DeLLe: Detecting Latest Local Events from Geotagged Tweets

Hong Wei[1], Hao Zhou[1], Jagan Sankaranarayanan[1], Sudipta Sengupta[2] and Hanan Samet[1]
[1]Department of Computer Science, University of Maryland, College Park, MD 20742
[2]Amazon Web Services (AWS), Seattle, WA 98101
[1]{hyw,hzhou,jagan,hjs}@cs.umd.edu [2]sudipta@amazon.com

ABSTRACT

Geotagged tweet streams contain invaluable information about the real-world local events like sports games, protests and traffic accidents. Timely detecting and extracting such events may have various applications but yet unsolved challenges. In this paper, we present DeLLe, a methodology for automatically **De**tecting **L**atest **L**ocal **E**vents from geotagged tweets. With the help of novel spatiotemporal tweet count prediction models, DeLLe first finds unusual locations which have aggregated unexpected number of tweets in the latest time period and thereby imply potential local events. Next, DeLLe calculates, for each such unusual location, a ranking score to identify the ones most likely having ongoing local events by addressing the temporal burstiness, spatial burstiness and topical coherence. Furthermore, DeLLe infers an event candidate's spatiotemporal range by tracking its event-focus point, which essentially reflects the most recent representative occurrence site. Finally, DeLLe chooses the most influential tweets to summarize local events and thereby presents succinct but yet representative descriptions. We evaluate DeLLe on the city of Seattle, WA as well as a larger city of New York. The results show that the proposed method generally outperforms competitive baseline approaches.

CCS CONCEPTS

•**Information systems** →*Data streaming; Clustering;*

KEYWORDS

Twitter, Geotagged Tweet Stream, Local Events, Event Detection

ACM Reference format:
Hong Wei, Hao Zhou, Jagan Sankaranarayanan, Sudipta Sengupta and Hanan Samet. 2019. DeLLe: Detecting Latest Local Events from Geotagged Tweets. In *Proceedings of 3rd ACM SIGSPATIAL International Workshop on Analytics for Local Events and News , Chicago, IL, USA, November 5, 2019 (LENS'19),* 10 pages.

1 INTRODUCTION

With people posting what is happening outside in the real world, tweets in Twitter encapsulate invaluable information on real-world events as they break. Accessing news tweets by location is of great interest (e.g.,see [1, 2] which are based on the NewsStand system [3–5]). Geotagged tweets are particularly interesting in the sense that they provide the complement information about the place of interest [6–14], e.g., where the events occur. In this paper, we aim to detect the latest local events from live geotagged tweet streams. A local event is defined as an unusual activity that appears at some specific time and place and also shows topical coherence. For instance, Figure 1 presents some exampling geotagged tweets about a soccer game held in the city of Seattle, WA. Timely discovering such local events has a wide range of applications. For example, people can acquire the latest information about such local activities in their living town, and thereby enhance their daily lives. In such cases, after learning what is happening, the commuters can actively make a decision to bypass the congested road segments or avoid the accident sites .

Figure 1: Examples of geotagged tweets about the soccer game of "Seattle Sounders" vs "D.C. United" at the stadium of "CenturyLink Field" at 7:30 PM, 2017-07-19. All the tweets were located at the stadium of "CenturyLink Field", i.e., the red grid cell in Figure 2a.

It is, however, challenging to detect local events from live geotagged tweet streams. First, detecting local events by capturing unusualness requires considering not only temporal historical patterns but also spatial circumstances. Some studies [15–18] measure the burstiness, intensity of increment in the number of tweets at a place over a short time period, as signals of local events. But burstiness does not always imply the occurrence of a local event. For example, the burstiness of tweets at a shopping mall or a famous coffee bar in the morning is often expected and not unusual. Some work improves this measure to capture temporally routine patterns by gathering time-aware statistics [19]. However, without geographical consideration, occasional nation-wide events may also accumulate a temporally unusual number of tweets at local places. For example, on presidential election nights, one may observe suddenly more tweets all over the places. Second, a local event, as it develops, may receive follow-up updates on its content and may also migrate geographically. For example, when a crime happens at a place, people expect to receive updates as investigation progresses. Another example is that a demonstration protest may follow a route moving from one place to another. Therefore, it is desirable to dynamically and timely monitor and track the development of an ongoing event, and report its latest updates.

LENS'19, Chicago, IL, USA

DOI: 10.1145/3356473.3365188

In this paper, we propose DeLLe to discover, track and describe local events from live geotagged tweets. The contribution of DeLLe lies in its four modules: *seeker*, *ranker*, *expander* and *summarizer*.

Seeker finds unusual locations which exhibit spatiotemporal unusualness with respect to the number of tweets and therefore potentially correspond to local events. For this purpose, seeker employs a novel prediction-based anomaly detection strategy. In particular, seeker first exploits convolutional LSTMs (ConvLSTM [20]) to predict the expected number of tweets in the future, which accounts for not only historical patterns but also neighboring locations. Next, seeker compares the predicted value with the actual number of tweets to determine the existence of unusualness. Unlike previous studies [16, 21] which claim anomalies only based on the local time series data of a location, we also consider the horizontal situation in other places simultaneously to mitigate the effects of global events.

Ranker suppresses the possibly noisy candidates of local events. In practice, not all spatiotemporal burstinesses necessarily correspond to an actual local event. We therefore bring order to the candidates with a ranking procedure by considering temporal burstiness, spatial burstines and topical coherence, and thereby select the top ones likely to be corresponding to the occurrence of local events.

