Robust Detection of Hyper-local Events from Geotagged Social Media Data

Ke Xie, Chaolun Xia, Nir Grinberg, Raz Schwartz, Mor Naaman
Rutgers University
{kx19,cx28,nirg}@cs.rutgers.edu,
{raz.schwartz,mor}@rutgers.edu

ABSTRACT

An increasing number of location-annotated content available from social media channels like Twitter, Instagram, Foursquare and others are reflecting users' local activities and their attention like never before. In particular, we now have enough available data to start extracting real-time local information from social media. In this paper, we focus on the problem of hyper-local event detection, with the goal of enabling a monitoring and alerts system for public management officers, journalists and other users. We present a method for real-time hyper-local event detection from Instagram photos data, using two computational steps. We first use time series analysis to detect abnormal signals in a small region. We then use a classifier to decide if the detected activity corresponds to an actual event. Testing on a large-scale dataset of New York City photos, our system detects hyper-local events with high accuracy.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Representation**]: Multimedia Information Systems; H.2.8.d [**Database Management**]: Database Applications—*Data mining*

General Terms

Data Mining

Keywords

Event detection, social media

1. INTRODUCTION

Popular social media platforms like Instagram and Twitter are making available huge amounts of user content that is associated with an exact location, and shared in real time. Users voluntarily share, in public settings, photos, videos and text reflecting their attention, interest, opinions, artistic expression, and – in some cases – what is happening around them.

At scale, this data presents an opportunity to discover hyper-local events using social media content alone. We define *hyper-local events* as events detected in a fine granularity, street or building level for instance, rather than city- or country-level events. These events can range from a concert, an exhibit, to a protest, to an emergency like a fire or even an explosion. A robust real-time system to detect hyper-local events could be useful for various stakeholders like city government, journalists, first responders, and, even more mundanely, for individuals looking for events to attend in their local area.

Local event detection using social media data remains a challenging task. First, much of the content people post on social media is not event-driven, with a wide range of topics, subjects and motivations for posting [12]. Thus, the "event" signal can easily be lost in the sea of available social media content. This problem is especially acute for local events, which usually involve only a handful of people. Even if event content is available, the social media content may contain remarkably little textual information, which often is low quality. Finally, the real-time requirement for the task poses additional constraints given the immense scale of the data.

Previous efforts have mostly focused on detection of global (i.e. events detected in global or country level) events in social media data. These efforts (e.g., [5, 11]), often build on a significant volume of messages that are associated with a keyword or a topic. Thus, the methods often require a large amount of data to accumulate before making a decision about a detection of event. For example, the work by Becker et al. [3] uses an incremental clustering method and declares an event when enough volume of content is seen for each cluster.

To tackle these challenges, we propose a two-step framework for real-time hyper-local event detection, combining a time-series predicting component and a classifier. Modeling this time series correctly and robustly helps detect abnormal signals that surfaces *candidate events* in hyper-local areas (a small geographical area). In the second step, discriminative features are extracted for the candidate events, and used to classify the candidate events into "actual" or "false" categories. Our approach can detect hyper-local events in real time even when the associated social media data is sparse.

The rest of this paper is structured as follows. In Section 2 we review previous research on event detection in social media. In Section 3 we define the problem of hyper-local event detection, and describe the data we collect and used in this work. Section 4 describes our detection framework in detail. We present the evaluation and results in Section 5.

2. RELATED WORK

The event detection task was formulated and studied in multiple domains, and had recently been given specific attention in the context of social media. In earlier work, for example, academic papers [9] and news article [6] are two domains where researchers tried to detect research topics and breaking news. Recently, with the emergence and adoption of social media platforms, there has been significant attention on detecting events from social media services such as Flickr, Twitter and others. Below, we expand on related work on social media event detection, which we group into two themes: document-pivot methods and feature-pivot method (following [2]).

The main idea of document-pivot method is to group related social media items ("documents") using similarity metrics based on text and other features. For example, researchers used an incremental clustering algorithm to accumulate content into topic clusters, and highlight clusters with high event score as event clusters [3]; Similar methods were used in [11]. Document-pivot methods were found useful by researchers of IBM Smart City project which used clustering to detect emergent events in tweets for New York city [15]. On the other hand, Aiello et al. found that document-pivot methods suffer from clustering fragmentation problems and depends on arbitrary threshold for the inclusion of a new document to an existing topic [2].

Taking a different approach, modeling social media items as nodes, and the similarity between one and another as weights of edges, researchers proposed a novel framework of using graph clustering algorithms like modularity maximization to assign items to clusters [18, 21]. Document-pivot methods have a number of drawbacks: these methods require an often-arbitrary (and generic) threshold for creating a cluster representing an event, and may suffer from cluster over- and under-segmention, making the threshold issue even more critical. As a result, these methods are also likely to introduce a gap between the time an event happens and the time an event cluster is detected.

