A PRELIMINARY APPROACH TO DETECT AND TRACK EVENTS IN SOCIAL MEDIA

BY

MINYI TANG

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Professor Tarek F. Abdelzaher
ABSTRACT

Many algorithms have been proposed to model spatiotemporal events in both sensor network and social networks. However, most of them cannot fulfill the task in a social network data streaming context. We proposed an evolving Mean Shift clustering based algorithm to formulate a robust system to automatically detect and track events in social network media. We also demonstrate its performance in empirical experiments. Our online system can be adapted and maintained without consuming too much system resources which may formulate a good basis for event detection and tracking in the domain of real-time social network media.
To my parents, for their love and support.
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Prof. Tarek F. Abdelzaher for the continuous support of my master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study.
# TABLE OF CONTENTS

LIST OF TABLES ................................................. vi
LIST OF FIGURES ............................................... vii

CHAPTER 1  INTRODUCTION ................................. 1

CHAPTER 2  GENERAL SYSTEM ARCHITECTURE .............. 3
  2.1 Crawler Layer ........................................ 4
  2.2 Parser layer .......................................... 4
  2.3 Data Analysis Service ................................. 4
  2.4 Data Summarizing Generation Service ................. 4
  2.5 Data Monitoring Service .............................. 5
  2.6 Renderer Layer ........................................ 6

CHAPTER 3  ALGORITHM AND ANALYSIS ...................... 7
  3.1 Problem Formulation .................................. 7
  3.2 Algorithm Analysis ................................... 8
  3.3 Intuition ............................................... 8
  3.4 Evolving Mean Shift Clustering Algorithm .......... 9

CHAPTER 4  EVENT DETECTION IN INSTAGRAM ............ 13
  4.1 Event Detection for Paris ............................. 14

CHAPTER 5  EVENT TRACKING ............................... 16
  5.1 Event Tracking for Marathons ......................... 16
  5.2 Event Tracking for U2 Concert ....................... 17

CHAPTER 6  FUTURE WORK ................................. 20
  6.1 Synthetic Data for More Experiment Result .......... 20
  6.2 Exploration of Wireless Sensor Network Tracking Algo-
      rithms on Existing Data .............................. 21
  6.3 System API and Integration ........................... 21

CHAPTER 7  CONCLUSION ................................. 22

REFERENCES .................................................. 23
LIST OF TABLES

5.1  A Snapshot of U2 Concert Tour City List . . . . . . . . . . . 18
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>System Architecture Overview</td>
<td>3</td>
</tr>
<tr>
<td>2.2</td>
<td>UI for renderer Layer</td>
<td>6</td>
</tr>
<tr>
<td>4.1</td>
<td>Paris Event Detection on November 12th</td>
<td>14</td>
</tr>
<tr>
<td>4.2</td>
<td>Paris Event Detection on November 13th (Terror Attack)</td>
<td>14</td>
</tr>
<tr>
<td>5.1</td>
<td>Event Tracking for Boston Marathon 2015</td>
<td>16</td>
</tr>
<tr>
<td>5.2</td>
<td>Event Tracking for NYC Marathon 2015</td>
<td>16</td>
</tr>
<tr>
<td>5.3</td>
<td>U2 Concert Tracking</td>
<td>18</td>
</tr>
<tr>
<td>5.4</td>
<td>U2 Concert Tracking Experimental Result (Distance)</td>
<td>19</td>
</tr>
<tr>
<td>5.5</td>
<td>U2 Concert Tracking Experimental Result (Time)</td>
<td>19</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

The rapid growth of social media such as Instagram and Twitter witnesses a booming trend of events posting and sharing online. For instance, people tend to share events on-line about what are happening around them in a real time manner. Thanks to the advances in positioning technology and popularity of the mobile devices, many social media information shared online also contains abundant geographical location data. Rich data in social networks enables us to find good samples of detecting and tracking events in the social network.