Expander tracks and updates the movement of an ongoing local event in both space and time using event-focus and content similarity. An event focus records its most important site of occurrence at certain time. While the content similarity between the tweets in two nearby locations is used as a measurement to check whether an ongoing local event moves to nearby locations or keeps bubbling up at the same place. In so doing, this module is dynamically monitoring the impact range of a local event.

Summarizer generates an abstract for a detected local event by selecting its most influential tweets. For human consumption, an event should be presented in a succinct description [21] but yet with up-to-date and key information. It is therefore important to choose representative tweets to summarize the detected local events. This module builds tweet authority graphs based on their textual similarities and subsequently runs random walk procedures to select the most influential tweets for events.

2 RELATED WORK

There has been a plural of works on detecting local events using tweets in Twitter. Atefeh [22] and Abdelhaq [23] provide two excellent surveys. In general, existing methods focusing on geotagged tweets can be classified into two strategies: *model dimension extension* and *geographical space tessellation.* Model dimension extension treats location as additional variables to existing models. For example, some studies treat location as latent variables in their generative topic model [24–27]. Location distance between tweets can also be incorporated to measure similarities [28–31] during clustering.

Geographical space tessellation divides space into small and disjoint cells for aggregating geotagged tweets. The motivation is that a local event usually has a limited spatial impact and would fall in the same or nearby cell(s). The grid is the simplest yet most commonly used way of tessellation. although other structures have also been explored including hierarchical triangular meshes [21], pyramid structures [32, 33], Voronoi tessellations [34] and k-d tree [35].

After aggregating tweets to tessellation cells, a simple way for event detection is to examine whether the number of the aggregated tweets or the arriving rate exceeds a certain threshold [15, 19]. This, however, is easily plagued by tweet distribution heterogeneity both temporally and spatially. Thus, various anomaly detection methods have been explored. The core idea is to use previous history of data to build a baseline (or make a prediction) and then compare with the actual value to check for significant discrepancies [16, 17, 21]. For example, TwitInfo [36] uses the weighted average of historical tweet counts to compute the expected frequency of tweets. But sole historical data often neglect the effects exerted by nearby geographical regions. Krumm and Horvitz [21] therefore include features like tweet counts from adjacent regions in their anomaly detection method. Our method is different from the above methods in two senses: 1) our prediction model captures both spatial dependencies and temporal patterns [10]; 2) when claiming an anomaly, we account for not only the history of a location itself but also the situation at other places in the meantime to mitigate the effect of unexpected global events.

The most related work to our task are EvenTweet [16] , Eyewitness [21], GeoBurst [28] and TrioVecEvent [24]. EvenTweet detect events by identifying and clustering temporal bursty keywords. However, using words instead of tweets as clustering elements, this method may group semantically irrelevant words together and in the meantime not sit well with event summarization. Eyewitness [21] discretizes space and time and finds tweet volume spikes as potential local events by comparing the predicted value with the observed value. However, it needs to perform an exhaustive sweep through different space and time pieces and thereby is not easy to modify for online processing. GeoBurst [28] generates candidate events by identifying pivot tweets based on geographical and semantic similarities and ranks them using spatiotemporal burstiness to filter out noisy ones. TrioVecEvent learns multimodal embeddings of tweets to address the information on location, time and text during clustering and is reported to achieve much better accuracy than its baseline approaches. However, neither of these two methods actively performs event detection on a given tweet stream unless a query time window is specified.

Due to the sparsity of geotagged tweets (1%), some methods try to acquire more local tweets by tracking local people [9, 37]. The location information in these methods, however, are usually in a very coarse resolution (e.g., city-level) and rarely used when grouping tweets together. Some methods try to first detect an event and then estimate its location afterwards, e.g., TwitterStand [2]. These methods are different from our focus as we instead try to extract local events from geotagged tweet streams.

3 PRELIMINARIES

3.1 Problem

Given a geotagged tweet stream, our goal is to identify the latest local events. Formally, suppose that t is the current (latest) time point and Δt is a short time interval, we define $\mathcal{D}_t$ to be the geotagged tweet stream up to t, and $\mathcal{D}_{t-\Delta t \to t}$ be the geotagged tweet stream from $t - \Delta t$ to t. In other words, $\mathcal{D}_{t-\Delta t \to t}$ essentially represents the latest geotagged tweets with respect to Δt. For simplicity, a geotagged tweet d can be seen as a tuple $\langle time_d, loc_d, txt_d, user_d \rangle$ in which $time_d$ is the publication time, loc_d is the geographical location (i.e., a pair of lat/long coordinates), txt_d refers to the textual content and $user_d$ is the user posting this tweet. The latest local

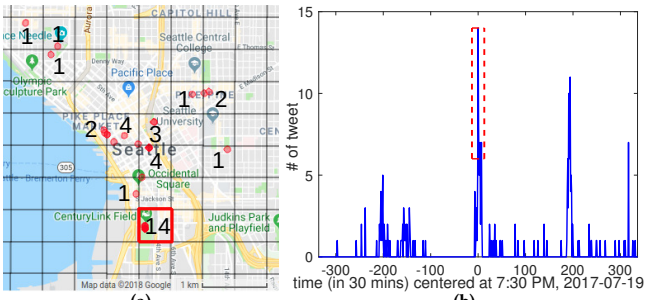


Figure 2: The soccer game in Figure 1 brings about an anomalous amount of tweets both spatially and temporally. (a) Spatial distribution of the tweets around the stadium at 7:30 PM - 8:00 PM. The stadium lies in the red square. Each red dot is a tweet, and the number in a grid cell refers to its number of tweets while an empty cell means no tweets. (b) Temporal distribution of the tweets at the game stadium. The tweets are aggregated every 30 minutes.

event detection problem is then to extract from $\mathcal{D}_{t-\Delta t \to t}$ all possible local events, where each event is a cluster of geographically, temporally and semantically close tweets.