Another approach to detecting events, feature-pivot method, is to formulate the problem as an abnormal signal detection task on individual features in the time series. These features are often words or phrases, or a set of topic words that appear in the social media items (extracted via LDA or similar methods) [2, 8]. The abnormal signal detection focus can help avoid the drawbacks encountered in the document-pivot methods. Previous work suggests that by tracking time-series built on keywords, finer granularity events could be detected [14, 21]. Researchers have considered multiple methods for detecting abnormal time series activities, e.g. using spatial scan statistics [14] or Wavelets [21]. Usually, though, no further judgement or classification is performed on the detected events. In this work, we use a geography-based feature (i.e. posts from a geographic area) and employ of Gaussian Process Regression (GPR) as our non-parametric time-series prediction algorithm. We incorporate a supervised classifier in our framework to further classify the candidate events.

Indeed, a better understanding of detected events can alleviate the effects of noisy data and filter those content containing no event information. For detecting events in specific domains, such as emergent events, specifying keywords (fire, firefighter etc) would yield satisfactory results [1, 4, 15]. The limitation with this scheme is that exhaustively specifying predefined keywords is not feasible, especially for social media data. A more general way to filter content is to train a binary supervised classifier to classify content as event-related [3, 16]. In this paper, we are pushing the event detection resolution and classification to a hyper-local level, instead of global or country level achieved previously.

Local event detection was attempted in [19, 20]. Our approach differs from these in a number of ways. First, we do not set hard threshold which makes our system more adaptable to other data sources. Second, we explicitly model volume signals and classify the data only when an event signal is presenting. Third, we present a novel classification model to separate detected events and improve the event detection accuracy.

3. PROBLEM DEFINITION AND DATA COLLECTING

In this section, we define the task of local event detection in social media data, and describe the data we collected and used in this paper.

3.1 Problem Definition

Following the definition from [3], we define our problem as follows. Consider a time-ordered stream of social media data M from a geographic location G. At any point in time t, our goal is to identify real-world local events and their associated data present in M published before time t. We assume an online setting for our problem, where we only have access to data posted before time t.

3.2 Data Collecting and Dataset

In this paper, we consider only geo-tagged data from Instagram, a popular photo sharing service. Specifically, we consider only Instagram photos p with a geocoded field p_l available that indicates the exact coordinates where the photos were taken. For each photo p, we also have the time of the photo p_t, and the caption p_c which is a free text string a user uses to describe the photo. While focusing here on Instagram data, we note that this general formulation can be applied to other data sources, like Twitter.

The dataset we use in the experiments are all the photos collected from Instagram API during November 15th, 2012 to December 15th, 2012. Specifically, we use Instagram region query API endpoint to gather all the photos bounded by a geo-region, $(40.6905, -74.0581)$ and $(40.8231, -73.8579)$ (representing the New York City area), for a total of $906,235$ photos.

4. LOCAL EVENT DETECTION FRAMEWORK

In this section, we first give an overview of the architecture for our framework followed by details of the main components in our framework. The main components of our framework include a Gaussian Process Regression (GPR) as time series prediction model, and a supervised classifier for candidate event classification.

4.1 Architecture and System Workflow

As illustrated in Figure 1, the system architecture consists of five main modules. First, the data collector reads an

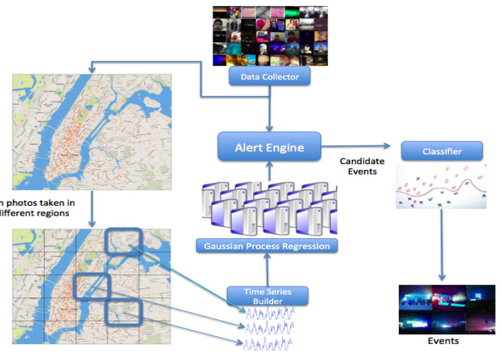

Figure 1: Architecture of our local event detection system. Including data collector, time-series builder, Gaussian Process regression model, alert engine and classifier. Arrows indicate input and output flow of each module.

Instagram photo stream using the Instagram API service, and store the photos in database. The photos are used by a time-series builder, that constructs a time-series model for **each** geographic subregion in our data. A geo-region is (in this work) defined arbitrarily by a bounding box, but could also be a polygon that represents, for example, a neighborhood. The time series is constructed using the number of unique users posting in a region G, we call volume, within a time window $t \in T$, where T is the time span we construct the time-series.

We train our GPR time-series prediction models for each geo-region G for the time span T we specify independently using the time series data for that time span. Once the GPR models are built, we could make predictions for future volume in each region. The output of GPR models for each time t and location G are tuples $[\mu_{G,t}, \sigma_{G,t}]$, representing the predicted mean value for volume and associated standard deviation for volume. The prediction for a certain region serve as volumes of data we expect to observe given no event is happening for that region at a certain time.

We denote $V_{G,t}$ as the number of unique users posting for a geolocation G at time t. If $V_{G,t}$ exceeds the prediction by some threshold, the alert engine (middle of Figure 1) would mark that time period t as a *candidate event* for that location G. Since a deviation does not necessarily mean a true event (e.g., can also be due to random noise), we use a trained classifier to review the features of a candidate event and mark it as true or false. An intended benefit of the classification step is that we can keep higher recall, and to be able to detect hyper-local low-volume events, by setting a low threshold for the candidate event generation, assuming the classification step will take care of the precision requirement.

