


# What Is New in Our City? A Framework for Event Extraction Using Social Media Posts

Chaolun Xia<sup>1</sup>, Jun Hu<sup>1</sup>, Yan Zhu<sup>1</sup>, and Mor Naaman<sup>2</sup>

<sup>1</sup> Rutgers University, 57 US Highway 1, New Brunswick, NJ 08901, USA  
cx28@cs.rutgers.edu

<sup>2</sup> Jacobs Institute, Cornell Tech, 111 8th Ave., New York, NY 10011, USA

**Abstract.** Post streams from public social media platforms such as Instagram and Twitter have become precious but noisy data sources to discover what is happening around us. In this paper, we focus on the problem of detecting and presenting local events in real time using social media content. We propose a novel framework for real-time city event detection and extraction. The proposed framework first applies bursty detection to discover candidate event signals from Instagram and Twitter post streams. Then it integrates the two posts streams to extract features for candidate event signals and classifies them into true events or noise. For the true events, the framework extracts various information to summarize and present them. We also propose a novel method that combines text, image and geolocation information to retrieve relevant photos for detected events. Through the experiments on a large dataset, we show that integrating Instagram and Twitter post streams can improve event detection accuracy, and properly combining text, image and geolocation information is able to retrieve more relevant photos for events. Through case studies, we also show that the framework is able to report detected events with low spatial and temporal deviation.

**Keywords:** Data mining · Social media · Event extraction

## 1 Introduction

With the growing popularity of mobile devices and applications, more and more people are sharing their moments with their friends and the public through mainstream social media platforms such as Facebook (Instagram) and Twitter. A recent report<sup>1</sup> shows that Instagram now has more than 200 Million monthly-active-users (MAU) and these users upload more than 1.5 billion photos and videos per month. Twitter has even larger traffic and popularity, 255 Million MAUs and 15 billion tweets per month.

Although a dominating proportion of posts from such social media platforms are about users' personal life [19], such as emotional feeling, opinions, food, travel and even self-portraits, there are still considerable amount of posts recording

<sup>1</sup> <http://jennstrends.com/instagram-statistics-for-2014/>

what are happening in our city. A user may upload photos of a fashion show to her Instagram account, or talk about emergency or crime on Twitter. These valuable social media posts have made it possible for researchers and developers to accurately detect and present local events in real time. Such techniques will benefit various users, ranging from government officers, journalists, to tourists and residents, etc. For example, a system quickly reporting fire or car accidents can help the local police to make a timely response to the emergency; detecting entertaining events in real time and representing them to nearby tourists or residents can provide opportunities in social engagement.

However, the problem of detecting and representing local events in real time from social media data streams remains challenging. First, event-related social media data are sparse, although the volume of posts from any popular social media platform is large. Second, most current research focuses on Twitter [2], while there are various types of social media platforms which can potentially contribute to the detection problem. However, the problem to choose or combine multiple data sources to detect events is challenging due to the heterogeneity of posts from different data sources. Third, after detecting events, to represent events with the most relevant posts is still challenging due to the noisy posts stream with heterogeneous content including image, text and geolocation.

To address these challenges, we propose a novel framework in this paper. This framework first detects candidate event signals from Instagram and Twitter post streams, and then extract features to classify whether an event signal is a true event or noise. Finally, it summarizes the detected event by retrieving relevant photos and topics and then estimating the occurrence time and location. Besides the proposed framework, our contributions also include that we analyze different methods to integrate Instagram and Twitter post streams, and experimentally show that they improve the detection accuracy. To our best knowledge, we are the first to integrate Instagram and Twitter posts to detect events in real time. For event summarization, we propose a method to retrieve relevant photos, which utilizes image content, text and geolocation information. Finally, we conduct case studies to show that our framework has low spatial and temporal deviation for detected events.

The rest of this paper is constructed as follows. Section 2 reviews the previous works. In Section 3, we formally define the local event detection problem. We introduce the detailed methodology and our system framework in Section 4. We analyze and discuss our experiment results in Section 5, and conclude our work in Section 6.

## 2 Related Work

There have been plenty of research regarding detecting events or news. They can be categorized according to several aspects, including types of events, data sources and methods [2].

Prior to detecting events from social media streams, [12][13][14][16] detect events from traditional media data. As a seminar work to this problem, [16]

uses an infinite-state automaton to model the term frequency in documents, and considers the burst, for example, the significant change in term frequency, as potential events. [13][14] detect events by modeling feature burst with spectral analysis and Gaussian mixture respectively. [12] heuristically identifies bursty terms and then groups these terms to discover potential events. Since all of them use information which has been existing for long time as data sources, their systems can hardly produce events in real time.

The introduction of social media platform brings new opportunities and challenges to the problem of event detection, and plenty of methods have been proposed. Inspired by the idea of detecting bursty feature, EDCoW [33] uses wavelet theory to model the signal of words and capture their bursts to detect events. [28] monitors bursty topics instead of unigram or tweet-segments [21]. Similarly, [7] detects events by discovering trending topics. Modeling trending topics can produce real-time detection, however, it is not applicable to our scenario of detecting and locating events since trending topics are usually weak signals for small scale event, and it has large error to estimate the location of trending topics [1][15][17]. Other than detecting the trending topics, based on influential theories of emotions, [32] automatically assigns a single tweet with an emotional label which is neutral or comes from one of the 6 Ekman’s emotions. Then they monitor the sudden change of tweets’ emotions in countries as the signals to detect events. All of the above works consider burst of certain features as signals of potential events. They model different bursty features including n-gram, terms, topics and emotions, and the common idea behind is absorbed into our framework.