Event detection and tracking have been one of the most popular and interesting topics in the novel data research area. And social networks imposes a good data platform for such researches. First of all

One interesting question is whether we can detect and track events based on information crawled from social media. Capturing the changing behavior of an event has a profound impact for people’s lives. Due to its real time sharing nature, social networks may be able to detect and tracking evolutionary events much faster than traditional media press. Taking Paris’ terror attack as an example, during our experiments, we observed a vast change from Instagram users’ posts near the attacked area on a real-time basis. This enables users to grasp real-time news updates and feeds from a completely novel data source.

In this research project, we study how to build a system that can automatically collect, parse, analyze data from Instagram and provide real time event detection and tracking functionality. More importantly, we propose a baseline approach that is able to detect and tracking events in social networks in a real-time manner.

We evaluate our approaches by using the crawled data from Instagram containing various hashtags including #paris, #bostonmarathon, #nycmarathon, #u2ieconcert, #u2 with a total of 1244295 post entries. We perform
empirical experiments on our data set and compare our results to the publicly available truth data. We show that our work is very accurate towards the purpose of real-time event detection and tracking.
CHAPTER 2

GENERAL SYSTEM ARCHITECTURE

Figure 2.1 indicates the system architecture for our project. It is composed of 4 Major components. The data crawler is implemented using Node.js framework. The crawler layer provides APIs for the client to subscribe to a new event or geo-locations. There are 3 other parts on the server side, mainly implemented in Python. The parser layer retrieves the objects that are sent by the crawler layer and deserializes them into lists of signals. It also filters out any Instagram posts that have little information, for instance, posts without and geographical location data or time data. The analysis layer implements our algorithm and logic to capture the changing event behavior. And when the result is computed, it will be sent back to the render layer with updated analyzed information. The renderer layer is implemented using javascript and HTML. It employs Google map APIs to render information on the map. Renderer layer is actively connected to analysis layer so it will illustrate the most updated crawled and analyzed data.

Figure 2.1: System Architecture Overview
2.1 Crawler Layer

Our system prepare for tracking by going through an initialization process. The process is used to register our service to Instagram, set up connections with Instagram API, get client input information and set up the listening service and get data stream from Instagram subscription API, it then feeds data into parser layer and store it into the database.

2.2 Parser layer

Parser layer reads data from the database, deserialize json format into controllable object so we can perform analysis on it. It also performs data cleaning on the data by filtering out any data objects with missing geographical information or temporal information. Parser layer is also responsible for feeding the updated data stream into Analysis Layer.

2.3 Data Analysis Service

Analysis layer implements the logic of the algorithm as is mentioned in Chapter 3. It employs Evolving Mean Shift algorithm to perform clustering for the given data. Analysis layer is able to capture the evolving event behavior and compute the keynote changes. It then stores all the information into the databases. meanwhile, All the computed data will flow into the renderer layer to form a real-time visual map for users.

2.4 Data Summarizing Generation Service

It is crucial that we can provider users with most intuitive summaries about each event snapshot during the tracking process. Given the fact that Instagram is heavily image based social network, we think one of the feasible approach is to select one of the user uploaded image to represent a event snapshot. Provided the social network nature of Instagram, it is inherent that we may want to use the most characteristic features that we can extract
from our data set. The underlying assumption is illustrative: people on social network will be more intended to like and comment on the posts that are demonstrative and descriptive; more socially informative images should be highly ranked among the candidate posts to represent an event snapshot.

Data summarizing generation service is primarily a machine-learning based service to select such informative images to describe the event snapshots. To simplify this problem, we trained a RankSVM model for better ranking. We selected \#likes, \#comments, \#timedifference, \#geodistance to construct our feature space. In short, we picked 700 posts in the \#nycmarathon2015 event and labeled each images (if available) with a related score in a scale of \([-1, 0, 1]\). The methodology behind this method is given in the study Pedregosa et al. After RankSVM is trained, the ranking score (Kendall tau) is calculated to select the best candidate for each event snapshot.

