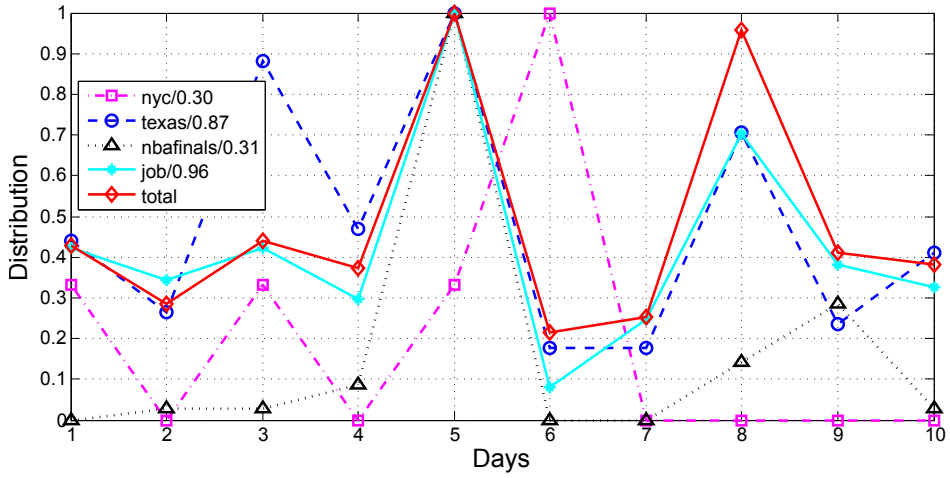
$$Cor(h, \mathcal{H}_{total}) = \frac{\sum_{u=1}^{n}(|v_u^h| - \overline{|v_u^h|})(|v_u^{\mathcal{H}}| - \overline{|v_u^{\mathcal{H}}|})}{\sqrt{\sum_{u=1}^{n}(|v_u^h| - \overline{|v_u^h|})^2 \sum_{u=1}^{n}(|v_u^{\mathcal{H}}| - \overline{|v_u^{\mathcal{H}}|})^2}} \tag{5.12}$$

Here, $V_{\langle u,c\rangle}(h) = \{v_1^h, v_2^h, \ldots\}$, $V_{\langle u,c\rangle}(\mathcal{H}) = \{v_1^{\mathcal{H}}, v_2^{\mathcal{H}}, \ldots\}$, and u denotes a unit of time, such as one day.

When a topic correlates with total volume of the tweets above a threshold, these



(a) New York



(b) Austin

FIGURE 5.10. The distribution of tweets of different topics in New York and Austin.

tweets are considered as uninteresting and are excluded from the topic association mining. Figures 10(a) shows that $\#nyc$ is a topic highly correlated to the number of tweets in New York during a day while $\#texas$ (correlation index of 0.87) is not. In Austin however as shown in Figure 10(b), $\#texas$ is a highly correlated topic (correlation index of 0.87) while $\#nyc$ and $\#nbafinals$ are not. $\#job$ is a highly correlated topic in both cities. So the topics like $\#job$ will not be considered in topic association mining since such topics are not affected by other topics or real-world events.

5.2.7. Experimental Results

In this section, we first introduce the dataset used in the experiment. Then we evaluate the effect of parameters of queries by comparing the running time of each query. Finally, we demonstrate the experiment results in some time-region frames to explain the meaning of the topic association. We show the results of topic filtering. We compare the results produced by our algorithms with human judges.