Some detection frameworks are specific-event driven, that is, assigning a specific event type to each detection task. TEDAS [22] was proposed to detect crime and disaster related events from twitter stream. Earthquake center and typhoon trajectory have been successfully estimated in [30]. Besides disasters, [11][20] use twitter posts to detect local festivals by monitoring the movements of crowds. Twitterstand [31] classifies tweets as news and non-news to detect news events. Different from these methods, our proposed framework is not restricted to any event type.

Considering the data source, most of the previous works collect data from Twitter posts [11][20][22]. We put two data collectors in Instagram and Twitter monitoring and collecting useful information from the live post streams from these two social media platforms. Our previous work [35] uses Instagram posts to detect events with high accuracy. Unlike them, in this paper we use the posts from both of the two popular OSNs together to detect events.

After detecting events, retrieving relevant content to represent existing events is a challenging problem. Focusing on Twitter content, [3][5] extracts tweets and topics for known events. [25] generates a journalistic summary of a sport event using status updates from Twitter. Including Twitter, [4] retrieves social media content across YouTube and Flickr for existing events. In [9], photo tags are used to detect events and then retrieve photos based on tags to represent an event. It does not use the rich image content but heavily rely on user generated tags that

are not always reliable [27]. Although the work in [27] combines photo content and tags to detect events, it needs to discover landmarks first and then detect events around the landmarks. To be different, our work does not rely on landmark discovery, thus we can detect more general events. As to retrieving images, most existing methods rank images based on certain similarity measurements to a specific query, a keyword or image. In large scale applications, approximate nearest neighbor algorithms and hashing method [18][24] are widely explored. However, these methods are not directly applicable to our problem since we do not have a specific query. Instead, our query is an entire event that consists of noisy photos, text and geolocations. In our system, we observe that for a true event, the images that are relevant to it usually share common patterns. While other irrelevant images are considered as noise, which are usually independent and randomly distributed.

### 3 Problem Definition

Following [32][35], we define an event as *a real world activity that occurs during time period  $T$  within a geographical area  $L$* . To detect such events in real time, we define a framework as follows. The framework takes the real-time streams of posts from Instagram and Twitter as the input. The system is expected to output detected events in sequence. For each detected event, we extract its related content namely the set of related images, topics (a set of keywords), the estimated occurrence location, and the estimated occurrence time.

### 4 System Framework and Methodology

In this section we introduce our architecture, each component and methods. The system framework is shown in Figure 1. Given a fixed geographical region  $L$  from which we want to detect events in real time, first we distribute event sensors over the entire region. Each event sensor is designed to be independently responsible for discovering events in a single sub-region  $l$ . In other words, we divide the entire region, i.e. New York City in this paper, into  $k$  sub-regions,  $L = \{l_1, \dots, l_k\}$ . For each sub-region  $l$ , we allocate an event sensor, which has three components, Event Signal Discovery, Event Signal Classification and Event Summarization. Although more advanced methods that divide an entire region to sub-regions according to topic distribution [1][17] or population density [20] might improve the overall performance, in this paper we do not focus on this problem, and we divide the entire New York City into  $N \times M$  grids of equal size.

The architecture of our system is shown in figure 1. The Event Signal Discovery component takes the input of Instagram and Twitter data streams in real time and outputs candidate event signals. The Event Signal Classification component takes candidate event signals as input, extracts various features for them, and finally outputs event signals which are classified as true events. The Event Summarization component selects the most relevant content, including photos and text to represent the event. Besides, it produces the estimated occurrence location and time of the event.

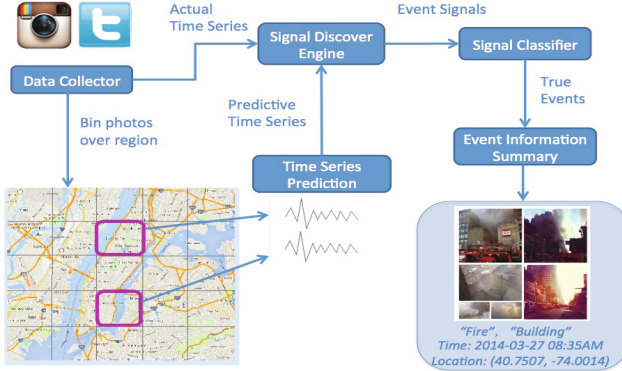


Fig. 1. Architecture of our system framework

#### 4.1 Event Signal Discovery

The Event Signal Discovery component contains 3 sub-components, data stream collector, time series estimator and bursty detector. The motivation behind follows the general idea of modeling bursts [16] of certain features as potential events. Unlike other papers which model the sudden change of emotions [32], the movements of crowds [20] or the trending topics/terms/n-grams [21][33][7], we adopt the method in [34][35] which considers the abnormal increase of social media posts as the potential signal of events. This is because we observe that the change in the number of posts is sensitive to event occurrence, especially to the occurrence of small-scale local events.

The data stream collectors keep collecting posts from Twitter and Instagram in real time. We only collect posts containing geo-location information. In order to find bursty signals, an event sensor monitors the change in the number of posts in a sub-region  $l$ . A time series of the post number is constructed for  $l$ . We use  $t$  to denote the time, and  $v_l(t)$  denotes the post number at  $l$  and within  $t$ . In practice,  $t$  stands for a time period and its window length in our experiments is 15 minutes.

The time series estimator is implemented by Gaussian Process Regressor (GPR) [29]. We use GPR due to its great performance in modeling various time series data such as stock prices [29]. Due to limited space, the detail of GPR is available in our previous work [35].