2.5 Data Monitoring Service

While we processed our data, we discovered a fundamental challenge that affects the quality of our summary selection. Because we are using Instagram’s subscription API as the crawler method, the posts pertinent to certain event does not contain sufficient social information for our summary selection service to perform. Taking \#NYCMarathon2015 as an example, we witnessed very few social information in the data stream by the time it arrived at our system and thus the summary generation for each event detected and tracked was nearly at random, neither was it convincing. To overcome this situation, we developed a Data Monitoring Service which will re-collect the social information for particular Instagram posts in hourly manner. More specifically, once the event snapshot is generated, data monitoring service will package all the relevant data points related to the current event snapshot as a bundle and store that in a data bundle pool. For every 2 hours, a Crontab Python process will wake up and re-crawl all the data bundles in the pool and updates the corresponding social behaviors by inquiring the Instagram API service the most concurrent status. Data summary will be re-generated and updated from the most concurrent social features.
2.6 Renderer Layer

Render layer provides a visual feedback to users. It employs Google Map API to precisely pin the location of analyzed data onto the map. User is able to see the pined location for a specific event. When clicking on the pin, user is able to view more detailed spatiotemporal information including geographical coordinates and time stamp. Figure 2.2 and Figure 2.3 denotes the front end show case for our renderer layer. The major purpose for our renderer layer is to design a good interface for user to interact with the most concurrent data stream. Thus, users are able to view the most updated data stream from Instagram posts, as well as the geo-location for each posts if available. On the other hand, renderer layer is also responsible to illustrate users with our analysis result passed back from Data Processing layer. Users should be able to view the event trace for each specific event. Also, we provided a click-based interaction with users to view more detailed information in each event snapshot.

(a) User interface to view the Instagram Data Stream  
(b) Visualization on the map for updated Instagram posts

Figure 2.2: UI for renderer Layer
In this section we aim to discuss the algorithms used in our system, more explicitly, the algorithm that is deployed in the analysis layer.

Before we illustrate our algorithm, we first define what “event”, “data observation” and “social network work data stream” as follows:

**Definition 3.1 Event**: Events in social network are real world happenings with time and space information that may be observed by user and reflected on the data in social network.

**Definition 3.2 Data Observation**: Data observations on social networks are users’ input to social networks which include their observation of real world happenings. In the case of Instagram, data observations refer to users’ posts which include textual or tag indicating the events that they pertain to.

**Definition 3.3 Social Network Data Stream**: Social network data stream is a collection of data observations which is transferred from social network website in a data stream fashion.

### 3.1 Problem Formulation

Our objective is design and develop an effective approach such that (i) it is able to identify and detect real world events from social network data stream, (ii) it is able to track and trace real world events from social data stream. (iii) it shall be light weighted so it can avoid the burden of large processing time and heavy system resources usage, yet it needs to be robust and effective.
3.2 Algorithm Analysis

We investigated on existing algorithms that might fit into this scenario. In this section we provide some analysis about various existing algorithms and how they are related to our project.

3.2.1 Social Network Detection Algorithms

Charu et al [1] proposed a clustering algorithm that is able to detect events in social streams, which employs a clustering based algorithm to incorporate both textual and temporal information. It adopts a tf-idf like approach to calculate the similarity between a data point and the clusters. Xiaowen et al [2] explore the properties of wavelet transform, to automatically handle the interaction between temporal and spatial scales. Though demonstrating a very good result in Twitter data, the mentioned social network detection algorithms won’t work in our scenario for two major reasons. Both algorithms require the witness of the entire data set in order to perform the detection algorithms, which is too heavy weighted for our system. For instance, our collection instagram posts pertain to #paris itself has a size of 356713 entries and will consume around 700 megabytes of data. Processing them at the same time will be way too resource consuming for our system. On the other hand, the data type in Instagram and Twitter varies largely. Twitter enforces no geographical and temporal constraint for a user to post data relevant to a specific event. For example, a New York Twitter user may positing his comment about 2012 World Cup in Brazil even after 3 years later. Instagram posts focuses more about real-time information sharing. People will be more likely to upload a picture indicating the real time happenings around them in this scenario due to Instagram’s real time photo sharing mechanism. [3][4].