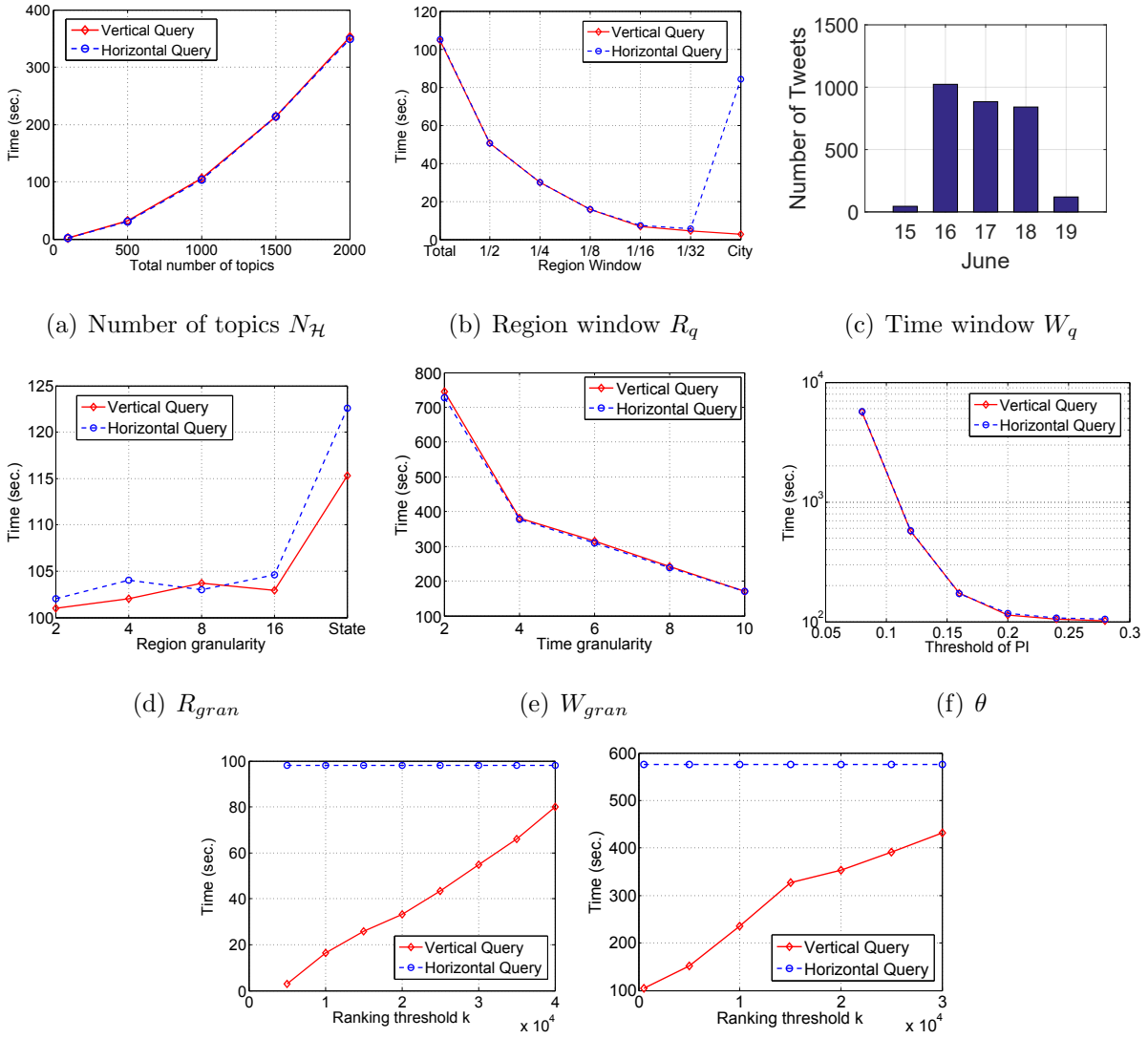
Parameter	Definition	Default value	Test scope
$N_{\mathcal{H}}$	The total number of topics used	1,000	$[100, 2000]$
W_q	The total time windows	20 days	[1, 20]
R_q	The total region window	US	[city level, U.S.]
W_{gran}	The granularity of time window	1	[1, 20]
R_{gran}	The granularity of region window	1	[city level, U.S.]
φ_q	Time threshold of neighbor relation	1 hour	fixed
δ_q	Distance threshold of neighbor relation	City level	fixed
θ	The threshold of the participation index	0.8	[0.04 0.8]
k	The ranking threshold of the query	40,000	[500 40,000]

TABLE 5.5. Parameters used in queries

We collected tweets published from May 21 to June 15, 2015, using Twitter Streaming API[1] with the spatial bounding box $[24°3'N, 127°7'W, 48°8'N, 65°5'W]$, which covers U.S. mainland. The dataset consists 27,956,257 tweets from 492,492 topics. We removed the topics with less than 1,000 related tweets, and finally, we selected 10,201,960 tweets which contain both geotag and hashtag from 2,000 topics.

Firstly, we evaluate the effect of different parameters on the running time. Table 5.5 gives the list of parameters and the default values. Figure 11(a) shows the running time of vertical query and horizontal query under a different number of topics. More topics mean

[1]https://dev.twitter.com/streaming/overview



(a) Number of topics $N_{\mathcal{H}}$ (b) Region window R_q (c) Time window W_q

(d) R_{gran} (e) W_{gran} (f) θ

(g) Ranking threshold k (1) (h) Ranking threshold k (2)

FIGURE 5.11. Running time under different setting of parameters.

more topic associations under the same settings and the increase of the number of topic associations can be exponential in the worst case. The running time in Figure 11(a) shows this trend. In our algorithm, each topic will have a time index in each city and a geographic index to record its distribution. So the increase of the number of topics will take more memory to store the indexes but it has little influence on the running time.

In this experiment, we use four parameters to describe the time-region frame. W_q and R_q are largest time and region frame and W_{gran} and R_{gran} are use used to divide the largest

time and region frames into smaller ones, i.e. we divide W_q and R_q based on the length of W_{gran} and R_{gran}.