4.2 Time Series Prediction

We use Gaussian Process Regression (GPR) as our time series prediction model in our framework. GPR was shown to perform well in various time-series data including stock price predicting and electricity usage prediction [10, 13]. In this section, we briefly introduce to GPR, and provide the details on how we applied it to the problem at hand.

We use GPR as the time-series predicting model in our framework for a number of reasons. First, GPR would produce a prediction with a variance which could be used to build a confidence interval of the prediction, while other widely used autoregressive models like STL or ARIMA would not. Second, by specifying appropriate kernels, GPR adapts to various kinds of time-series simply by replacing the kernels. Third, once kernels are specified, no other parameters need to be tuned, an ideal property to reduce the complexity of our framework.

4.2.1 *Gaussian Process Regression Overview*

A Gaussian Process is a collection of random variables, any finite subset of which have a joint Gaussian distribution. We could simply write a Gaussian Process as

$$y = f(\mathbf{x}) + \varepsilon \tag{1}$$

where $f(x)$ is the real process and ε is a additive independent identically distributed Gaussian noise with variance σ_n.

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2}$$

Here, we view the input of a GPR as pairs of vectors $\mathbf{x}, \mathbf{y}$ where $\mathbf{x}$ could be a feature vector and $\mathbf{y}$ could be the target vector. In this paper, we take $\mathbf{y}$ as a single value (i.e. the volume). In this setting, we could view the feature vector $\mathbf{x}$ as an index, and $m(\mathbf{x})$ is the mean function which is usually set to zero though not necessarily. $k(\mathbf{x}, \mathbf{x}')$ is the covariance function which gives the covariance of points $f(\mathbf{x})$ and $f(\mathbf{x}')$. For example, one widely used kernel function is *squared exponential* kernel which specify the covariance of pairs of random variables as

$$cov(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}') = \sigma_f exp(-\frac{1}{2l^2}|\mathbf{x} - \mathbf{x}|^2) \tag{3}$$

σ_f and l are hyperparameters which control the magnitude and length-scale of the process. The role of covariance function in the GPR framework is similar to that of the kernels used in Support Vector Machine [7]. A particular choice of such a covariance function induces our assumption of the underlying function that generate the data.

As any finite subset of a GPR has a joint Gaussian distribution, when observing a new test point x_*, we could conditioned on the observed data and get the predictions mean of $f(x_*)$ as follow

$$\mu(x_*) = \mathbf{k_*}^T (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{4}$$

$$\sigma^2 = \mathbf{k}(\mathbf{x_*}, \mathbf{x_*}) - \mathbf{k_*}^T (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k_*} \tag{5}$$

In the predictive distribution, $\mathbf{k}_*$ is covariance vector $\{cov(\mathbf{x_*}, \mathbf{x_1}), cov(\mathbf{x_*}, \mathbf{x_2}), ..., cov(\mathbf{x_*}, \mathbf{x_n})\}$ where $\mathbf{x_i}$ is the feature vector of i_{th} training point. K is the covariance matrix simply by taking all the pairs of training examples and compute $k(\mathbf{x}, \mathbf{x}')$. $\mathbf{y}$ would be the values of the target values. We could interpret the mean prediction as a linear combination of the target values given a new test point, and the weights of the linear combination is controlled by the "similarity" of the test point and the training point which is decided by the covariance function we choose to use. As in our case, we are using GPR as time series prediction model, so the only feature we are using is the time stamp and the target value would be the volume of data at the specific location. Intuitively in time-series prediction, for prediction of a target point, the closer the training points are, the greater influence they would have for the target point.

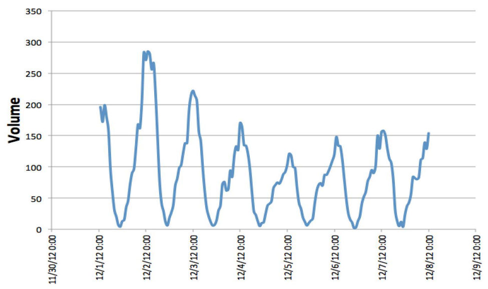

Figure 2: Volume change at Times Square over time binned by hour

4.2.2 *Kernels for Time Series Modelling*

Taking the volume binned as one hour unit at Time Squares for the first week of Dec 2012 as a typical example, we can see from Figure 2 that there are daily and weekly (mostly weekend related) periodicity we need to model

To model the properties, we combine the following three kernels as

$$K_c = prod\{k_1, k_2\} + prod\{k_1, k_3\} \tag{6}$$

in which k_1, k_2 and k_3 are three separate kernels.

$$k_1(x, x') = \theta_1 exp(-\frac{2sin^2(\pi(||x - x'||)/\theta_2)}{\theta_3^2}) \tag{7}$$

$$k_2(x, x') = \theta_4^2(1 + \frac{(x - x')^2}{2\theta_5\theta_6^2})^{-\theta_5} \tag{8}$$

$$k_3(x, x') = \theta_7^2 exp(-\frac{(x - x')^2}{2\theta_8^2}) \tag{9}$$

The operator *prod* here is an element wise product with the corresponding elements of the covariance matrix and K_c is the combined kernel in our GPR model.