3.3 Intuition

During our investigation, we observed two major patterns for spatiotemporal event detection and tracking. The objective is achievable by either using statistical modeling or clustering method. Statistical modeling tries to interpret
the data from a random process generation [5]. Thus, the major challenge is to estimate the hidden parameter using various approaches. Clustering, on the other hand [1], is used to track or detect social events by trying to cluster data on time and space scale. With the awareness of potential huge resources that might be used for statistical method, we explore the opportunity of adopting a clustering algorithm that is light weighted, efficient that is capable to fulfill the task of event tracking and detection without consuming too much system resources.

3.4 Evolving Mean Shift Clustering Algorithm

In this section we proposed our Evolving Mean Shift Clustering Algorithm for the purpose of event detection and tracking on social network.

3.4.1 Mean Shift Clustering

As is discussed in [6], Considering an input data set $X = \{x_i\}$ in dimension $d$ and a density kernel $K$, a Mean Shift clustering of $X$ is obtained as follows:

For every point $p_i$, initialize $x^0_i = x_i$ and iteratively compute $x^{k+1}_i$ from $x^k_i$ by performing a gradient ascent of the density kernel. Upon convergence $x^{\infty}_i = \bar{x}_i$ is a local maximum of the density kernel. Points of $X$, which converge towards the same local maximum are then clustered together. It should be noted, that the underlying geometric structure of the clusters is of course dependent on the kernel that is used. In particular, changing the bandwidth of the kernel results in more or fewer local maxima of the resulting density kernel.

More specifically, for data points from $\{x_0, x_1, ....x_n\}$ in dimension $d$ and a density kernel $K$, the multivariate kernel density estimate obtained with kernel $K(x)$ and window radius $h$ is:

$$f(x) = \frac{1}{nh^d} \sum_{i=0}^{n} K(\frac{x - x_i}{h}) \quad (3.1)$$

For radially symmetric kernels, it suffices to define the profile of the kernel $k(x)$ satisfying:
\( K(x) = c_{k,d} k(\|x\|^2) \) (3.2)

where \( c_k, d \) is a normalization constant which assures \( K(x) \) integrates to 1. The modes of the density function are located at the zeros of the gradient function \( \nabla f(x) = 0 \).

The gradient of the density estimator (3.1) is

\[
\nabla f(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=0}^{n} (x_i - x) g(\|x - x_i\|/h) \\
= \frac{2c_{k,d}}{nh^{d+2}} \left( \sum_{i=0}^{n} g(\|x - x_i\|/h) \left( \sum_{i=0}^{n} x_i g(\|x - x_i\|/h) \right) - x \right) \tag{3.3}
\]

The first term is proportional to the density estimate at \( x \) computed with kernel \( G(x) = -c_{k,d} g(\|x\|^2) \) and the second term

\[
\text{min}_h(x) = \left( \frac{\sum_{i=0}^{n} x_i g(\|x - x_i\|/h)^2}{\sum_{i=0}^{n} g(\|x - x_i\|/h)^2} - x \right) \tag{3.4}
\]

is the mean shift. The mean shift vector always points toward the direction of the maximum increase in the density. The mean shift procedure, obtained by successive \( x_i^{k+1} = x_i + m_h(x^k) \).

### 3.4.2 Classic Mean Shift Analysis

In our research project, Adopting Mean Shift clustering algorithm faces two fundamental challenges:

1. Mean Shift clustering algorithm treats all the data objects in the data steam same, ignoring the time stamp when the data object arrives. Therefore, at any time stamp \( t_k \) where \( 0 \leq k \leq T \), the result produced by Mean Shift will be clusters which group up all the available data points. This is not the case especially in social network data stream. Data object \( d_i \) where \( i << t \) should impose less impact to our clustering result at time stamp \( Ti \). For instance, when we are tracking the related events to Boston Marathon, there will be posts before the event start, when Instagram users are posting
pictures and posts about how they are preparing for the incoming marathon. If we treat all the data observations as same, these posts will largely impact the final accuracy.