Firstly, we test the influence of the region window R_q. We divide the cities in U.S. into 1 to 32 groups equally and record the average querying time on these groups. Figure 11(b) gives the results of this experiment. The numbers on the x-axis represent the R_{gran} proportion of each group of the total region window. For example, 1/2 means that we divide the whole area of the U.S. into two groups and the y-axis is the average running time of the queries on these two groups. The results show that the running time has a linear relationship with the scope of the region window, e.g. the running time of two groups is about half of the "total" case and twice as much as the "1/4" case. The two queries perform differently in the "city" case, where we set each city as the region window. Running the queries in city granularity will generate a large number of topic associations. So the horizontal query has a longer running time since the vertical query can prune more topic association candidates. We will provide more analysis in the following section.

The influence of time window size is shown in Figure 1(b). Similar as the results of changing the total region window, the running time has a linear relationship with the time window. The increase of time and region window needs additional indexes to record the distribution of new tweets. If we only increase the region window, for each topic set, we need to scan more cities to calculate the participation. If we increase the time window only, for each city we need to scan a larger time window by the time index. That explains why the running time increases linearly with the region and time window.

Then we fixed the largest time and region windows and change the granularity of the sub-time and sub-region windows. If we set smaller time or region window, each query will be faster, but the query needs to be executed more times. When changing the granularity of the region window (Figure 11(d)), we can see that the total running time does not change too much. It only increases in the "state" case because some state only contains one city and the query will take longer time. Figure 11(e) is the results of the running time of different time window granularities. The total running time can be smaller when we set a longer time

window granularity. Actually, we can estimate the results in Figure 11(d) and 11(e) by the results in Figure 11(b) and 1(b).

City:	New York	Chicago	Houston
1	braves, bruins	rockets, rays	traffic, houstonastros
2	nationaldonutday, uclfinal	astros, rockets	traffic, houstontexans, houstonastros
3	sundayfunday, sunday	astros, rockets, rays	traffic, houstontexans
4	gsw, nbafinals2015	astros, rays, texasedmfamily	houstontexans, houstonastros
5	nbafinals, cle	rockets, rays, texasedmfamily	whodatnation, gramfam

TABLE 5.6. Vertical query

The threshold of the participation index greatly affects the number of topic associations. As shown in Figure 11(f), with the decreasing of the value of the threshold θ, the running time increase quickly since there will be an exponential growth of the number of topic associations. Here the ranking threshold k of the vertical query is big enough, so the optimization is not useful and the running time of both the two queries is similar.

	gsw, nbafinals2015	nbafinals, cle
Frame 1	New York, June 7	New York, June 7
Frame 2	New York, June 4	New York, June 4
Frame 3	New York, June 16	New York, June 14

TABLE 5.7. Horizontal query

In Figure 11(g) and 11(h), we test the influence of the ranking threshold k of vertical query. In these two experiments, we set the value of the threshold of participation index θ as 0.8 and 0.12 separately. In both the two cases, the ranking threshold greatly affects the running time. So when this threshold is not large enough, during the generating step,

we can prune a part of topic associations which have their participation index larger than the threshold θ. So the running time of the vertical query will be much smaller than the horizontal query. With the increase of the ranking threshold k, the gap of the running time of the two queries becomes smaller.

The Topic Association: In this experiment, we use the results of the two queries to show that our algorithm can detect meaningful topic associations. We also compare the results with/without the topic filtering method.

Vertical Query: Table 5.6 shows the top-5 topic associations (sorted by participation index) in New York, Chicago, and Houston from May 25 to Jun 15. The relationship between hashtags in most topic associations can be easily revealed by investigating the real world event they refer to. For example, "$\#gsw, \#nbafinals2015$" and "$\#nbafinals, \#cle$", $\#gsw$ and $\#cle$ refer to the two teams which play against in the NBA final, 2015.

$\#nbafinals$	PI	$\#amazoncart$	PI
$\#gsw$	0.79	$\#amazonwishlist$	0.95
$\#cle$	0.76	$\#einkaufen$	0.94
$\#nbafinals2015$	0.71	$\#bargains$	0.92

TABLE 5.8. Super topic association query

Horizontal Query: Table 5.7 shows examples of horizontal query results. Here we select two topic associations in Table 5.6, $gsw, nbafinals2015$ and $nbafinals, cle$. For the first topic association, the days when it achieves highest participation indexes is June 7, 4, and 16, and for the second example, they are June 7, 4, and 14. If we check the schedule of NBA finals in 2015, we will find that there exists a match in all these days. So we can see that the horizontal query will return the most related time-region frames.

Super Topic Association Query: Table 5.8 lists two examples for super topic association query. We set $\mathcal{H}'$ as $\{\#nbafinals\}$ and $\{\#amazoncart\}$ and the query returns the topic associations containing those hashtags.