We model the time-series using $prod\{k_1, k_2\}$ to capture the daily periodic pattern with disturbances. Since the volume produced might be conditioned on many unknown real-world situations such as the weather that day, by using $prod\{k_1, k_2\}$ we allow the "decay away" from the exact periodic pattern which is desirable to tolerate un-modeled noises in the setting. The second term in Equation 6 is used to model the long term periodic pattern like weekend spikes. However, this periodic trending is not static, because since the date is moving towards to Christmas (In the experiment we are using data from December 2012, see Section 5 for details), the trend is slightly rising. Thus, allowing the periodic kernel to adapt the trend is suitable. The hyper-parameters for K_c is then $\{\{\theta_1, \theta_2, \theta_3\}, \{\theta_4, \theta_5, \theta_6\}, \{\theta_1^*, \theta_2^*, \theta_3^*\}^1, \{\theta_7, \theta_8\}\}$.

4.2.3 *Distributed Parameter Estimation and Computation Complexity*

Given the time-series data and choice of kernel functions as in the section 4.2.2, we need to optimize 11 hyper-parameters. We follow [13] and adopt setting the hyper-parameters by maximizing the marginal likelihood.

The complexity of computing the marginal likelihood is determined by the $O(n^3)$ operation of the inversion of a n by n covariance matrix K. Though $O(n^3)$ operation is expensive, however, as in our case we are modelling each region independently and to predict near future we only need very recent historical data. Therefore the n is small (usually around 400, see experiment section for detail). The total complexity for our algorithm is $O(kn^3)$ where k is the number of geo-regions we divide. In our setting, $k = 625$ and we use a cluster of 60 machines to parallelly build the GPR models which could finish building within 1 hour on our live system.

4.3 Candidate Event Classification

In the first step of our two-stage event detection framework, the time series GPR-based model outputs a set of candidate events. However, not all of the candidate events are actual events. In fact, to improve recall, we aim to set the threshold of GPR as low as possible, so that more candidate events are collected, also resulting in a large number of false positives. To filter out these false positives, we use a supervised classifier. In this section we explain how we classify a candidate event into one of actual/false event categories.

$$D_{KL}(P_c||P_{c_h}) = \sum_{w \in W_c \cup W_{c_h}} P_c(w) \ln \frac{P_c(w)}{P_{c_h}(w)} \tag{10}$$

$$D(c, c_h) = \frac{1}{2}[D_{KL}(P_c||P_{c_h}) + D_{KL}(P_{c_h}||P_c)] \tag{11}$$

Each of the candidate events is a set of content items $M_{G,t}$, representing the items shared from location G at time t where the event was detected. We compute 22 features for each candidate event based on these content items. In our case, each item is an Instagram photo. It should be noted that even if a candidate event captures an actual event, it is likely that there are items in the set that are not related to the event, but were still posted at the same place and time. Therefore, we need to design not only discriminative but also robust features to prevent from the impact of "noisy" items. Specifically, we consider four types of features for a candidate event: spatial, textual, meta and historical features. The complete list of features is shown in Table 1.

4.3.1 *Spatial Features*

Spatial features capture the geographical distribution of items in candidate events. The assumption is that reports of actual physical-world events are geographically proximate to each other, and are likely to form a dense cluster with small diameter even in an already-constrained geographical region. Given the latitude and longitude information of all items, we compute three geographic distribution features for each candidate event. First, we compute the mean of pairwise Euclidean distance between items in the candidate event, as well as the standard deviation. However, this formulation is sensitive to outliers. Thus we use a third feature we call geo-entropy. We evenly divide the region G^c where the candidate event c is taking place into k smaller subregions g_i^c $(1 \leq i \leq k)$, and compute the probability distribution $P(x = g_i^c)$ that an item x is posted in the subregion g_i^c in Equation 12. The geo-entropy $E(c)$ is the entropy of the probability

[1] $\theta_1, \theta_2, \theta_3$ and $\theta_1^*, \theta_2^*, \theta_3^*$ are hyper-parameters for daily periodic kernel and weekly periodic kernel respectively.

Category	Name	Description
Spatial	Average geo-distance	Mean of pairwise item geographical distance
	Std of geo-distance	Standard deviation of pairwise item geographical distance
	Geo-entropy	Entropy of item geographical distribution ($k = 9$)
Textual	Top-k word popularity	Average frequency of top-k words ($k = 3$)
	Top word popularity 1	Frequency of items containing the most frequent word
	Top word popularity 2	Frequency of items containing the second-most frequent word
	Top word popularity 3	Frequency of items containing the third-most frequent word
	Average caption distance	Mean of pairwise caption topic distribution KL-divergence
	TFIDF value 1	The largest TFIDF value of all words
	TFIDF value 2	The second-largest TFIDF value of all words
	TFIDF value 3	The third-largest TFIDF value of all words
	Hashtag popularity 1	Frequency of items containing the most frequent hashtag
	Hashtag popularity 2	Frequency of items containing the second-most frequent hashtag
	Hashtag popularity 3	Frequency of items containing the third-most frequent hashtag
Meta	Average caption length	Average length of captions
	Caption percentage	Percentage of items with a non-empty caption
	Predicted std	Predicted standard deviation by GPR
	Z-score	Deviation of the predicted volume from the observed volume
	Top location popularity	Frequency of the most frequent location tag
Historical	Geo-distance difference	Difference between the average geo-distance of c and c_h
	Geo-entropy difference	Difference between the geo-entropy of c and c_h
	Topic difference	KL-divergence between the caption topic distributions in c and c_h