Once the GPR model is built on the historical data, it is able to predict the number of posts at  $l$  during  $t$ , as  $\hat{v}_l(t)$  for any given  $t$ . When the data stream collector gathers the true number of posts in a sub-region  $l$  at  $t$ , we compare the actual number of posts, i.e.  $v_l(t)$ , with the predicted number of posts, i.e.  $\hat{v}_l(t)$ . If there is a large deviation between these two numbers, this signal is marked as a potential event signal. Following bursty detection, we are only interested in when the predicted number of posts is larger than the actual number of posts. Typically, we define an abnormality score as  $[\hat{v}_l(t) - v_l(t)]/\hat{\sigma}(t)$ .  $\hat{\sigma}(t)$  is the

predictive standard deviation given by GPR. It indicates the confidence of the prediction and a smaller  $\sigma(\hat{t})$  indicates stronger confidence. If the abnormality score in sub-region  $l$  during time  $t$  exceeds a given threshold, the Event Signal Discovery component outputs a candidate event  $e(l, t)$  that stands for the set of all the Instagram and Twitter posts that are posted during time  $t$  and within location  $l$ .

## 4.2 Event Signal Classification

Once Event Signal Discovery component produces a candidate event signal  $e(l, t)$ , the Event Signal Classification component first extracts features from  $e(l, t)$  and classifies it as true or false by a supervised learning model. Since a candidate event signal (shortened as candidate event)  $e(l, t)$  is a set of Instagram and Twitter posts bounded by location  $l$  and time  $t$ , we can extract various types of features from them. Based on these features, the classifier determines whether  $e(l, t)$  represents a true event or not. Note that, even if there is an event at location  $l$  and time  $t$ , not all the posts in  $e(l, t)$  is related to that event. Thus we will choose relevant posts to represent the event which is discussed in Section 4.3. At this step, we focus on extracting robust features from the Instagram and Twitter post streams.

**Feature Fusion.** Before design specific features for candidate events, we first model the fusion of Instagram and Twitter posts. We previously assume when the number of total posts (including Instagram and Twitter) bounded by location  $l$  and time  $t$  suddenly increases, some event  $e(l, t)$  may happen. However, we do not know which data source, Instagram or Twitter, records this event, or both. This is caused by the heterogeneity of Instagram and Twitter posts and users. Although they are both popular social media, their users have different habits and interests. Instagram is more about recording personal life and daily activity while Twitter is considered as an influential news media [19]. Thus, it is expected that some events are recorded by only one data source while some are recorded by both. We can either extract features from Instagram posts or Twitter posts only, or from both of them. In this paper, we consider two methods to fuse two data sources for feature extraction and classification.

The first fusion method is to integrate Instagram and Twitter posts at data level, i.e. before feature extraction. In this way, we need to consider a Twitter post and a Instagram post as homogeneous. For each event signal  $e(l, t)$ , we extract its features vector  $\mathbf{x}_e$  from all the posts during time period  $t$  within location  $l$ . This method mitigates the sparsity problem of geo-tagged posts, and it is expected to benefit the classification of small-scale events with a few of posts in total.

The second method is to integrate Instagram and Twitter posts at feature level, i.e. after separate feature extraction. We extract feature vector  $\mathbf{x}_e^I$  from Instagram posts and extract feature vector  $\mathbf{x}_e^T$  from Twitter posts respectively, and then concatenate them to form the final feature vector  $\mathbf{x}_e$ . Note that by this method, the size of feature vector  $\mathbf{x}_e$  is nearly doubled compared to the first

method. The benefit of this method is that, we can extract different features from Twitter and Instagram, and further incorporate other inhomogeneous data sources.

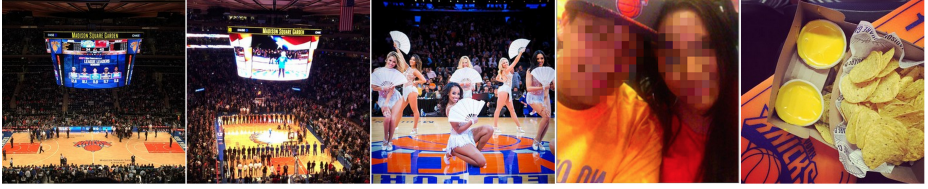
**Feature Extraction.** To represent an event signal  $e(l, t)$ , we extract four types of features from all the posts bounded by  $l$  and  $t$ , namely topic features, emotional features, spatial features and social features. Formally, we use  $P_e = \{p_1, \dots, p_n\}$  to denote the set of posts associated with the event  $e(l, t)$  and  $n = |P_e|$ . Note that, here we do not extract feature from a single post, instead, we extract features from the set of posts  $P_e$  associated to event signal  $e(l, t)$ .

First we extract five topic features from posts’ text, i.e. photo captions and/or tweets. We first build a background topic distribution  $\theta_B$  for location  $l$ . We use word unigram language model to represent the topic distribution of the background posts, i.e. all the posts during last 24 hours within  $l$ . We also build the event topic distribution  $\theta_E$  for all the posts in  $P_e$  in the same way. We calculate (1) the total number of words that are in  $\theta_E$  but missing in  $\theta_B$ . A novel word, which has never appeared before, may indicate something new. We calculate (2) the average KL-divergence  $[KL(\theta_B||\theta_E) + KL(\theta_E||\theta_B)]/2$ . We expect that the topic distribution changes when there are true events. We also compute (3) average number of hashtags in  $p_i$ , (4) the average text length of  $p_i$  and (5) the average frequency of the 3 most frequent words in  $P_e$ .