Before Filtering	After Filtering
nyc, newyork	braves, bruins
job, hiring	nationaldonutday, uclfinal
job, nyc	sundayfunday, sunday
job, newyork	gsw, nbafinals2015
hiring, jobs	nbafinals, cle

TABLE 5.9. Topic filtering

Topic Filtering: In Table 5.9 we compare the querying results with and without the topic filtering. We apply the same settings with the experiment for the vertical query. Here we list the top-5 topic associations which have the highest participation indexes in New York from May 25 to Jun 15. If we do not apply the topic filtering method, we will detect lots of topic associations including $nyc, newyork$ and $job, hiring$. They have a very high participation index because people discuss them very often and they can form supporting cliques easily. After we filter such topics, we can see that the new results are more meaningful. Most of them are caused by some real-world event.

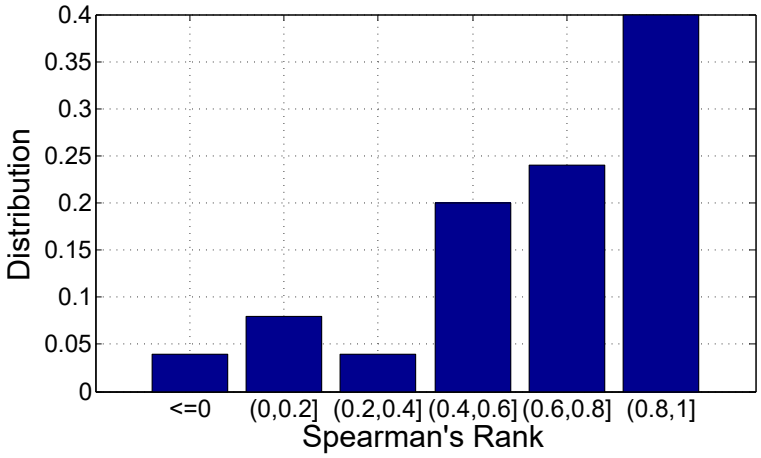


FIGURE 5.12. Compare the ranking based on the participation indexes with the human judgment.

To verify the effectiveness of participation index, which greatly affects the final results,

we compare the experimental results with human judgment in this experiment. Since it is difficult to give an appropriate measurement to judge the tightness between topics, we only compare the tightness of topic sets. First, we create topic set groups. Each topic set group contains 5 topic sets and each topic sets contain two topics. The participation indexes of these five topic sets should be in the range of $(0, 0.05]$, $(0.05, 0.15]$, $(0.15, 0.3]$, $(0.3, 0.45]$, and $(0.45, 1]$ respectively. The gap of participation indexes of each two topic set should be larger than 0.05. Based on the participation indexes, we can get a rank of these topic sets X. Then we manually compare the tightness of topic sets in the same group and give a rank Y based on their judgment. To make the judgment easier and more accurate, the topic sets in the same group will share a common topic. Finally, we use Spearman's rank correlation coefficient to compare the human judgment with the experiment (rank based on the participation index).

$$\rho = 1 - \frac{6\sum(x_i - y_i)^2}{n(n^2 - 1)} \tag{5.13}$$

The x_i and y_i is the ranking of the ith element in the topic set group given by participation index and human judgment. Since we fix the size of topic set group as five, the value of n is 5. The average value of the Spearman's rank correlation of our 200 cases is 0.68, which illustrates that the participation index can measure the tightness of topics effectively. Figure 5.12 shows the distribution of the values of the Spearman's rank correlation coefficient. Most cases have the Spearman's rank correlation coefficient larger than 0.4 and 40 percent cases have the value larger than 0.8.

CHAPTER 6

CONCLUSION

In this dissertation, we study the location estimation, event detection, and topic association analysis. For location estimation, we propose different indexes to measure social closeness and friend structure, the friend co-location, and the local social coefficient. We investigate the relationship between the two indexes and geographic distance. Based on this, we develop two user location estimation models, the friend co-location based model and the local social coefficient mode. Since the two models work better in different cases, we further combine these two models to avoid the disadvantages of each. To improve the estimation accuracy, we also propose an anomaly friends elimination method and a confidence-based iteration method. Finally, we test these models on two different datasets and demonstrate how the models work under different situations. The experiment results show that our method can improve the estimation accuracy by 5%-20% compared with the baseline algorithms.

Then we propose a framework for local event detection. The proposed LEDS framework can detect smaller-scale and more types of local events. The framework uses terms instead of number of tweets to detected events which are more sensitive. The density-based clustering and time-location-distribution-based method can help to cluster tweets which may be related to the same event effectively. Finally, we extract the time, location, and content information from each cluster. The results show that greater variety of local events can be detected by our method including shows, disasters, parties, business activities, and personal meetings.