Table 1: Complete feature table for a candidate event c

distribution $P(x)$, and captures the geographic distribution of the items in a more robust fashion.

$$P(x = g_i^c) = \frac{\#\ of\ items\ in\ g_i^c}{\sum_{j=1}^{k} \#\ of\ items\ in\ g_j^c} \quad (12)$$

$$E(c) = -\sum_{i=1}^{k} P(g_i^c) \ln P(g_i^c) \quad (13)$$

4.3.2 Textual Features

The majority of items are associated with a short caption (e.g. "I love #Knicks"); we expect that the topics in the captions of actual events are more semantically consistent with each other than those of non-events. However, a caption is too short (53 letters including symbols on average in our data) to build topic models. Instead, for each candidate event, we extract 11 features by computing the normalized frequency and TFIDF value of top-k words (ranked by normalized frequency and TFIDF value respectively) in Table 1. We also parse out hashtags, e.g. *#Knicks*, from captions, and compute their normalized frequency as features. If the values of these features are very large, items within the candidate event are semantically consistent with each other, which might indicate an actual event.

4.3.3 Meta Features

The basic characteristics and statistics of the candidate event can also provide information about its nature. We use several "meta" features that reflect the basic but discriminative information of a candidate event: a) the Z-score of a candidate event (its deviation from expected volume) together with the predicted standard deviation computed from GPR time series model; b) the average caption length and the percentage of items with a caption; c) the normalized frequency of the most frequent location tag (e.g. *Seoul Garden*) in each candidate event.

4.3.4 Historical Features

The spatial, textual and meta features listed above ignore the background information, e.g. the geographical distribution of items in the region when no event is taking place. The occurrence of an actual event is likely to influence the item distribution in that region. Motivated by this, for each candidate event c, we first create its background by aggregating all the items posted in the same region and the same period across each day of the past week, as a dummy event c_h corresponding to c. We compute several "historic" features to quantitatively express the difference between the item distribution of candidate event c and its background c_h. First, two features quantify the change of item graphical distribution caused by the occurrence of an actual event: the difference between the geo-distance of c and c_h and the difference between the geo-entropy of c and c_h. Second, we design features that capture the divergence in topic between event items and other items in that region using the averaged KL-divergence between the caption topic distributions of c and c_h. As shown in Equation 10 and 11, W_c and W_{c_h} denote the vocabulary constructed from c and c_h respectively. P_c and P_{c_h} denote the word frequency distribution (bag-of-words model) of c and c_h respectively where $\sum_{w \in W_c} P_h(w) = 1$ and $\sum_{w \in W_{c_h}} P_{c_h}(w) = 1$.

5. RESULTS

In this section, we evaluate the trade off in setting the size of subregion, G. We then evaluate the effectiveness of GPR as candidate event detector, and our schema of feature extraction, by comparing them with baselines in the literature. Finally, we report the accuracy of our framework for the local event detection task.

5.1 Tradeoff of Subregion Size

To detect events, the first step in our framework identifies abnormal activity using the time-series we build for each geo-region G, as described in Section 5.2. However, an open question is how large or small the area G can be to robustly and reliably detect abnormal activity. If the area G is too large, we will not be able to detect smaller events that do not have a large volume of posts. If the area G is too small, the volume curve for each region would become extremely variable and volatile, generating too many candidate events.

We experiment with four different size configurations, including equally dividing our overall region (covering New York City) into 10×10 (2.72 square kilometers, averagely 767 photos per square each day), 15×15 (1.21 square kilometers, averagely 340 photos per square each day), 20×20 (0.68 square kilometers, averagely 191 photos per square each day) and 25×25 (0.43 square kilometers, averagely 122 photos per square each day) regions. For each setting, candidate events are generated, randomly sampled and annotated . We use $Event_{ration}$ as defined in Equation 14 as a metric to measure the ratio of actual events in all the candidate events detected, and this metric is used to evaluate how good each dividing setting is. Intuitively, the higher the $Event_{ratio}$ is, the better we could eliminate noise and keep more actual events. We find that the 20×20 setting (0.43 square kilometers) performed best in terms of $Event_{ratio}$. Due to page limitation we eliminate details here. Below, we use this subregion size in all our experiments.