The second type is emotional features. Inspired by [32] in which the authors experimentally prove when there are large event occurring, user emotions on Twitter change. In order to capture emotional changes, we compute the number of emotion-related punctuations and words from  $P_e$ : (1-2) the number of exclamations and question marks respectively and (3-8) the total number of words from  $P_e$  categorized to each of the six Ekman’s emotions [32] respectively. Similar to topic features, we construct a background emotion-related word and punctuation count vector  $E_B$ , and take the deviation between  $E_B$  and the (1-8) features as the (9-16) features, indicating the change of emotion with location  $l$  and time  $t$ .

The third type is geolocation features. They are (1) mean and (2) standard deviation of pairwise post geo-distance, i.e.  $dist(p_i, p_j) \forall p_i, p_j \in P_e$ ; and (3) the entropy [35] of the spatial distribution of all posts in  $P_e$ . The intuition behind these features is that we observe that when there is an event, the event-related posts tend to form a cluster. Similarly, we also compute these features from the background posts, and take the corresponding difference from (1-3) features as (4-6) features.

The fourth class includes a social feature. We compute the average number of mentioned users, i.e.  $@Alex$  of all posts in  $P_e$ . We finally extract 28 features of four categories in total. We also normalize the topic and emotional features by text length.



**Fig. 2.** Five sampled Instagram photos from a detected Knicks NBA game event in NYC. From journalists’ perspective, the first three images are considered representative to summarize the event. Although the last two images were uploaded at the stadium and their captions are also about game, they are not informative for describing this event. Typically, the forth photo has a user privacy issue.

### 4.3 Event Summarization

In this section, we introduce our methods to summarize a detected event. A candidate event signal that consists of a number of Instagram and Twitter posts bounded by time period  $t$  and location  $l$  is classified as a true event or not. Provided that the classifier in Section 4.2 judges an event signal is a true event, we still do not know what the event is, a concert or a car crash, because the event classifier in this framework is designed to be general, i.e. independent of event type or scale.

Therefore we summarize an event from 4 aspects: topics, photos, occurrence location and occurrence time. Extracting topics from user generated posts is well studied [3][4][5][8]. Thus it is not our focus in this paper, and we use existing methods to select keywords from tweets and photo captions as the topics of an event. Besides, it is straightforward to estimate the occurrence location and time of an event in our framework. Due to their simplicity, the methods are discussed together with performance in the experiments, Section 5.4. Here we only cover the method we proposed to retrieve relevant photos for a detected event. Since tweets are seldom associated with photos, we only retrieve photos from Instagram posts to represent an event.

We need to retrieve photos because not all the Instagram post bounded by time period  $t$  and Location  $l$  are related to that event. For example, an Instagram photo was uploaded near a fire accident event, but the image content is about beers. Besides, we observe that users frequently upload self-portraits or food in events, which are not helpful for other users to understand the event. Moreover, some photos involve user privacy issue as shown in Figure 2. Therefore, we need to select relevant and representative photos to visually summarize an event. For simplicity, we name them **event-related** photos.

Our proposed method is based on the following observations. We observe that event-related photos usually share similar image content. For example, photos related to a fire accident usually record smoke, fire or the police. We also observe similarity of text associated to event-related photos. For example, users are likely to use “fire” or “smoke” to describe the photo related to a fire. Besides image content and text, most events occur in a fixed place, such as NBA matches, thus



event-related photos tend to geographically form a cluster in the event center. Different from event-related photos, we observe that, for most noisy photos, their image content and text share very limited similarity to each other, and they are not necessarily close to the geographic center. However, we also observe outliers. For example, a user uploads a photo of her food during a NBA game and write a caption “A wonderful NBA game! # Knicks!” to the photo. In this example, when we compute the relevance score of the photo by only considering its geolocation and/or text, we find many popular text algorithms consider this photo highly relevant to the event. But when we compute its relevance score based on image content, the relevance diminishes.

Inspired by the above observations, for each photo  $x$  in an event  $e$ , we individually compute the image content relevance score (to the event  $e$ ) given the image content of  $x$  only, as  $s_c(x, e)$ , the text relevance score given the text of  $x$  only, as  $s_t(x, e)$ , and the geolocation relevance score given the geolocation of  $x$  only, as  $s_l(x, e)$ . Note that, here  $P_e$  denotes the set of photos associated to event  $e$ , and thus  $x \in P_e$ . After that, we linearly combine the three individual relevance scores into the finalized relevance score  $s(x, e)$  in Eq (1) that denotes how the photo  $x$  is overall relevant to the event  $e$ .

$$s(x, e) = a_c s_c(x, e) + a_t s_t(x, e) + a_l s_l(x, e) \quad (1)$$

$a_c$ ,  $a_t$  and  $a_l$  are the weights for the three independent relevance scores. Conventionally, we specify  $a_c + a_t + a_l = 1$  and  $a_t, a_c, a_l \geq 0$ . The weights are the marginal effects of individual relevance score contributed to the overall relevance score. Intuitively, the larger a weight is, the larger positive impact that the corresponding single relevance score has on selecting event-related photos. Since we model retrieving event-related photos as an unsupervised ranking problem, the choices of  $a_c$ ,  $a_t$  and  $a_l$  are discussed through experiments. We introduce the models to compute the three individual relevance scores as follows.

**Image Content Relevance Model.** To compute the image relevance score function  $s_c(x, e)$ , we use color histogram and GIST features [26] as image descriptor. These two image features are known for effectively describing discriminative scene characteristics. Our image relevance ranking method is adapted from an unsupervised image outlier removal method [23].