Finally, we introduce the algorithm for mining topic associations on Twitter data. We apply participation index as the measurement of topic closeness. We define three different kinds of queries to mining the topic association. For each query, we design the algorithm for it. Two optimization methods, topic filtering and topic combination are used to help in getting a better query results. We test our method on a large Twitter data set with 27,956,257 tweets which contain both hashtags and geo-tag. The results of the experiment show that

the mining algorithms are effective and efficient. We also compare the experimental results with human judgment to verify the effectiveness of using the participation index to measure the closeness among topics.

In the future work, we will focus on improving the accuracy of real-time location estimation especially when the user is traveling to other cities. We will also try to improve the results of the event detection by filtering more meaningless events and provide a better description of the events. We can also extend the work of mining topic association to event association detection.

REFERENCES

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz, *Eventweet: Online localized event detection from twitter*, Proceedings of the VLDB Endowment 6 (2013), no. 12, 1326–1329.

[2] Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu, and Benyuan Liu, *Predicting flu trends using twitter data*, INFOCOM WKSHPS, IEEE, 2011, pp. 702–707.

[3] Charu C. Aggarwal, Yan Li, Jianyong Wang, and Jing Wang, *Frequent pattern mining with uncertain data*, Proceedings of the 15th ACM SIGKDD, 2009.

[4] Rakesh Agrawal and Ramakrishnan Srikant, *Fast algorithms for mining association rules in large databases*, Proc. VLDB, 1994.

[5] Rakesh Agrawal, Ramakrishnan Srikant, et al., *Fast algorithms for mining association rules*, VLDB, vol. 1215, 1994, pp. 487–499.

[6] Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer, *Web-a-where: geotagging web content*, SIGIR, ACM, 2004, pp. 273–280.

[7] Paolo Arcaini, Gloria Bordogna, Dino Ienco, and Simone Sterlacchini, *User-driven geo-temporal density-based exploration of periodic and not periodic events reported in social networks*, Information Sciences (2016).

[8] Sitaram Asur and Bernardo A Huberman, *Predicting the future with social media*, WI-IAT, vol. 1, IEEE, 2010, pp. 492–499.

[9] Lars Backstrom, Eric Sun, and Cameron Marlow, *Find me if you can: improving geographical prediction with social and spatial proximity*, WWW, ACM, 2010, pp. 61–70.

[10] Hila Becker, Mor Naaman, and Luis Gravano, *Beyond trending topics: Real-world event identification on twitter*, (2011), 438–441.

[11] Jon Louis Bentley, *Multidimensional binary search trees used for associative searching*, Communications of the ACM 18 (1975), no. 9, 509–517.

[12] Thomas Bernecker, Tobias Emrich, Hans-Peter Kriegel, Matthias Renz, Stefan Zankl, and Andreas Züfle, *Efficient probabilistic reverse nearest neighbor query processing on uncertain data*, Proc. VLDB Endow. (2011).

[13] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment 2008 (2008), no. 10, P10008.

[14] Han Bo and Paul COOK1 Timothy BALDWIN, *Geolocation prediction in social media data by finding location indicative words*, Proceedings of COLING 2012: Technical Papers (2012), 1045–1062.

[15] Coen Bron and Joep Kerbosch, *Algorithm 457: finding all cliques of an undirected graph*, Commun. ACM (1973).

[16] Ceren Budak, Theodore Georgiou, Divyakant Agrawal, and Amr El Abbadi, *Geoscope: Online detection of geo-correlated information trends in social networks*, VLDB 7 (2013), no. 4, 229–240.

[17] Ceren Budak, Theodore Georgiou, and Divyakant Agrawal Amr El Abbadi, *Geowatch: Online detection of geo-correlated information trends in social networks*, Tech. report, Technical report, UCSB, 2013.

[18] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, *Mafia: a maximal frequent itemset algorithm*, TKDE (2005).

[19] Simon Carter, Manos Tsagkias, and Wouter Weerkamp, *Twitter hashtags: Joint translation and clustering*, ACM WebSci (2011).

[20] Mete Celik, James M. Kang, and Shashi Shekhar, *Zonal co-location pattern discovery with dynamic parameters*, Proceedings of the Seventh IEEE ICDM, 2007.

[21] Junghoon Chae, Dennis Thom, Harald Bosch, Yun Jang, Ross Maciejewski, David S Ebert, and Thomas Ertl, *Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition*, VAST, IEEE, 2012, pp. 143–152.

[22] Swarup Chandra, Latifur Khan, and FB Muhaya, *Estimating twitter user location using social interactions–a content based approach*, SocialCom, IEEE, 2011, pp. 838–843.

[23] Tao Cheng and Thomas Wicks, *Event detection using twitter: a spatio-temporal approach*, PloS one 9 (2014), no. 6, e97807.