Typically, the occurrence of a local event often brings about an unusually considerable amount of relevant tweets at the happening location for a certain time period. For example, a soccer game started at 7:30 PM at the stadium "CenturyLink Field" near the center of Seattle, yielding many tweets with keywords "Sounders", "soundersfc" and "CenturyLink Field" etc (Figure 1) at that location during the game. Figure 2 shows that an unusual amount of such tweets were observed both geographically and temporally.

Motivated by the above observation, we propose a prediction-based method for detecting the latest local events, called DeLLe. The key idea of DeLLe is to first detect spatiotemporal unusualness as possible candidates of local events and then select the ones that most likely corresponding to local events.

3.2 System Overview

Figure 3 demonstrates DeLLe's overall design. DeLLe can work in two modes: batch mode and online mode. The major difference is that the batch mode exploits a disjoint discretization in the time dimension while the online mode utilizes a continuous sliding time window and correspondingly a set of updating modifications for online processing. We will detail these modifications in Section 5.

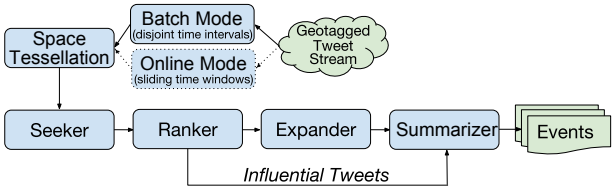


Figure 3: System Overview

We utilize a uniform grid to tessellate the spatial region into squares of size $\Delta l \times \Delta l$, where Δl is the side length of the square. Although more complex tessellation structures [21, 33, 38] have been explored, grid tessellation is the simplest yet most commonly used way of subdividing geographical space [16, 39–42]. More importantly, it enables us to exploit state-of-the-art spatiotemporal tweet count prediction model [10] by treating each grid cell as a pixel and therefore the whole grid as image-like data. In reality, local events may not fall neatly on the grid cell boundaries. Therefore, we propose a module of *expander* to connect nearby cells which share similar content to alleviate that issue.

After discretizing space, the tweets are subsequently fed into a pipeline of four modules: *seeker*, *ranker*, *expander* and *summarizer*. *Seeker* finds spatiotemporal unusualness in the number of tweets as potential candidates of local events. *Ranker* selects which set of unusualness found by the *seeker* are most likely to be local events. Afterwards, the *expander* tries to infer a local event's span in both time and space under the metric of semantical similarity. At last, the *summarizer* generates an abstract for a detected local event by selecting the latest top influential tweets.

4 THE BATCH MODE

In this section, we present the workflow of DeLLe in its batch mode. In order to detect the local events from $\mathcal{D}_{t-\Delta t \to t}$, we discretizes the geotagged tweet stream into a set of disjoint intervals, i.e.,$\{\cdots [t - 2\Delta t, t - \Delta t), [t - \Delta t, t)\}$. In the following, we explain how to do tweet count prediction, unusualness detection, event expansion, and summarization for this time series of tweets.

4.1 Seeker

After tessellating the space into an $M \times N$ grid and time into periods of Δt, the task of seeker is to identify grid cells that show an unusual aggregation of tweets in latest geotagged tweet stream $\mathcal{D}_{t-\Delta t \to t}$ or $\mathcal{D}_T$ where T denotes the last time interval of length Δt.

4.1.1 ***Tweet Count Prediction.*** The goal of tweet count prediction is to use previously historical tweet count data in a local region to forecast on the number of tweets to appear in the next time step [10]. On an $M \times N$ grid map, the tweet count values in the grid cells at time step τ can be written in a tensor $\mathbf{X}_\tau \in \mathbb{R}^{M \times N}$ where $\mathbf{X}_\tau(m, n)$ is the tweet count in the grid cell (m, n) at time step τ. Therefore, the prediction problem is formulated as follows:

Definition 4.1. The tweet count prediction problem $\mathcal{P}$ is to generate a prediction $\mathbf{Y}_T$, which is an estimation of $\mathbf{X}_T$, given a list of historical observations $\{\mathbf{X}_\tau | \tau = 0, \cdots, T-1\}$.

Making high-quality predictions of tweet count in a region is challenging due to complex spatial and temporal dependencies. Additionally, there are studies pointing out that spatiotemporal data also has a certain periodic pattern [21, 43], which indicates that we should also capture the periodic time-varying changes in tweet volume. For instance, if there are consistent bursty tweets at a coffee shop (e.g., Starbucks) in the morning, it should not be mistakenly reported as unusual. The advances in deep learning have motivated a few recent studies to introduce deep neural networks into modeling spatiotemporal data for better capturing spatial and temporal dependences [43, 44]. In this paper, we utilize a residual Convolutional LSTM (ConvLSTM [20]) based prediction model [10], which is reported to have state-of-the-art accuracy.