For a detected event  $e$ , its corresponding posts set is denoted as  $P_e = \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbb{R}^d, i = 1, 2, \dots, n\}$ . Since each post is always associated with an image, here we use the same notation for a post ( $x$ ) and its image ( $\mathbf{x}$  in vector space). For each image  $\mathbf{x}_i$ , we learn a scoring function  $s(\mathbf{x}_i)$  to manifest its relevance to the event:

$$\min_{s \in \mathcal{H}, y_i \in \{t^+, t^-\}} \sum_{i=1}^n (s(\mathbf{x}_i) - y_i)^2 + \alpha s^T L s - \frac{2\beta}{n - n^-} \sum_{y_i > 0} s(\mathbf{x}_i) \quad (2)$$

Note that, the value of  $s(\mathbf{x}_i)$  is exactly the image relevance score  $s_c(x, e)$  in Eq (1).  $n^-$  is the number of posts which is considered as irrelevant to the events,

$L$  is the graph Laplacian matrix, computed from the  $k$  nearest neighbor graph. We construct the neighborhood graph  $G$  by defining the affinity matrix  $W$  as:

$$W_{ij} = \begin{cases} \exp(-\frac{\text{dist}(x_i, x_j)}{\sigma^2}), & \text{if } \mathbf{x}_i \in \mathcal{N}_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathcal{N}_k(\mathbf{x}_i) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$\text{dist}(\cdot)$  is the Euclidean distance,  $\mathcal{N}_k(\mathbf{x}_i)$  is the set of  $k$ -nearest neighbors of  $\mathbf{x}_i$ , and  $\sigma$  is the bandwidth parameter.  $L$  in Eq (2) is the graph Laplacian matrix of  $G$ , computed as  $L = D - W$ , where  $D$  is a diagonal matrix with diagonal elements defined as  $D_{ii} = \sum_{j=1}^n W_{ij}$ .  $\alpha$  and  $\beta$  are two model parameters balancing the regularization of graph Laplacian term and the effect of pushing the average positive example away from the margin.

During optimization, we do not have any label supervision on whether an image is event-related or not, therefore we are essentially solving an unsupervised learning problem:  $y_i$  is unknown in our optimization problem. As suggested by Eq (2), we treat  $y_i$  as a variable which is softly labeled as  $t^+$  or  $t^-$  during optimization. Following the experiment results in [23], we dynamically set  $(t^+, t^-)$  as  $(\sqrt{\frac{n^-}{n-n^-}}, \sqrt{\frac{n-n^-}{n^-}})$ , where  $n^-$  is updated in each iteration. Eq (2) is minimized by alternating optimization: iterating between fixing  $y$  to minimize  $s$  and fixing  $s$  to minimize  $y$ , until convergence. The first subproblem, fixing  $y$  to minimize  $s$  is achieved by solving a constrained eigenvalue problem with a closed form solution. The other subproblem, fixing  $s$  to minimize  $y$ , is achieved via sorting and sweeping cut to find an optimal threshold. Throughout our experiment, the similarity between any two posts is measured in Gaussian kernel space. Finally, we use the score  $s(\mathbf{x}_i)$  as the image content relevance score for images  $\mathbf{x}_i$ , i.e.  $s_c(x, e)$ .

**Text and Relevance Model.** We directly use the method in [3][5] to compute the relevance score of a photo’s text to the event. For each photo’s text, we represent it by a character  $n$ -gram language model where each photo’s text is converted to a large and sparse vector. Then we compute the textual centroid  $\mathbf{c}_t$  of these photos as  $\mathbf{c}_t = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$  where  $\mathbf{x}_i$  denotes the  $i$ -th photo’s text (in vector space) of the event. According to [3][5], the text relevance of a photo  $\mathbf{x}$  to the event  $e$  could be computed by the closeness of  $\mathbf{x}$  to the centroid  $\mathbf{c}_t$ . Thus, we compute  $s_t(\mathbf{x}, e)$  as the cosine similarity between  $\mathbf{x}$  and  $\mathbf{c}_t$ .

**Geolocation Relevance Model.** Each photo is associated with a coordinate  $(u, v)$  which denotes its latitude and longitude respectively. Similarly, we compute the geographical centroid  $c_l$  of the event as  $(\frac{1}{n} \sum_{i=1}^n u_i, \frac{1}{n} \sum_{i=1}^n v_i)$  where  $u_i$  and  $v_i$  respectively denote the latitude and longitude of the  $i$ -th photo of the event. Thus, the geolocation relevance score  $s_l(x, e)$  can be computed by the earth surface distance between  $x$  and  $c_l$ .

**Interpretations and Advantages.** In the above method, we extract image, text and geolocation features from a photo and the event to compute the three

relevance scores separately, and finally linearly combine them into the overall relevance score. An alternative method is not to compute the individual relevance scores separately. Instead, it extracts the same image, text and geolocation features but concatenates all the three feature vectors into a longer feature vector, and then apply a unified model to assign relevance score to the photo given its event. However, such a unified model has problems caused by the heterogeneous characteristics of image, text and geolocation information. First, the dimensions of the three feature vectors are largely different. An efficient text representation is  $n$ -gram model which transfers a piece of text to a large and sparse feature vector. However, the geolocation information is efficiently represented as a feature vector in two dimensions only. Thus, if we just simply concatenate them without robust feature selection, the geolocation information is easily overwhelmed in such feature space. Second, the hypotheses of relevance (or similarity or closeness) are semantically different in the three aspects. In modeling the relevance of image content, many previous researches find the similarity between photos defined in Gaussian kernel space is proper. While in modeling geolocation closeness, the earth surface distance is naturally the best. Thus, it is not ideal to model all the three types of information in a unified distance space.