[24] Zhiyuan Cheng, James Caverlee, and Kyumin Lee, *You are where you tweet: a content-based approach to geo-locating twitter users*, Proceedings of the 19th ACM international conference on Information and knowledge management, ACM, 2010, pp. 759–768.

[25] Eunjoon Cho, Seth A Myers, and Jure Leskovec, *Friendship and mobility: user movement in location-based social networks*, Proceedings of the 17th ACM SIGKDD, ACM, 2011, pp. 1082–1090.

[26] Aaron Clauset, Mark EJ Newman, and Cristopher Moore, *Finding community structure in very large networks*, Physical review E 70 (2004), no. 6, 066111.

[27] Ryan Compton, David Jurgens, and David Allen, *Geotagging one hundred million twitter accounts with total variation minimization*, arXiv preprint arXiv:1404.7152 (2014).

[28] Michele Coscia, Fosca Giannotti, and Dino Pedreschi, *A classification for community discovery methods in complex networks*, Statistical Analysis and Data Mining 4 (2011), no. 5, 512–546.

[29] Clodoveu A Davis Jr, Gisele L Pappa, Diogo Rennó Rocha de Oliveira, and Filipe de L Arcanjo, *Inferring the location of twitter messages based on user relationships*, Transactions in GIS 15 (2011), no. 6, 735–751.

[30] Xiao Feng, Shuwu Zhang, Wei Liang, and Zhe Tu, *Real-time event detection based on geo extraction and temporal analysis*, ADMA, Springer, 2014, pp. 137–150.

[31] David Flatow, Mor Naaman, Ke Eddie Xie, Yana Volkovich, and Yaron Kanza, *On the accuracy of hyper-local geotagging of social media content*, Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, ACM, 2015, pp. 127–136.

[32] Santo Fortunato, *Community detection in graphs*, Physics Reports 486 (2010), no. 3, 75–174.

[33] Carol Friedman and Robert Sideli, *Tolerating spelling errors during patient validation*, Computers and Biomedical Research 25 (1992), no. 5, 486–509.

[34] D.G. Corneil G.D. Mulligan, *Corrections to bierstone's algorithm for generating cliques*, Commun. ACM (1972).

[35] Michelle Girvan and Mark EJ Newman, *Community structure in social and biological networks*, Proceedings of the National Academy of Sciences 99 (2002), no. 12, 7821–7826.

[36] Kimberly Glasgow and Clayton Fink, *Hashtag lifespan and social networks during the london riots*, Social Computing, Behavioral-Cultural Modeling and Prediction, Springer, 2013, pp. 311–320.

[37] K. Gouda and M.J. Zaki, *Efficiently mining maximal frequent itemsets*, ICDM, 2001.

[38] Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He, *Document summarization based on data reconstruction.*, AAAI, 2012.

[39] Alexander Hinneburg and Daniel A Keim, *An efficient approach to clustering in large multimedia databases with noise*, KDD, vol. 98, 1998, pp. 58–65.

[40] Tin Kam Ho, Jonathan J Hull, and Sargur N Srihari, *Decision combination in multiple classifier systems*, Pattern Analysis and Machine Intelligence, IEEE Transactions on 16 (1994), no. 1, 66–75.

[41] Jianbin Huang, Heli Sun, Yaguang Liu, Qinbao Song, and Tim Weninger, *Towards online multiresolution community detection in large-scale networks*, PloS one 6 (2011), no. 8, e23829.

[42] Yan Huang, Jian Pei, and Hui Xiong, *Mining co-location patterns with rare events from spatial data sets*, Geoinformatica (2006).

[43] Yan Huang, Shashi Shekhar, and Hui Xiong, *Discovering colocation patterns from spatial data sets: A general approach*, TKDE (2004).

[44] Bernardo A. Huberman, Daniel M. Romero, and Fang Wu, *Social networks that matter: Twitter under the microscope*, CoRR abs/0812.1045 (2008).

[45] David Inouye and Jugal K Kalita, *Comparing twitter summarization algorithms for multiple post summaries*, PASSAT and SocialCom, IEEE, 2011, pp. 298–306.

[46] David Jurgens, *That's what friends are for: Inferring location in online social media platforms based on social relationships*, ICWSM, 2013.

[47] Krishna Y Kamath, James Caverlee, Kyumin Lee, and Zhiyuan Cheng, *Spatio-temporal dynamics of online memes: a study of geo-tagged tweets*, WWW, International World Wide Web Conferences Steering Committee, 2013, pp. 667–678.

[48] Sheila Kinsella, Vanessa Murdock, and Neil O'Hare, *I'm eating a sandwich in glasgow: modeling locations with tweets*, Proceedings of the 3rd international workshop on Search and mining user-generated contents, ACM, 2011, pp. 61–68.