We now give a brief introduction to this tweet count prediction model [10]. Figure 4 illustrates the structure of the neural network model. Zhang et al. [43, 44] pointed out that making predictions on spatiotemporal data relies on not only the observations of *recent time* but also those in *near history* and *distant history*, and model these temporal dependencies as temporal *closeness*, *period* and *trend*. A similar observation on tweet count data is also found [10]. Inspired by this, the model consists of three main branches: *closeness*, *period* and *trend*, to incorporate temporal pattern information at different scale in tweet data, together with a meta-data branch to

Table 1: List of main notations in Section 4

Δl	the side length of a grid cell
$t, \Delta t$	current time, length of time interval
τ, T	time step, the latest time step
$\mathcal{D}_t, \mathcal{D}_T$	tweet stream up to t, tweet stream during T
$\mathbf{X}_\tau, \mathbf{Y}_\tau$	tweet count at τ, prediction of tweet count at τ
(m,n)	a grid cell in an $M \times N$ grid map
c, p, q	closeness, period, trend
$\mathbf{E}_T, \mathcal{E}_T$	prediction error, a list of prediction errors up to T
$k_{\Delta \mathbf{E'}_T}$	unusualness threshold
$\mathcal{Y}_T(m,n)$	a list of historical predictions on (m,n)
$\mathcal{W}_T(m,n)$	the set of keywords in (m,n) at T
$SDD_T^w(m,n)$	spatial density distribution of word w in (m,n) at T
$TS(d', d'')$	topical similarity between tweet d' and d''
$TB_T(m,n)$	temporal burstiness of cell (m,n) at T
$SB_T(m,n)$	spatial burstiness of cell (m,n) at T
$TC_T(m,n)$	topical coherence of cell (m,n) at T
$TEC_T(m,n)$	the set of Temporal Expansion Cells of cell (m,n)
$SEC_T(m,n)$	the set of Spatial Expansion Cells of cell (m,n)
k	the number of tweets for event summarization as well as in calculating topical coherence

capture features such as time-of-day (e.g., in minutes), day-of-week.

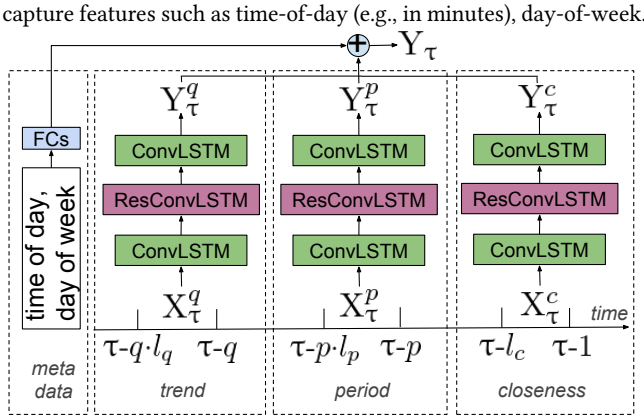


Figure 4: Twitter Count Prediction Model. ResConvLSTM: Residual ConvLSTM block; FCs: Fully-Connected Layers, i.e. Dense layers.

To be specific, the *closeness* sequence is a list of l_c continuous tweet count values right before the current time step and is denoted by $X_\tau^c = [X_{\tau-l_c} \quad X_{\tau-(l_c-1)} \quad \cdots \quad X_{\tau-1}]$. The *period* sequence is a l_p-long list of historical tweet count values periodically sampled every p interval: $X_\tau^p = [X_{\tau-p\cdot l_p} \quad X_{\tau-p\cdot(l_p-1)} \quad \cdots \quad X_{\tau-p\cdot 1}]$. Similarly, the *trend* sequence is a l_q-long list of historical tweet count values periodically sampled but every q interval: $X_\tau^q = [X_{\tau-q\cdot l_q} \quad X_{\tau-q\cdot(l_q-1)} \quad \cdots \quad X_{\tau-1\cdot q}]$. In practice, p is set to a duration of one-day to capture daily periodicity and q to one-week to reveal weekly trend. Each of X_τ^c, X_τ^p and X_τ^q is separately fed into three designated neural networks which share the same structure but with different weights, to generate separate predictions Y_τ^c, Y_τ^p and Y_τ^q, respectively. At last, a parametric-matrix-based fusion is applied to combine the 3 prediction results, together with the meta-feature of time, to generate the final prediction. In the following, we will briefly explain the key stacking blocks in Figure 4, including the ConvLSTM layer, the ResConvLSTM block and the FC layer.

(1) *ConvLSTM layer.* The Convolutional Long Short-Term Memory (ConvLSTM) was first proposed in [20] and is an extension of FC-LSTM [45]. ConvLSTM innovatively uses a convolution operator in the state-to-state and input-to-state transitions, and thereby overcomes the traditional LSTM's ignorance of spatial information when the temporal sequence is multi-dimensional.

(2) *ResConvLSTM block.* By adding skip connections directly from the output of lower layers to the input of higher layers, residual networks [46] have proven to be effective to alleviate the problem of vanishing gradient in deeper networks during the training process and achieved significantly better performance in many applications. As shown in Figure 5, the tweet count prediction model also assemblies a residual block of ResConvLSTM using ConvLSTM layers. This is a key difference from ST-ResNet [44] which uses a regular convolutional layer.

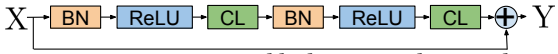


Figure 5: ResConvLSTM block. BN: Batch Normalization; ReLU: Rectifier Linear Unit; CL: ConvLSTM

(3) *FC layer.* To help capture the regular time-varying changes, meta-data features such as time-of-day, day-of-week are also hooked in the model by stacking two fully-connected layers. The first is an embedding layer for features and the second maps from low to high dimensions to make the output have the same shape as the target [44].