5.2 Effectiveness of GPR in Event Detection

The goal of using a time series model in our framework is to initially identify candidate event signals. As such, we expect the GPR process to identify many candidate events, most of them false. Our goal here is to examine the GRP output to learn more precisely about the ratio of true to false events detected, and compare it to a baseline method. We use $Event_{ratio}$ as in Equation 14 as a metric. Recall that the method that has higher $Event_{ratio}$ value could detect more actual events while introducing less noise. We compare our GPR model with a baseline method we call *Hourly Window* method. The baseline models the dynamics of volume within a day for each hour, such as 12:00 to 13:00, as the average of the same hour over all the days in historical data. Also, a standard deviation could be computed in the same manner. An event will be detected by this baseline if the volume deviates 3σ threshold from the mean (3σ is also set for GPR as well; we settled on it by experimentation).

$$Event_{ratio} = \frac{\text{\# of actual events}}{\text{\# of candidate events}} \tag{14}$$

We generate two sets of candidate events using GPR and the baseline respectively. To this end, we imitate a streaming of data which "flow" sequentially over the first week of December. For each day of that week, we build time-series at 00:00 of each day for all the subregions using 2 weeks of historical data. We found two weeks of data is enough to model the daily and weekly periodicity.

We computed models for all our regions G; as there are 25 by 25 subregions that we are dividing into, 625 separate GPR models needed to be built, for each day of our test. For each time series, we use a bin of one hour, giving us 336 data points for the 14 days of historical data. These are relatively low numbers, making the $O(n^3)$ time for building a GPR model acceptable. Nevertheless, we use a cluster of 60 machines to train the GPRs in parallel. To avoid local minimum as described in Section 4.2, we randomly sample the initial hyper-parameters 20 times and choose the one with the best marginal likelihood. That is, for each day, 12,500 GPR models need to be trained (20×625). We could complete all the computation on the cluster within one hour, and this training process only needs to be done every 24 hours in an offline fashion. For the baseline, we compute the hourly mean and standard deviation for each hour, and use the mean and variance for that day as predicted mean and predicted variance.

Then, in real time, every 5 minute, by comparing the predicted volume with what is observed in that subregion from the data streaming, we generate a candidate event when the volume exceeds the predefined threshold, 3σ. We produce candidate events from December 1st to December 7th using both GPR and baseline model. Two of the authors randomly labeled a random set of 150 candidate events generated by each of the two models (the annotator agreement with high with $\kappa = 0.85$; below, we use crowd sourcing to generate a larger ground truth dataset of candidate events). The results are shown in Table 2. As we can see in the table, GPR and the baseline generate roughly the same number of candidate events while GPR has two times of $Event_{ratio}$ as baseline method. We conclude that using GPR would effectively generate candidate events with a higher $Event_{ratio}$. In the following experiments, we use GPR to generate candidate events.

5.3 Evaluation of Event Classification

As noted above, the candidate events, by design, could be false alarms. We train a classifier using features discussed in Section 4.3. In this section, we evaluate the accuracy of our classification compared to the approached introduced in [3, 17], which we refer to it as *Text-baseline*.

5.3.1 Data Annotation and Ground Truth

In the experiments, we use Instagram data from the first week of December to generate candidate events (as described in Section 5.2). To get the ground truth labels for candidate events we generate, we used CrowdFlower, a crowd-sourcing platform. We got multiple annotations for each of the $2,558$ candidate events generated for the first week of December by GPR. For each candidate event, we created a page that showed the photos with captions in that candidate event. Following our definition, for each event, we asked the annotators to look over the content items for that event (usually not more than a few dozens), and determine whether there exist at least three photos consistently describing an unusual activity or abnormal occurrence.

For each candidate event, we ask two annotators to annotate it independently. We discard all the candidate events if the two independent annotators disagree with each other on their labels. Of $2,558$ candidate events, 865 candidate events (34%) were discarded, 181 were marked as true events (7%), and $1,512$ were marked as non-events (60%).

5.3.2 Evaluation Setup

We use the 1,693 annotated candidates events and evaluate each of the classification setting using 10-fold cross validation. We report results using logistic regression, Naive Bayes and SVM classifiers. We use the following settings:

	# of Candidate Events	# of Sampled Event	# of Actual Event in Sample	$Event_ratio$
GPR	2558	150	20	**13.3%**
Hourly Window	2492	150	10	6.6%

Table 2: $Event_{ratio}$ for GPR and Hourly Window method

- *Text-baseline*: Baseline setting. Regard all the caption in a candidate event as a document and compute the TF-IDF features for each document as a feature vector.

- *STMH features*: The feature extraction schema used in our framework, including **S**patial features, **T**extual features, **M**eta features and **H**istorical features, all discussed in Section 4 (abbreviated as STMH features).

In [3], the authors noted that the data is highly skewed towards negative examples, and resampled the events to equalize positive and negative examples and generate a balanced dataset. However, resampling would make the task artificially easy because either use their clustering method or our GPR model, we would generate more non-events than actual events. Thus, we report results both balanced data (for comparison) and unbalanced data. Note that for unbalanced data, the accuracy we report is balanced accuracy which is defined as Equation 15. Using this metric could avoid the classifier taking advantange of the imbalanced data (classify all future data as the majority class) since if so the balanced accuracy would drop to chance.