(2). As described so far, the Mean Shift algorithm is too slow to be used for social data stream clustering. This mainly involves two aspects: (1). The computational time of Mean Shift algorithm is $O(\sigma n^2)$ where $\sigma$ is the number of iterations used for Mean Shift to converge and $n$ is the number of data observations. In our case, number of data observations can be really huge. Take u2 concert tour as an example, we crawled total of 201397 posts during the given time interval Running time to apply Mean Shift algorithm on the entire data set will cost 2 minutes 35 seconds. While most of the speed up Mean Shift algorithms aim to find adoptive . (2) No intermediate results are stored during the computation of Mean Shift algorithm. Even though that Mean Shift is an iterative algorithm that tries to find the local density peak for data observation $d_i$ in each iterations. None of the information is stored and visible to the final user. Thus, in order to see the result, user must re-run the algorithm.

In our project, we aim to solve the above two challenges by proposing the Evolving Mean Shift algorithm that will fit into our research scenario. We want to use the Mean Shift clustering algorithm in a way that it is light weighted with temporal awareness. Moreover, we want our algorithm to run in a continuous manner, avoiding the burden of re-running the algorithm multiple times.

3.4.3 Evolving Mean Shift Algorithm

In this section we illustrate our algorithm used in the research project and provide a detailed walk through for the implementation of the algorithm.

1).Overview: Mean Shift is a density based clustering technique, with an additional bandwidth $h$ which is used to described how closely clusters are expected. In general, Mean Shift is regarded as a natural multi-scale clustering algorithm which is widely used in image segmentation, image tracking, etc.

Consider an input data stream set $X = \{x_i\}$ in dimension $d$ and a kernel density estimate $K$, Mean Shift clustering algorithm performed on $X$ is
obtained as follows: We maintain a sliding window of size $w$ and only perform analysis on the data observations $\{d_i, d_{i+1}, ... d_{i+w-1}\}$ with mean shift algorithm to produce clusters $\{c_i, c_{i+1}, c_{i+2}, ...\} \in C$, where $c_i$ represents the local maximum in the density feature space. Each $c_i$ also maintained a Tree-HashMap based data structure to store all the data points belong to it. At any time, any existing clusters $\{c_i, c_{i+1}, c_{i+2}, ...\} \in C$ are updated with the latest data observations from the sliding window by removing out-dated data observations and refreshing its new center location. Then we apply the standard Mean Shift algorithm to re-cluster and re-compute the clusters. This is equivalent to computing the Mean Shift over the input. But instead applying Mean Shift algorithm to all the dataset, it will only try to compute and updated cluster information based on the updated data observations that are passed by the sliding window. Also, we also passed Although extremely simple, this strategy proved to be robust and efficient during our experiments. It also allows us to run Mean Shift over infinitely growing data sets with bounded memory, as long as the range of the data remains bounded, which is a required property in our scenario.
In this section, we demonstrate that our algorithm will be able to perform event detections.

Definition 4.1 Static Data Observations: Static data observations refer to the social network data observations that are related to a target that is not changing its location over time. For instance, any posts that have location hashtag #paris and #london are treated as static data observations because location will not move over time.

When we perform our algorithm on a social network stream composed of static data observations, we are eventually capturing all the major events that are related to this static target (with geographical and temporal information). One of the application, as shown below, is to detect events that happen in a specific area.

For the experiment, we have been crawling all the Instagram stream data with hashtag #paris starting from February 2\textsuperscript{nd} 2015 to December 3\textsuperscript{rd}. We assume that all the Instagram posts with hashtag #paris in its post text body are the human observations that is related to Paris.