In Figure 4, ConvLSTM layers usually have 32 hidden states except for the output which has 1 to reflect the tweet count value. In addition, the size of the filter in ConvLSTM is set to 3×3, because the spatial correlation of tweet count data is quite local, i.e., the number of tweets in a grid cell is correlated with the ones in the nearby grid cells instead of grid cells farther away [10].

4.1.2 ***From Prediction To Unusualness.*** We define the prediction error to be $\mathbf{E}_T = \mathbf{Y}_T - \mathbf{X}_T$, where $\mathbf{X}_T$ is the latest tweet count on a spatial $M \times N$ grid and $\mathbf{Y}_T$ is the prediction of $\mathbf{X}_T$. $\mathbf{E}_T(m,n)$ indicates the prediction error of the grid cell (m,n). Intuitively, a significant negative $\mathbf{E}_T(m,n)$ indicates a local event as there were many more tweets than usual. Following [21], we define precision of our prediction model to be $\sigma_{\mathcal{E}_T}$, where $\sigma_{\mathcal{E}_T}(m,n)$ is the standard deviation of the grid cell (m,n) w.r.t. its history of prediction errors $\mathcal{E}_T(m,n) = \{\cdots \mathbf{E}_{T-1}(m,n), \mathbf{E}_T$. To account for the precision of the prediction model, we re-define the prediction error as:

$$\mathbf{E'}_T = \mathbf{E}_T \oslash \sigma_{\mathcal{E}_T} \tag{1}$$

where $\oslash$ denotes the element-wise division operation.

To detect unusual grid cells, we utilize an image restoration framework called Deep Image Prior [47]. Our intuition is that unusualness in $\mathbf{E'}_T$ is like spike noise in an image, and Deep Image Prior can be used to denoise corrupted images without prior knowledge of training data. Suppose that $\mathbf{E''}_T$ is the restored image of $\mathbf{E'}_T$, and $\Delta\mathbf{E'}_T = \mathbf{E''}_T - \mathbf{E'}_T$, we claim a grid cell (m,n) is unusual if

$$|\Delta\mathbf{E'}_T(m,n) - \mu_{\Delta\mathbf{E'}_T}| \geq k_{\Delta\mathbf{E'}_T} \cdot \sigma_{\Delta\mathbf{E'}_T} \tag{2}$$

where $\mu_{\Delta\mathbf{E'}_T}$ and $\sigma_{\Delta\mathbf{E'}_T}$ are the mean and standard deviation of grid cells in $\Delta\mathbf{E'}_T$, respectively. $k_{\Delta\mathbf{E'}_T}$ is a predefined threshold for determining the unusualness of a grid cell. Different from [21], our approach accounts for both history of a grid cell and information of other cells on the whole region when detecting unusualness in a location. This is important in differentiating global events which might cause an unusual number of tweets on a local grid cell.

4.2 Ranker

The *seeker* module described above outputs a set of unusual grid cells. In this section, we make a ranking of these unusual locations to identify the top ones that are most likely corresponding to the occurrence of local events, by addressing temporal burstiness, spatial burstiness and topical coherence.

4.2.1 ***Temporal Burstiness.*** For a grid cell (m, n), suppose that $\mathcal{Y}_T(m, n)$ represents a history of estimations on its number of tweets up to the time step T, and is defined as:

$$\mathcal{Y}_T(m, n) = \{\cdots \mathbf{Y}_{T-1}(m, n), \mathbf{Y}_T(m, n)\} \tag{3}$$

Then we use z-score to quantify the grid cell (m, n)'s temporal burstiness [28] at T, denoted as $TB_T(m, n)$ and defined as:

$$TB_T(m, n) = \frac{\mathbf{X}_T(m, n) - \mu_{\mathcal{Y}_T(m,n)}}{\sigma_{\mathcal{Y}_T(m,n)}} \tag{4}$$

where $\mu_{\mathcal{Y}_T(m,n)}$ and $\sigma_{\mathcal{Y}_T(m,n)}$ are the mean and standard deviation of $\mathcal{Y}_T(m, n)$, respectively. Recall that $\mathbf{X}_T(m, n)$ is the actual number of tweets in grid cell (m, n) at time step T.

4.2.2 ***Spatial Burstiness.*** Given a grid cell (m, n), the spatial burstiness is measured by the spatial density distribution of keywords of the tweets in (m, n). The intuition is that a low spatial density distribution means that the keyword is widely spread over space and a high distribution means that the keyword occurs only at a few locations. Therefore, the keywords in local events should have higher spatial density distribution to be spatially bursty.

Suppose that $\mathcal{D}_T(m, n)$ is the tweet set in grid cell (m, n) at T, and $\mathcal{W}_T(m, n)$ is the set of keywords (e.g., after removing stop words) in (m, n), i.e., $\mathcal{W}_T(m, n) = \{w \mid w \in txt_d \text{ and } d \in \mathcal{D}_T(m, n)\}$. Let $SDD_T^w(m, n)$ be the spatial density distribution of keyword w in grid cell (m, n) at T, i.e.,

$$SDD_T^w(m, n) = \frac{\#\text{ of } w \text{ in grid cell } (m, n)}{\sum_{(m', n') \in M \times N} \#\text{ of } w \text{ in grid cell } (m', n')} \tag{5}$$

We now define the spatial burstiness of grid cell (m, n) as:

$$SB_T(m, n) = \sum_{w \in \mathcal{W}_T(m,n)} SDD_T^w(m, n) \tag{6}$$

4.2.3 ***Topical Coherence.*** The topical coherence is to capture the semantical similarity of tweets in a grid cell. In other words, the tweets posted on the same event should be discussing similar content and probably using similar vocabularies. Twee2Vec [48] learns the vector-space representations of tweets using a character-based bi-directional recurrent neural network model, and has been demonstrated to have good performance in the application of clustering semantically similar tweets together [49]. To measure the topical similarity between tweets, we use Tweet2Vec implementation[1] to encode a textual tweet in character sequence to a vector embedding with a default dimension size of 500.