## 5 Experiments

In this section, we first introduce the dataset and parameter setting. Then we evaluate event detection accuracy by Instagram and Twitter post streams. We also evaluate the event-related photo retrieval. Finally, we sample detected true events to evaluate the temporal latency and spatial accuracy by case studies.

### 5.1 Dataset and Setting

We use Twitter APIs and Instagram APIs to crawl geo-tagged posts in New York City. Each crawled Instagram post (shortened as photo) is associated with an image, a text, a pair of coordinates, created time and other information. Each crawled tweet is associated with a non-empty text, a pair of coordinates, created time and other information. From 2012-12 to 2014-06, we collected 12,453,448 geo-tagged tweets and 31,188,195 geo-tagged photos.

**Event Classification Annotation.** We use crowdsourcing to accomplish this labeling task: given an event signal  $e = (l_e, t_e)$  and its associated posts, it is labeled based on whether there is a true event during time period  $t_e$  within location  $l_e$ . We first used Amazon Mechanical Turk to label the discovered event signals and then invited three journalists from a local newsroom in New York city to calibrate the labeling to guarantee our dataset is as correct as possible. We sampled 1945 events signals with associated posts to label. As a result, we get 1084 events signals with valid and confident labeling. Among them, 477 events signals are labeled as true events while the other 607 are labeled as false events (noise).

**Image Relevance Annotation.** To evaluate our proposed relevant photo retrieving method, we randomly select 153 true events which contain at least 8 photos<sup>2</sup>. For each event, we label all its photos (if the number of photos in an event exceeds 35, we sample 35 photos). We tried to use a third-party crowdsourcing to label, but find out their correctness is largely below our expectation. Therefore we trained an independent user, and ask the user to label whether a photo is relevant to a given event based on these criteria, 1 for relevant, 0.5 for partially relevant and 0 for irrelevant. For example, we consider self-portraits and food as irrelevant. We also give the user the location and time information and topics of the event to facilitate labeling. On average, an event has 24.7 photos, and 39.4% of its photos are labeled as relevant, 5.8% are partially relevant and the rest, i.e. 54.8% are irrelevant.

**Parameter Setting.** To monitor the entire New York City, we divide NYC into  $25 * 25$  sub-regions (0.45 square kilometers for each geo-region). We also turned the window size  $t$  in Gaussian Process Regressor to be 15 minutes. A reasonably long time interval will lead to large detecting latency while a tiny interval will cause the decrease of the detection accuracy since there may be very few posts during a tiny time interval. The experiments on the choice of these parameters are in our previous work [35].

## 5.2 Detection Accuracy

In this section, we evaluate the performance of Event Signal Classification. Before extract text-related features, we preprocess posts’ text by NLTK [6]. We remove stopwords, non-English characters and urls. We also separate capitalized and concatenated words, such as from “ILoveThisGame” to “I”, “love”, “this” and “game”. Then we use 10-fold cross-validation to evaluate the effectiveness of feature extraction and fusion with standard classifiers. To avoid the variance caused by different classifiers, we run all the experiments with three popular and representative supervised classifiers, Support Vector Machine (SVM), Logistic Regression (LR) and Random Forest (RF).

We show the evaluation (on test data) to the event signal classifiers with different settings in Table 1. In the setting of the Instagram-only method, we discard all Twitter posts. We only extract features from Instagram posts and train all the three classifiers with Instagram data. Then we discard all Instagram posts but extract features and train the classifiers from Twitter data only, as the Twitter-only method. From Table 1, we can find that if we just use a single data source to classify the candidate events, Instagram data outperforms Twitter data. Furthermore, we evaluate the event classifiers on integrated data with two fusion methods. We find that the classifiers trained with the two fusion methods, no matter in data-level or feature-level, both outperform the classifiers trained on a single data source. We investigate results in detail and conclude this

<sup>2</sup> In some special cases, we find there is more than one true event simultaneously recorded in a candidate event signal. We consider them as true events but do not label their photos to avoid ambiguity.

**Table 1.** Event Signal Classification Results

Features	Classifier	Precision	Recall	F score	Overall Accuracy
Instagram-only	SVM	0.833	0.740	0.784	82.01%
	LR	0.855	0.719	0.781	82.28%
	RF	0.793	0.780	0.786	81.36%
Twitter-only	SVM	0.845	0.675	0.751	80.25%
	LR	0.815	0.681	0.742	79.15%
	RF	0.761	0.719	0.739	77.67%
Data-level Fusion	SVM	0.876	0.755	0.811	84.50%
	LR	0.866	0.759	0.809	84.22%
	RF	0.830	<b>0.849</b>	<b>0.839</b>	<b>85.70%</b>
Feature-level Fusion	SVM	<b>0.883</b>	0.746	0.809	84.50%
	LR	0.856	0.774	0.813	84.31%
	RF	0.835	0.836	0.836	85.51%

improvement is caused by the following reasons. First, although we have plenty of geo-tagged posts, in certain sub-regions, we still encounter severe data sparsity problem. Either fusion method brings us more valuable data to mitigate this problem. Second, small-scale events whose weak signals are easily overwhelmed in noisy content. But when we find the weak signals in both data sources, our system are more confident to detect them. Due to the limited length, the evaluation of feature importance is not included. In short, by investigating the weights in Logistic Regression, topic and spatial features are far more discriminative than the emotional and social features.