When examining the data set, we notice that despite the fact that there are various kinds of the posts on Instagram, most posts will happen in central Paris, especially around some tourist destinations such as Eiffel Tower, Muse du Louvre, etc. When performing Evolving Mean Shift algorithm on it, we will get very static clusters with centers around these locations. However, on the evening of 13 November 2015, a series of coordinated terrorist attacks occurred in Paris and its northern suburb, Saint-Denis, our system detected a significant change when processing the incoming data stream.
4.1 Event Detection for Paris

Figure 4.1 shows that event detection for Paris on a normal day. We also provide a tourist map provided by Google Map for comparison. Because Evolving Mean Shift share the same nature with classic Mean Shift, which tries to formulate clusters which best fits into the data. Many posts around central Paris will be merged into one clusters based on their comparatively longer distance to other clusters. It is easy to observe that cluster centers which represent events are located where Les Invalides, Eiffel Tower along with other famous tourist destinations.

![Evolving Mean Shift Event Detection Results](image1)
![Google Map Tourist Guide](image2)

Figure 4.1: Paris Event Detection on November 12th

![Evolving Mean Shift Event Detection Results](image3)
![Paris Terror Attack Location](image4)

Figure 4.2: Paris Event Detection on November 13th (Terror Attack)

As is shown in Figure 4.2, on November 13th, our system detected a booming trend of posts around the attacked area. Two new cluster were formed with summaries about the surge of data observations near the Stade de France and Rue de la Fontaine-au-Roi. We would like to note that for Rue de Charonne, Rue de la Fontaine-au-Roi and Boulevard Voltaire are very close from a geographical perspective, it is naturally that they will be combined into one cluster due to high spatiotemporal similarity.
This is a very meaningful empirical experiment because we showed that our system can serve as a good tool for users to find most real-time feed other than the transitional media press. Due to the real-time nature of Instagram, our system is able to respond and react in a shorter time for the any emergency during our process of static event detection.

More specifically, we demonstrate that due to the multivariant and non-parametric nature of Evolving Mean Shift Clustering algorithm, our system is only able to capture the most impactful events based on users’ online social behaviors, it also adapts to any random and emerge data changes and can provide visual feedback to users in a real time fashion.
5.1 Event Tracking for Marathons

In this section, we demonstrate our project is able to track specific events that are changing locations as time passing by. We crawled data for hashtags #bostonmarathon, #nycmarathon and #u2 to tracking the respectively Boston Marathon on October 1st, 2015 starting from 6am to 4pm, New York City Marathon on November 1st, 2015 from 6am to 4pm and U2 concert tours from May 1st to December 1st, 2015.

Figure 5.1: Event Tracking for Boston Marathon 2015

Figure 5.2: Event Tracking for NYC Marathon 2015

In Figure 5.1 and Figure 5.2 we compared our result for the marathon
event tracking with the marathon course maps given respectively by Boston Marathon and New York City Marathon official websites. We observed that our algorithm was able to predict the overall movement of the marathons with regard to the time. We compare only the course inside city for very practical reasons: both marathons started in the suburbs of Boston and New York City, and proceeded along highways which are designed specifically for runners. Before they entered the cities, no posts were detected by our system due to the lack of observers alongside. We think it is still very meaningful to compare the marathon courses just within the cities since no algorithms will be able to track the events without any data observations reported.

Although there exists certain amount of noisy Instagram posts, these examples demonstrate that our system is able to detect and track events. We also want to note that our prediction of the event courses is off the truth data by a small amount of range not only because of the noisy data but also because of the diverging population density in different areas. For Instance, our prediction of the NYC marathon was largely affected by a large amount of data observations in New York Central Park, where naturally there will be more people willing to share the event and upload it to Instagram, rather than the Bronx, where there are few residents and visitors and almost no data observations. Thus our cluster centers are dragged into Central Park area at all times. For Boston Marathon, the course starts and ends at a comparatively evenly distributed area with regard to the resident and visitor population. Thus, our system is able to predict a more precise and accurate tracking result.