Let $TS(d', d'')$ be the topical similarity between tweets d' and d''. To measure the topical coherence of the tweets in cell (m, n), we construct a graph, called *Tweet Influence Graph.*

Definition 4.2. (Tweet Influence Graph). The tweet influence graph on the grid cell (m, n) at T, is an undirected graph $G_T = (V_T, E_T)$ where V_T is the set of all tweets in $\mathcal{D}_T(m, n)$, E_T is the set of edges between tweets, and the weight of an edge between d' and d'' is their topical similarity $TS(d', d'')$.

We now employ PageRank [50], a random walk procedure, on the tweet influence graph to bring orders to the influence of tweets in $\mathcal{D}_T(m, n)$ and thus identify the top k tweets with the most influence, denote by $\mathcal{D}_T^k(m, n)$. The *topical coherence* is thus defined as:

$$TC_T(m, n) = \frac{\sum_{d' \in \mathcal{D}_T^k(m,n), d'' \in \mathcal{D}_T^k(m,n)} TS(d', d'')}{k^2} \tag{7}$$

[1] https://github.com/vendi12/tweet2vec_clustering

The rationale is that if the tweets in $\mathcal{D}_T(m, n)$ are about the same local event, the most topically influential tweets should have higher topical similarity between each other. One may point out that such a topical coherence measurement would suppress a grid cell having multiple topically unrelated ongoing events. We argue that such a case is very rare with a fine space and time discretization.

4.2.4 ***Ranking Function.*** As the final step, we now define the ranking score of the grid cell (m, n) by aggregating its *temporal burstiness*, *spatial burstiness* and *topical coherence*, after rescaling them to $[0, 1]$ with respect to other grid cells:

$$\mathcal{R}_T(m, n) = TB'_T(m, n) \cdot SB'_T(m, n) \cdot TC'_T(m, n) \tag{8}$$

where $TB'_T(m, n) = (TB_T(m, n) - TB_T^{min})/(TB_T^{max} - TB_T^{min})$ with TB_T^{max} and TB_T^{min} being the maximum and minimum of topical burstiness among all grid cells at T. Spatial burstiness and topical coherence are rescaled in the same way, receptively.

4.3 Expander

Suppose that we choose the top K unusual grid cells after ranking at T, and claim that they are the candidates most likely to be local events in $\mathcal{D}_T$. In reality, different local events might have different spatial and temporal ranges, e.g., spanning over a larger region than the grid cell size $\Delta l * \Delta l$ or for a longer duration than the time discretization interval Δt. We therefore, in this section, try to infer the spatiotemporal range of these local event candidates.

The basic idea of expander is to connect (spatially or temporally) nearby grid cells if they share similar content. As presented in Algorithm 1, the expansion consists of two parts: temporal expansion and spatial expansion. The temporal expansion checks whether the occurrence of previous event candidates continues to the present, and updates them if so. The spatial expansion examines whether nearby grid cells are relevant to the same event.

During the expansion, for each event candidate, we maintain a grid cell as its *event-focus* grid cell. The event-focus grid cells are initially set to be the most unusual cells (i.e., the top ranking cells in Equation 8). As time proceeds, the event-focus grid cell of an event might stay at the same grid cell (e.g., a sit-down protest), or move to another one (e.g., a demonstration protest), or simply no longer exists (e.g., the ending of an event). Meanwhile, new event-focus grid cells might join as well if new events happen. Note that during the spatial expansion, several event-focus grid cells might exist adjacently and need to be merged if they are about the same content. For a given cell (m, n), we denote by $SEC_T(m, n)$ (Spatial Expansion Cells) the cells for it to examine for spatial expansion at T, and similarly $TEC_T(m, n)$ (Temporal Expansion Cells) for temporal expansion. $SEC_T(m, n)$ and $TEC_T(m, n)$ are defined as follows.

$$\begin{aligned} SEC_T(m, n) &= \{(m \pm i, n \pm j)_T \mid i, j \in \{-1, 0, 1\}\} \setminus \{(m, n)_T\} \\ TEC_T(m, n) &= SEC_T(m, n) \cup \{(m, n)_T\} \end{aligned} \tag{9}$$

The spatial range for expansion is currently set to adjacent cells incident at an edge or vertex and can extend further if necessary.

Whether or not two adjacent grid cells are connected depends on their content similarity. We treat each grid cell containing its tweets as a document and thus build a term-document matrix. In this matrix, each row represents a token (non-stop words) in tweets, each column represents a document, i.e., a grid cell, and each element can refer to the token frequency (or TF-IDF) per document. Next, the content similarity between two grid cells can be calculated

Algorithm 1: Expander

Input: C_{T-1} — the set of *event-focus* grid cells at time $T-1$, C'_T — the top K ranking grid cells at time T (i.e., the newly identified *event-focus* grid cells), ε_{cs} — content similarity threshold.