### 5.3 Relevant Photo Retrieval

To evaluate the efficiency of the relevant photo retrieving method in Eq (1), we use Normalized Discounted Cumulative Gain (NDCG@k) in Eq (4) as the metric. For each event, we rank its photos decreasingly by overall relevance scores in Eq (1), and then compute NDCG@k for the ranking.

$$NDCG_k = \frac{1}{z_n} \sum_{i=1}^k \frac{2^{r_i} - 1}{\log_2(i + 1)} \quad (4)$$

$z_n$  is a normalization factor.  $r_i$  is the actual relevance score of the ranked  $i$ -th photo, and it is given by our labeler, 1, 0.5 or 0.  $k$  is a free parameter to control the number of ranked photos to compute NDCG@k. To reduce the variance caused by  $k$ , we compute the NDCG@k for  $k$  from 1 to 10. We compare the combined relevance model in Eq (1) with three baselines, image content relevance model, text relevance model and geolocation relevance model introduced in Section 4.3. As shown in Table 2, we have the following observations. First, among all the three baselines, the relevance model based on text information works the best. Second, the combination of all the three single relevance models constantly performance better than any of the three single relevance models. Third, by grid search, we empirically find that around  $(a_c, a_t, a_l) = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ ,

**Table 2.** NDCG@k of Relevance Models over  $k$ 

Relevance Model	NDCG@k									
	k=1	2	3	4	5	6	7	8	9	10
Image Content	0.517	0.505	0.495	0.486	0.476	0.465	0.459	0.451	0.445	0.439
Text	0.516	0.547	0.558	0.563	0.572	0.568	0.565	0.562	0.556	0.548
Geolocation	0.334	0.342	0.355	0.366	0.383	0.393	0.402	0.404	0.406	0.405
Combined	<b>0.652</b>	<b>0.656</b>	<b>0.642</b>	<b>0.639</b>	<b>0.629</b>	<b>0.622</b>	<b>0.612</b>	<b>0.598</b>	<b>0.593</b>	<b>0.582</b>

the combined relevance score reaches the maximal on our labeled dataset. This implies the importance of each factor’s contribution to the overall relevance.

#### 5.4 Spatial and Temporal Deviation

In this paper we focus on local event detection in real time, thus we also evaluate the detecting deviation of spatial and temporal factors of events. The detecting deviation of the spatial factor of an event is the geographical distance between the coordinates where the event actually occurred and the coordinates our framework estimated for the event. Similarly, the detecting deviation of the temporal factor of an event is the time period between when the event actually occurred and the time our framework estimated for the event. However, since it is expensive to manually collect accurate spatial and temporal information of an event, we choose 20 events to evaluate their spatial deviation, and 5 events to evaluate their temporal deviation as case study.

We first evaluate spatial deviation of detected events. Many events are held dynamically in a wide region, e.g. New Year parade in China town and marathon, thus we are unable to track all the areas associated with that event. Therefore we only consider events that take place in a fixed area, such as fire accident and basketball games. For each event, we find the name of the associated place, and then take the coordinates of the associated place from Google Maps as the actual event coordinates, in a pair of longitude and latitude. On the other hand, our system calculates the geographic center of of all Instagram and Twitter posts related to that event, as the estimated coordinates of the event. More specifically, the estimated longitude is the arithmetic mean of the longitudes of all related posts, the same for estimated latitude. Then a spatial deviation, i.e. spherical distance, is calculated between the estimated coordinates and the actual coordinates of an event. On average the estimated coordinates of these 20 events are 104.46 meters far from their actual coordinates with a standard deviation of 37.75. Table 3 shows the results for 5 example events.

Similarly, here we evaluate the temporal deviation of detected events. To acquire the exact knowledge of when events occurred, we manually check with websites, the police or news reports for their actual occurrence time. Meanwhile, we take the time of the earliest post among all posts related to that event as the estimated time. We report the time interval between actual time and estimated time of an event as its temporal deviation. Table 4 shows the results of 5 detected

**Table 3.** Spatial Deviation of Detected Events

Event Name	Actual Location (lat, lon)	Deviation(m)
NBA Knicks Game	40.750733, -73.992743	63.1
Trey Songz Concert	40.751222, -73.994749	109.7
Christmas Eve	40.758250, 73.981217	173.1
Fire Accident	40.750369, 73.992726	106.2
Car Crash	40.739061, -74.001488	78.1

**Table 4.** Temporal Deviation of Detected Events

Event Name	Date	Estimated Time	Actual Time	Time Interval
NBA Knicks game	Nov 30 2012	19:11pm	19:30pm	19 mins <b>in advance</b>
Boxing Cotto vs Trout	Dec 01 2012	20:07pm	21:00pm	53 mins <b>in advance</b>
Fire in West Village	Nov 17, 2013	10:40am	10:26am	14 mins later
Fire in 34st	Mar 27, 2014	08:44am	08:35am	9 mins later
Car Crash	Feb 12 2014	08:26am	05:45am	3 hours later

events as examples. We can find that “NBA Knick Game” and “Boxing: Cotto vs Trout”, which are two planned events, are detected prior to the actual event time. This is because as more people arrived to the stadium in advance and started to post about the coming events, our system detected the local unusual increasing trends before the game actually started. For “Fire in West Village”, “Fire in 34 St”, which are two emergencies, our event responded 14 minutes and 9 minutes after the events happened respectively. Notice that for the “Car Crash” event, our system responded 3 hours later. This failure is probably because it happened at 5:45AM, when most of local residents were still sleeping. In this case, few related posts can be detected at the early stage of this event.