[49] Longbo Kong, Zhi Liu, and Yan Huang, *Spot: Locating social media users based on social network context*, Proceedings of the VLDB Endowment 7 (2014), no. 13, 1681–1684.

[50] John Krumm and Eric Horvitz, *Eyewitness: Identifying local events via space-time signals in twitter feeds*, (2015).

[51] Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu, *On recommending hashtags in twitter networks*, Social Informatics, Springer, 2012, pp. 337–350.

[52] Erwan Le Martelot and Chris Hankin, *Fast multi-scale detection of relevant communities in large-scale networks*, The Computer Journal 56 (2013), no. 9, 1136–1150.

[53] Ryong Lee and Kazutoshi Sumiya, *Measuring geographical regularities of crowd behaviors for twitter-based geo-social event detection*, LBSN, ACM, 2010, pp. 1–10.

[54] Ryong Lee, Shoko Wakamiya, and Kazutoshi Sumiya, *Discovery of unusual regional social activities using geo-tagged microblogs*, WWW 14 (2011), no. 4, 321–349.

[55] Rui Li, Kin Hou Lei, Ravi Khadiwala, and Kevin Chen-Chuan Chang, *Tedas: A twitter-based event detection and analysis system*, ICDE, IEEE, 2012, pp. 1273–1276.

[56] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang, *Towards social user profiling: unified and discriminative influence model for inferring home locations*, Proceedings of the 18th ACM SIGKDD, ACM, 2012, pp. 1023–1031.

[57] Zhi Liu and Yan Huang, *Closeness and structure of friends help to estimate user locations*, International Conference on Database Systems for Advanced Applications, Springer, 2016, pp. 33–48.

[58] Alan M MacEachren, Anthony C Robinson, Anuj Jaiswal, Scott Pezanowski, Alexander Savelyev, Justine Blanford, and Prasenjit Mitra, *Geo-twitter analytics: Applications in crisis management*, 25th International Cartographic Conference, 2011, pp. 3–8.

[59] Jalal Mahmud, Jeffrey Nichols, and Clemens Drews, *Home location identification of twitter users*, CoRR abs/1403.2345 (2014).

[60] Robert Munro, Sanjay Chawla, and Pei Sun, *Complex spatial relationships*, Proceedings of the Third IEEE ICDM, 2003.

[61] Ani Nenkova and Lucy Vanderwende, *The impact of frequency on summarization*, Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101 (2005).

[62] Mark EJ Newman, *Mixing patterns in networks*, Physical Review E 67 (2003), no. 2, 026126.

[63] Mark EJ Newman and Michelle Girvan, *Finding and evaluating community structure in networks*, Physical review E 69 (2004), no. 2, 026113.

[64] Jukka-Pekka Onnela, Samuel Arbesman, Marta C González, Albert-László Barabási, and Nicholas A Christakis, *Geographic constraints on social network groups*, PLoS one 6 (2011), no. 4, e16939.

[65] Saša Petrović, Miles Osborne, and Victor Lavrenko, *Streaming first story detection with application to twitter*, NAACL, Association for Computational Linguistics, 2010, pp. 181–189.

[66] Jay M Ponte and W Bruce Croft, *A language modeling approach to information retrieval*, Proceedings of the 21st annual international conference on Research and development in information retrieval, ACM, 1998, pp. 275–281.

[67] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha, *Communities in networks*, Notices of the AMS 56 (2009), no. 9, 1082–1097.

[68] Reid Priedhorsky, Aron Culotta, and Sara Y Del Valle, *Inferring the origin locations*

of tweets with quantitative confidence, Proceedings of the 17th ACM conference on Computer supported cooperative work and social computing, ACM, 2014, pp. 1523–1536.

[69] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin, *Twitter user geolocation using a unified text and network prediction model*, arXiv preprint arXiv:1506.08259 (2015).

[70] Jörg Reichardt and Stefan Bornholdt, *Statistical mechanics of community detection*, Physical Review E 74 (2006), no. 1, 016110.

[71] Alan Ritter, Oren Etzioni, Sam Clark, et al., *Open domain event extraction from twitter*, ACM SIGKDD, ACM, 2012, pp. 1104–1112.

[72] Stephen Roller, Michael Speriosu, Sarat Rallapalli, Benjamin Wing, and Jason Baldridge, *Supervised text-based geolocation using language models on an adaptive grid*, Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Association for Computational Linguistics, 2012, pp. 1500–1510.

[73] Adam Sadilek, Henry Kautz, and Jeffrey P Bigham, *Finding your friends and following them to where you are*, WSDM, ACM, 2012, pp. 723–732.

[74] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo, *Earthquake shakes twitter users: real-time event detection by social sensors*, WWW, ACM, 2010, pp. 851–860.