Output: The updated *event-focus* grid cells C_T at time T

/* Temporal Expansion */

foreach *event-focus grid cell* $(m,n)_{T-1} \in C_{T-1}$ **do**

  // $TCC_T(m,n)$ are the grid cells at time T
  // that are temporally connected to $(m,n)_{T-1}$

  $TCC_T(m,n) \longleftarrow \{\, c \mid c \in TEC_T(m,n), \varepsilon_{cs} \leq \texttt{ContentSimilarity}(c,(m,n)_{T-1})\}$;

  if $TCC_T(m,n) \neq \emptyset$ **then**

    $c' = \underset{c \in TCC_T(m,n)}{\text{argmax}}\ \texttt{ContentSimilarity}(c,(m,n)_{T-1})$;

    // c' is now a new event-focus grid cell
    // transited from $(m,n)_{T-1}$

    $C'_T \longleftarrow C'_T \bigcup \{c'\}$;

/* Spatial Expansion */

foreach *event-focus grid cell* $(m,n)_T \in C'_T$ **do**

  // $SCC_T(m,n)$ are the grid cells at time T
  // that are spatially connected to $(m,n)_T$

  $SCC_T(m,n) \longleftarrow \{\, c \mid c \in SEC_T(m,n), \varepsilon_{cs} \leq \texttt{ContentSimilarity}(c,(m,n)_T)\}$;

  // If $(m,n)_T$ is spatially connected to other
  // event-focus grid cells, merge them.

  if $SCC_T(m,n) \bigcap C'_T \neq \emptyset$ **then**

    $Temp = (SCC_T(m,n) \bigcap C'_T) \bigcup \{(m,n)_T\}$;

    $c' = \underset{c \in Temp}{\text{argmax}}\ \texttt{TopicalCoherence}(c)$;

    $C'_T \longleftarrow (C'_T \setminus Temp) \bigcup \{c'\}$;

return C'_T

by their corresponding column vectors under the metric of cosine similarity. More advanced document similarity techniques such as Latent Semantic Analysis (LSA) [51] may further be applied on the term-document matrix to measure the similarities between documents at a lower rank. It is, however, usually a time-consuming process due to the introduction of Singular Value Decomposition (SVD). For approximation as well as efficiency, we limit each column vector to contain the information on its most frequent k_{cs} tokens during cosine similarity calculation, where k_{cs} is a predefined value.

4.4 Summarizer

The module of *summarizer* selects the most representative tweets from a cluster of tweets in an event, and thereby produce a succinct description. When summarizing an event across several time steps, the tweets at the latest time step T are preferred to earlier ones in order to reflect the newest dynamic updates on events.

The general idea of event summarization expects that the tweets associated with the event demonstrate a meaningful description of the event for human consumption [21]. For this purpose, we exploit the most influential tweets in the grid cells. As discussed in Section 4.2.3, $\mathcal{D}_T^k(m,n)$ consists of the top k tweets with the most influence at the grid cell (m,n). The summarization works as follows. First, if an event is limited to one grid cell, then its top k tweets are the summarization set of tweets. i.e., $\mathcal{D}_T^k(m,n)$. Second, if an event impacts several grid cells, then we look at the top grid cells with the largest topical coherence scores defined in Equation 7 to select which tweets to form the summarization. To be specific, suppose that an event e's spatial impact at T consists of a set of grid cells, denoted by SI_T^e. The subset of SI_T^e used for summarization is defined as:

$$SISum_T^e = \{(m^i,n^i) |\ i = 1 \cdots k'\} \tag{10}$$

where $1 \leq k' \leq k$, specifying that the summarization tweets are only from the top-k' grid cells with the largest topical coherence scores in SI_T^e. The topical coherence score in each grid cell weighs how many tweets it will contribute to the summarization. For example, let $TC_T(m^i,n^i)$ be the i-th largest topical coherence score, then the number of tweets its grid cell (m^i,n^i) should contribute is:

$$k^i = round(\frac{TC_T(m^i,n^i)}{\sum_1^{k'} TC_T(m^i,n^i)}) * k. \tag{11}$$

And such k^i tweets come from the top k influential tweets in (m^i,n^i), denoted by $\mathcal{D}_T^{k^i}(m^i,n^i)$. Therefore, the summarization tweet set is:

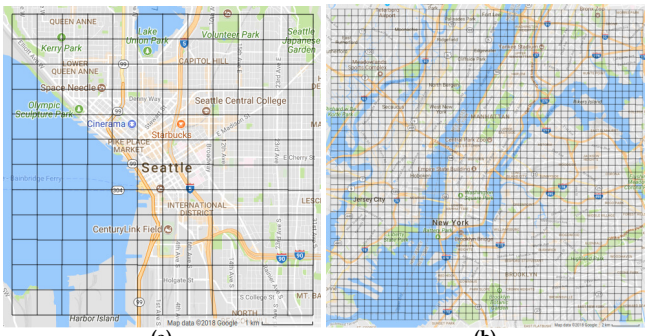
$$SumTweets_T^e = \bigcup_i^{k'} \mathcal{D}_T^{k^i}(m^i,n^i) \tag{12}$$

5 ONLINE MODIFICATIONS

In this section, we present the modifications that allow DeLLe to process tweets in an approximately online way. The major modification is to utilize a continuous moving sliding window instead of disjoint intervals of time. For example, suppose that the current time is t, the window length is Δt, and the current sliding window is at $[t-2\Delta t, t-\Delta t)$. Then the next sliding window to consider in the online processing is at $[t-2\Delta t+\Delta s, t-\Delta t+\Delta s)$, instead of $[t-\Delta t, t)$ as in the batch mode. Δs denotes the moving step length in the sliding window. In what follows, we describe the changes to the modules in the batch mode needed to enable online processing.