5.3.3 Evaluation Results

The results appear in Table 3. Each row of this table is different settings with various classifiers (NB, LR, SVM), and columns are the evaluation metric on both resampled data and unsampled data. Standard metrics are used here including precision, recall, as well as accuracy and Fscore as defined in Equation 16 and Equation 17 . We observe that no matter on unsampled data or resampled data, our approach outperforms the baseline. Specifically, on resampled data, our approach performs 3.2% to 15.6% more accurately than the baseline method with respect to accuracy on either Naive Bayes, Logistic Regression or SVM. On unsampled skewed data, which is the actual situation we need to deal with in real world, our approach performs 11.6% to 24.4% more accurately than the baseline. The best accuracy of 89.3% is achieved using Logistic Regression on unsampled data. The results demonstrate that our selection of features could distinguish actual events and otherwise more effectively than the baseline. Especially, our approach outperforms the baseline significantly on unsampled data.

$$balanced \quad accuracy = \frac{0.5 * TP}{TP + FN} + \frac{0.5 * TN}{TN + FP}{}^{1} \qquad (15)$$

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \qquad (16)$$

$$Fscore = 2 * \frac{precision * recall}{precision + recall} \qquad (17)$$

[1]TP=True Positive, TN=True Negative, FP=False Positive, FN=False Negative

As event detection algorithms are ultimately used to detect events in the future using data from the past which means there should be chronological order of training and testing data we use. To ensure our method is able to detect events on unseen future data stream, one more experiment is performed. We use annotated candidate events from the first week of December as training set, and test on the candidate events generated during the second week of December to show how the SMTH features could generalize regardless of when the candidate events are generated. For this task, we obtain 86.9% accuracy using SVM, which is close to the best accuracy 89% we obtain using cross-validation and these results demonstrates our framework could generalize its event detection ability to unseen future data.

5.4 Qualitative Analysis of Detected Events

In this section we qualitatively analyze some of the events our system detected during the first weeks of December. The various categories of hyper-local events detected included concerts, sport games, exhibits. We coded the events to categories, and the results are shown in Table 4. The diverse categories of events show that our system could detect not only major events like concert but also minor yet important or urgent events like protesting and car explosion. Also, the locations of the 12 events that are detected distribute from Brooklyn to Times Square, which indicates our system cover wide range of the city.

An examination of the detected events shows two events for the evaluation period that took place in Manhattan's Time Square, one of these, for instance, was a protest, another event is that stars from The Hobbit showed up at Time Square and people crowed to take photos. These events show that hyper-local events can be detected even in a very popular area, where scores of tourists take thousands of pictures and upload to Instagram every day. Our system also detected other emergencies. Photos of a critical event are shown in Figure 3, in chronological order: a car caught fire, and then policemen came followed by firefighters. In our system log, we detected the explosion only five minutes after the car caught on fire as shown in the first photo of Figure 3. The events our system detect demonstrate our system has the desired capability of extracting event signals from noisy and sparse social media data in real time.

6. CONCLUSION AND FUTURE WORK

In this paper, we studied the task of robustly detecting hyper-local events from social media data. Social media has become a vital information source; accurately detecting local events from social media could benefit various stakeholders including journalists, city government employees, first responders – and people who simply wish to explore events happening around them.

On the other hand, the task is challenging given the amount of noise in social media signals. We address the problem using a framework consisting mainly of two parts, the Gaussian

		Resampled Data				Unsampled Data			
		Precision	**Recall**	**Fscore**	**Accuracy**	**Precision**	**Recall**	**Fscore**	**Accuracy**^{2}
Text baseline	NB	0.87	0.68	0.76	0.79	0.30	0.57	0.40	0.71
	LR	0.74	0.65	0.69	0.71	0.17	0.66	0.28	0.64
	SVM	0.81	0.75	0.78	0.79	0.21	0.73	0.32	0.70
SMTH	NB	**0.92**	0.71	0.80	0.82	**0.64**	0.69	**0.67**	0.82
	LR	0.88	0.85	**0.86**	**0.87**	0.51	**0.88**	0.65	**0.89**
	SVM	0.85	**0.86**	0.86	0.85	0.49	0.87	0.62	0.88

Table 3: Classification results on balanced data and unbalanced data comparing with baseline

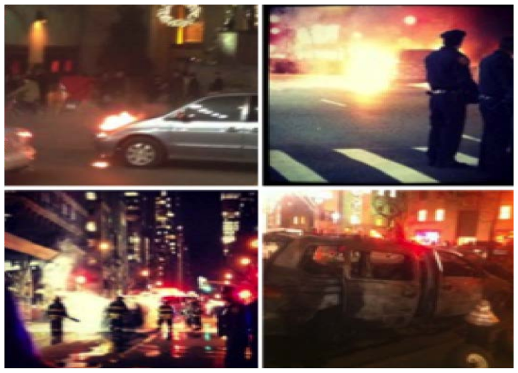
Figure 3: An car explosion accident detected by our framework ordered by chronological sequence

Process regression time-series predicting model used to detect abnormal signal and the event classifier used to further judge whether the abnormal signal is an event. The framework we described could effectively eliminate the noise exist in social media streaming data, and is capable of detecting building-level events robustly and with high accuracy. We validate our framework on a large scale dataset we collect from Instagram containing all the photos taken in New York City during November and the first two weeks of December. The results are promising that various types of events are found including minor but critical events.