[75] Takeshi Sakaki, Masahide Okazaki, and Yoshikazu Matsuo, *Tweet analysis for real-time event detection and earthquake reporting system development*, TKDE 25 (2013), no. 4, 919–931.

[76] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling, *Twitterstand: news in tweets*, ACM SIGSPATIAL, ACM, 2009, pp. 42–51.

[77] Salvatore Scellato, Cecilia Mascolo, Mirco Musolesi, and Vito Latora, *Distance matters: geo-social metrics for online social networks*, Proceedings of the 3rd conference on Online social networks, 2010, pp. 8–8.

[78] Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo, *Socio-spatial properties of online location-based social networks.*, ICWSM 11 (2011), 329–336.

[79] Axel Schulz, Aristotelis Hadjakos, Heiko Paulheim, Johannes Nachtwey, and Max Mühlhäuser, *A multi-indicator approach for geolocalization of tweets.*, ICWSM, 2013.

[80] Shashi Shekhar and Yan Huang, *Discovering spatial co-location patterns: A summary of results*, SSTD, 2001.

[81] SocialMediaToday, *http://socialmediatoday.com/irfan-ahmad/1854311/twitter-statistics-ipo-infographic*, 2013.

[82] Takuya Sugitani, Masumi Shirakawa, Tenshi Hara, and Shojiro Nishio, *Detecting local events by analyzing spatiotemporal locality of tweets*, WAINA, IEEE, 2013, pp. 191–196.

[83] Ramine Tinati, Leslie Carr, Wendy Hall, and Jonny Bentwood, *Identifying communicator roles in twitter*, WWW, ACM, 2012, pp. 1161–1168.

[84] Oren Tsur and Ari Rappoport, *What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities*, WSDM, ACM, 2012, pp. 643–652.

[85] Subodh Vaid, Christopher B Jones, Hideo Joho, and Mark Sanderson, *Spatio-textual indexing for geographical search on the web*, SSTD, Springer, 2005, pp. 218–235.

[86] Yves van Gennip, Blake Hunter, Raymond Ahn, Peter Elliott, Kyle Luh, Megan Halvorson, Shannon Reid, Matthew Valasik, James Wo, George E Tita, et al., *Community detection using spectral clustering on sparse geosocial data*, SIAM Journal on Applied Mathematics 73 (2013), no. 1, 67–83.

[87] Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova, *Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion*, Information Processing & Management 43 (2007), no. 6, 1606–1618.

[88] Lizhen Wang, Pinping Wu, and Hongmei Chen, *Finding probabilistic prevalent colocations in spatially uncertain data sets*, IEEE TKDE (2013).

[89] Xiao Fan Wang and Guanrong Chen, *Complex networks: small-world, scale-free and beyond*, Circuits and Systems Magazine, IEEE 3 (2003), no. 1, 6–20.

[90] Xufei Wang, Huan Liu, Peng Zhang, and Baoxin Li, *Identifying information spreaders*

in twitter follower networks, Tech. Report TR-12-001, School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, 2012.

[91] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai, *Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs*, CIKM, ACM, 2011, pp. 2541–2544.

[92] Andreas Weiler, Michael Grossniklaus, and Marc H Scholl, *Run-time and task-based performance of event detection techniques for twitter*, CAiSE, Springer, 2015, pp. 35–49.

[93] Jianshu Weng and Bu-Sung Lee, *Event detection in twitter.*, ICWSM 11 (2011), 401–408.

[94] Lei Yang, Tao Sun, Ming Zhang, and Qiaozhu Mei, *We know what@ you# tag: does the dual role affect hashtag adoption?*, WWW, ACM, 2012, pp. 261–270.

[95] M.J. Zaki, *Scalable algorithms for association mining*, TKDE (2000).

[96] Chengxiang Zhai and John Lafferty, *A study of smoothing methods for language models applied to ad hoc information retrieval*, Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2001, pp. 334–342.

[97] Xin Zhang, Nikos Mamoulis, David W. Cheung, and Yutao Shou, *Fast mining of spatial collocations*, ACM SIGKDD, 2004.

[98] Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming Li, *Topical keyphrase extraction from twitter*, ACL HLT, Association for Computational Linguistics, 2011, pp. 379–388.

[99] Xiangmin Zhou and Lei Chen, *Event detection over twitter social media streams*, The VLDB journal 23 (2014), no. 3, 381–400.

[100] Yinghua Zhou, Xing Xie, Chuang Wang, Yuchang Gong, and Wei-Ying Ma, *Hybrid index structures for location-based web search*, CIKM, ACM, 2005, pp. 155–162.

[101] Qinghua Zou, W.W. Chu, and Baojing Lu, *Smartminer: a depth first algorithm guided by tail information for mining maximal frequent itemsets*, ICDM, 2002.