## 6 Conclusion and Future Work

In this paper, we proposed a general framework for real-time event detection from Instagram and Twitter post streams. Our proposed system uses three components to discover and classify the events. Then we can extract high-level knowledge from detected events. Extensive experiments on NYC social media data show the promising results. Based on our general framework, a lot of future work can be investigated to potentially boost the performance. For example, we plan to further study how to adaptively divide sub-regions in the city based on their topic distributions [10]. Also, more sophisticated feature fusion approaches for event knowledge extraction can be investigated.

**Acknowledgments.** This work was partially supported by a Magic Grant from the Brown Institute for Media Innovation, and the National Science Foundation grants Numbers 1054177 and 1017845. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

1. Ahmed, A., Hong, L., Smola, A.J.: Hierarchical geographical modeling of user locations from social media posts. In: WWW, pp. 25–36 (2013)
2. Atefeh, F., Khreich, W.: A survey of techniques for event detection in twitter. *Computational Intelligence* (2013)
3. Becker, H., Chen, F., Iter, D., Naaman, M., Gravano, L.: Automatic identification and presentation of twitter content for planned events. In: ICWSM (2011)
4. Becker, H., Iter, D., Naaman, M., Gravano, L.: Identifying content for planned events across social media sites. In: WSDM, pp. 533–542 (2012)
5. Becker, H., Naaman, M., Gravano, L.: Selecting quality twitter content for events. In: ICWSM (2011)
6. Bird, S.: Nltk: The natural language toolkit. In: Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics (2002)
7. Budak, C., Georgiou, T., El Abbadi, D.A.A.: Geoscope: Online detection of geo-correlated information trends in social networks. In: Proceedings of the VLDB Endowment (2013)
8. Chakrabarti, D., Punera, K.: Event summarization using tweets. In: ICWSM (2011)
9. Chen, L., Roy, A.: Event detection from flickr data through wavelet-based spatial analysis. In: CIKM, pp. 523–532 (2009)
10. Flatow, D., Naaman, M., Xie, K.E., Volkovich, Y., Kanza, Y.: On the accuracy of hyper-local geotagging of social media content. In: WSDM (2015)
11. Fujisaka, T., Lee, R., Sumiya, K.: Discovery of user behavior patterns from geo-tagged micro-blogs. In: Proceedings of the 4th International Conference on Uniquitous Information Management and Communication, ICUIMC, pp. 36:1–36:10 (2010)
12. Fung, G.P.C., Yu, J.X., Yu, P.S., Lu, H.: Parameter free bursty events detection in text streams. In: VLDB, pp. 181–192 (2005)
13. He, Q., Chang, K., Lim, E.-P.: Analyzing feature trajectories for event detection. In: SIGIR, pp. 207–214 (2007)
14. He, Q., Chang, K., Lim, E.-P., Zhang, J.: Bursty feature representation for clustering text streams. In: SDM (2007)
15. Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J., Tsioutsouluklis, K.: Discovering geographical topics in the twitter stream. In: WWW, pp. 769–778 (2012)
16. Kleinberg, J.: Bursty and hierarchical structure in streams. In: KDD, pp. 91–101 (2002)
17. Kling, C.C., Kunegis, J., Sizov, S., Staab, S.: Detecting non-gaussian geographical topics in tagged photo collections. In: WSDM (2014)
18. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: ICCV, pp. 2130–2137 (2009)
19. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: WWW, pp. 591–600 (2010)
20. Lee, R., Wakamiya, S., Sumiya, K.: Discovery of unusual regional social activities using geo-tagged microblogs. *World Wide Web* **14**(4), 321–349 (2011)
21. Li, C., Sun, A., Datta, A.: Twevent: Segment-based event detection from tweets. In: CIKM, pp. 155–164 (2012)
22. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.-C.: Tedas: a twitter based event detection and analysis system. In: ICDE, pp. 1273–1276 (2012)



23. Liu, W., Hua, G., Smith, J.R.: Unsupervised one-class learning for automatic outlier removal. In: CVPR (2014)
24. Liu, W., Wang, J., Kumar, S., Chang, S.-F.: Hashing with graphs. In: ICML, pp. 1–8 (2011)
25. Nichols, J., Mahmud, J., Drews, C.: Summarizing sporting events using twitter. In: Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces, pp. 189–198 (2012)
26. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision* **42**, 145–175 (2001)
27. Papadopoulos, S., Zigkolis, C., Kompatsiaris, Y., Vakali, A.: Cluster-based landmark and event detection on tagged photo collections. *IEEE Multimedia* (2010)
28. Parikh, R., Karlapalem, K.: Et: Events from tweets. In: WWW Companion, pp. 613–620 (2013)
29. Rasmussen, C.E., Williams, C.K.I.: *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (2005)
30. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: WWW, pp. 851–860 (2010)
31. Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M.D., Sperling, J.: Twitterstand: news in tweets. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 42–51. ACM (2009)
32. Valkanas, G., Gunopulos, D.: How the live web feels about events. In: CIKM, pp. 639–648 (2013)
33. Weng, J., Lee, B.-S.: Event detection in twitter. In: ICWSM (2011)
34. Xia, C., Schwartz, R., Xie, K., Krebs, A., Langdon, A., Ting, J., Naaman, M.: Citybeat: Real-time social media visualization of hyper-local city data. In: WWW Companion, pp. 167–170 (2014)
35. Xie, K., Xia, C., Grinberg, N., Schwartz, R., Naaman, M.: Robust detection of hyper-local events from geotagged social media data. In: Proceedings of the 13th Workshop on Multimedia Data Mining in KDD (2013)