5.2 Event Tracking for U2 Concert

In Figure 5.3 and Table 5.1 we illustrated our event tracking for U2 concert tours. We extracted several snapshot from our tracking algorithm and the concert tour route that is published on U2 official website. Although the concert happened on a much larger temporal and geographical scale, our System was able to track it very accurately.
As a matter of fact, during May 18th 2015 to December 1st, U2 has been holding 73 concerts world wide. Our algorithm was able to trace all of them. Figure 5.8 and 5.9 demonstrate a experiment result for the comparison for our event tracking prediction and the truth data.
As is shown in Figure 5.8 and Figure 5.9, our prediction of event tracking differs in a very limited scale both temporally and geographically. With regard to distance, our prediction is off by the truth data in the range of $[0.53, 8.12]$ miles. With regard to time, our prediction error is in the range of $[0.2, 7.5]$ hours.
6.1 Synthetic Data for More Experiment Result

In chapter 4 and chapter 5 we demonstrate that our system has a relatively good performance on the empirical experiment data. But for better measurement of our system and algorithm, we may need synthetic data to perform experiments on.

This is mainly due to very practical reasons: For many events in real world happenings, there is lack of accurate data available to test the accuracy. For instance, for the Boston marathon the marathon runners are proceeding through a certain area but in different time stamps based on the individuals’ running speed. It is indeed very hard to argue whether the event happens in a specific area at a specific time. While the leading champion passes the finish line, the remaining runners may still be in the process of marathon in the middle of the course of marathon.

Thus, in order to perform a better measurement, we may need to apply synthetic data to validate our system and algorithm. Finding the right model to generate such data is a interesting yet difficult problem. We would like to find an good statistical model which can be extracted from the data set we have and can be a comparatively good representation of the existing data observations in order to make the synthetic data experiments meaningful. Some researches [3] has shown that CSR, Complete Spatial Randomness will be a good property to be measured. CSR asserts that the spatiotemporal data to follow a homogeneous Poisson point process, which implies that the observation of social media data observations can follow a Poisson distribution distribution. Although [3] shows an assessment that the distribution of Twitter data set follows such a pattern, it remains unknown whether this is also true for Instagram data set. The need to validate such properties
remains as a part of future works that needs to be done.

One the other hand, many spatiotemporal models have been proposed including various statistical models to simulate various spatiotemporal events. We will still investigate which of them will be a best representation of our Instagram data set in order to generate a good synthetic data to assess the quality measurement of our algorithm.

6.2 Exploration of Wireless Sensor Network Tracking Algorithms on Existing Data

Even though our system focuses mainly on the real-time data, it is very intrigue and interesting that how traditional wireless sensor network tracking can be applied in this scenario. This remains a open area where no previous research has been done to the best of our knowledge. We are actively employing some of the existing wireless sensor network algorithms to our current data set. Note the object is not to process it at real time but rather have a more accurate way to describe the the social network signals in a offline manner.

6.3 System API and Integration

Currently, we are actively developing APIs for our system. We would like to provide users with their own choice about what event they want to track. The APIs layers will be through both the Renderer Layer and Data Analysis Layer as is discussed in Chapter 2. We also would like our system to be adaptive to users’ input and can generate real-time feedback for the events that users are most interested in.
In this research project, we have proposed a novel approach for spatiotemporal events detection and tracking in social networks. More explicitly, we developed a system that can automatically crawl, analyze and render Instagram event data. We also propose a light weighted yet robust Evolving Mean Shift algorithm which will work in a streaming social data environment.

We achieve the acceleration of classic Mean Shift algorithm by adopting a sliding window to enforce the temporal constraint and to reduce the system computational cost and by caching intermediate result for swift updating and re-computing the cluster result.

We demonstrate that our system fulfills its functionality to detect and track events in our empirical experiments. In terms of real-world applications to the domain of real time detection and tracking for social network media, especially real-time photo sharing social networks like Instagram, a working prototype was developed. The research presented in this study can serve as a basis for any further investigations.
REFERENCES