We used Instagram in this work, but our framework is adaptable for various social media streaming types that support geolocation information including Twitter, Facebook, and Foursquare. By replacing suitable kernels in Gaussian Process regression models and engineer suitable features, the framework is flexible enough to extend to other data sources without much work. However, it is an open question whether other services that are not based on photographic documention reflect events in the same way that Instagram does. And also, the data quality, for example the accuracy of gps data on different platforms might differ thus influence the result of detected events. In future work, we intend to investigate whether events can be detected in streams of different types, and whether these streams detect and highlight different types of activities. We would also aim to combine multiple streams, as well as to categorize the events detected automatically.

References

[1] Puneet Agarwal, Rajgopal Vaithiyanathan, Saurabh Sharma, and Gautam Shroff. Catching the long-tail: Extracting local news events from twitter. *Sixth International AAAI Conference on Weblogs and Social Media (ICWSM 12)*, 2012.

[2] Luca Maria Aiello, Petkos Georgios, Martin Carlos, Corney David, and Papadopoulos Symeon. Sensing trending topics in Twitter. *IEEE Transactions on Multimedia*, 2013.

[3] Hila Becker, Mor Naaman, and Luis Gravano. Beyond trending topics: Real-world event identification on twitter. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 11)*, 2011.

[4] Hila Becker, Mor Naaman, and Luis Gravano. Selecting quality twitter content for events. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 11)*, 2011.

[5] Ling Chen and Abhishek Roy. Event detection from flickr data through wavelet-based spatial analysis. *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 523–532, 2009.

[6] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S Yu, and Hongjun Lu. Parameter free bursty events detection in text streams. *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, August 2005.

[7] Agathe Girard, Carl Edward Rasmussen, Joaquin Quinonero-Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs? application to multiple-step ahead time series forecasting. *MIT Press*, 2003.

[8] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis. Discovering geographical topics in the twitter stream. *WWW '12: Proceedings of the 21st international conference on World Wide Web*, April 2012.

[9] Jon Kleinberg. Bursty and hierarchical structure in streams. *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

Event Type	Sample Events	Location	Time	Description
Concert & Show	One Direction Concert	Madison Square Garden	03 Dec 2012 22:10:17	band concert
	Z100's Jingle Ball	Madison Square Garden	07 Dec 2012 15:20:09	live performance
Sports	Basketball	Madison Square Garden	30 Nov 2012 19:11:48	Knicks vs Wizards
	Boxing	Madison Square Garden	01 Dec 2012 20:07:49	Cotto vs Trout
Emergencies	Protest	Times Square	06 Dec 2012 16:52:08	union workers protest
	car explosion	Fifth Avenue	05 Dec 2012 18:55:12	a minivan exploded
Activities	Hendricks Gin party	financial district	06 Dec 2012 21:13:56	party
	House of Hoops party	House of Hoops	02 Dec 2012 16:20:15	aniversary party
	R/GA holiday party	R/GA office	06 Dec 2012 22:06:19	Christmas party
	Arrojo underground show	lower Manhattan	02 Dec 2012 16:18:26	Nick Arrojo speak
	Talib Kweli show	Brooklyn Bowl	02 Dec 2012 21:35:09	night show
	Brooklyn Night Bazaar	Brooklyn	01 Dec 2012 22:40:24	a temporal bazaar

Table 4: Various categories of events detected

[10] Douglas J Leith, Martin Heidl, and John V Ringwood. Gaussian process prior models for electrical load forecasting. *Probabilistic Methods Applied to Power Systems, 2004 International Conference on*, pages 112–117, 2004.

[11] Michael Mathioudakis and Nick Koudas. Twittermonitor: trend detection over the twitter stream. *Proceedings of the 2010 international conference on Management of data*, pages 1155–1158, 2010.

[12] Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. Is it really about me?: message content in social awareness streams. *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, pages 189–192, 2010.

[13] Carl Edward Rasmussen and Christopher K I Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. December 2005.

[14] Tye Rattenbury, Nathaniel Good, and Mor Naaman. Towards automatic extraction of event and place semantics from flickr tags. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 103–110, 2007.

[15] Haggai Roitman, Jonathan Mamou, Sameep Mehta, Aharon Satt, and LV Subramaniam. Harnessing the crowds for smart city sensing. *Proceedings of the 1st international workshop on Multimodal crowd sensing*, pages 17–18, 2012.

[16] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. *Proceedings of the 19th international conference on World wide web*, pages 851–860, 2010.

[17] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. TwitterStand: news in tweets. *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, November 2009.

[18] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. *Proceedings of International Conference on Weblogs and Social Media (ICWSM)*, 2009.

[19] Maximilian Walther and Michael Kaisser. Geo-spatial event detection in the twitter stream. *Advances in Information Retrieval*, pages 356–367, 2013.

[20] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2541–2544, 2011.

[21] Jianshu Weng and B Sung Lee. Event detection in twitter. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM 11)*, 2011.