Seeker In the online processing, with a small moving step Δs, two consecutive sliding windows mostly overlap each other and might present little difference. Consequently, if the prediction model takes the previous consecutive windows as the input, it probably generates a prediction very similar to the current sliding window and thus fails to detect anomalous aggregation of tweets. Therefore, to make predictions in the online processing, we still use the data in the previously disjoint time interval as the input. For example, the last time interval in the *closeness* sequence used for predicting the tweet count at $[t-\Delta t, t)$ is $[t-2\Delta t, t-\Delta t)$, instead of $[t-\Delta t-\Delta s, t-\Delta s)$, which is the last sliding time window.

Ranker As the sliding window proceeds, the tweets in the grid cells may also change, as well as the scoring factors in the ranking Equation 8. Recalculating some scores like *spatial burstiness* and *topical coherence* from scratch can be very time-consuming. Therefore, we leverage historical results to update the changes caused by inserting new tweets as well as deleting old tweets. For example, in updating the *spatial burstiness* scores, the system maintains a keyword list which specifies the frequency of a keyword's appearance in each grid cell. Thus, only simple addition or subtraction is necessary for updating frequencies of words. The more complex changes come from updating the scores of *topical coherence* as the tweet influence graph may evolve when inserting new tweets or deleting obsolete tweets. To handle such changes, we exploit OSP [52], a fast random walk algorithm on dynamic graphs using Offset Score Propagation. The core idea of OSP is to first calculate an offset seed vector based on the adjacency difference between



(a) (b)

Figure 6: (a) 12×12 grid map in Seattle. (b) 46×46 grid map in NYC.

old and new graphs.Next, such a seed vector is propagated across the new graph, resulting in offset scores. Finally, OSP adds up the old and offset random walking scores to get the final scores.

Expander The most time-consuming part in this module is calculating the content similarities between grid cells using their most frequent keywords, which may change as news tweets come in or old tweets go away. For a fast implementation, each grid cell maintains a local heap to track the top frequent keywords in it.

Summarizer The summarizer is easy to modify for online processing because the topical coherence of each grid cell and its most influential tweets have already been calculated in the modified ranker module. Therefore, the essential task is to, for each event, maintain a list of top-k' grid cells with the largest topical coherence scores, by using a priority queues.

6 EVALUATION

DeLLe is implemented in Python and evaluated on a computer with an Intel Xeon E5 CPU, an Nvidia Quadro P6000 GPU and 64GB memory. The tweet count prediction model is built using Keras [53].

6.1 Experimental Settings

6.1.1 Datasets. The evaluation is performed on two sets of geotagged tweets collected from 2015-07-09 to 2017-07-23 in two cities: Seattle, WA (SEA) and New York City (NYC) [10]. Their geographical regions are two bounding boxes spanning from [47.579784, -122.373135] to [47.633604, -122.293062] in SEA, and from [40.647984, -74.111093] to [40.853945, -73.837472] in NYC as illustrated in Figure 6. The total number of tweets after removing spam tweets [10], is 756, 457 and 9, 353, 721, respectively. We take the data from 2017-06-23 to 2017-07-23 for testing and local event detection, and its previous data for training the tweet count prediction model.

6.1.2 Baseline Approaches. The baseline approaches are below:

- EvenTweet [16] first identifies temporal bursty keywords and spatial local keywords and then clusters them to find local events.
- Eyewitness [21] finds tweet volume spikes in discretized time and space as potential local events by comparing the actual number of tweets with the predicted value using a regression model.
- GeoBurst [28] first generates candidate events by seeking pivot tweets based on geographical and semantic similarities and then ranks them with spatiotemporal burstiness to remove noisy ones.
- TrioVecEvent [24] first learns multimodal embeddings of tweets on the domains of location, time, and text and then uses a Bayesian mixture clustering model to find event candidates.

6.1.3 Parameter Settings. We run DeLLe in its batch mode by default and will evaluate its difference from the online processing in Section 5. The major parameters in DeLLe are set as follows. For space and time, we set the side length of grid cells Δl = 500m and the length of time interval Δt = 30 minutes (by dividing a natural integral hour into two intervals) since such values provide fine enough resolution for local event detection as well as yield good performance for tweet count prediction [10]. As a result, we have a 12×12 grid map in SEA and 46×46 in NYC. For the moving step length in sliding windows, we set Δs = 5 minutes, which is long enough for the online processing latency in our system. In the seeker module, we set the length of *closeness*, *period* and *trend* to l_c = 3, l_p = 1 and l_q = 1 as in [10] because such a setting achieves the best prediction accuracy. We set the threshold for determining the unusualness of a grid cell $k_{\Delta \mathbf{E}'_T}$ = 3, a commonly used value for anomaly detection. As for the PageRank procedure to calculate topical coherence in the ranker module, we use the default damping factor 0.8 and run 20 iterations in all cases. After tuning, in calculating content similarity between grid cells for expansion, we set the number of frequent tokens k_{cs} to 5 and the content similarity threshold ε_{cs} to 0.7. We set k = 5 as the number of the most influential tweets used for calculating the topical coherence as well as the number of tweets used to summarize a local event [21, 28]. In each time interval, we select at most K = 5 unusual grid cells as the local event candidates. Because not every time interval does necessarily have K local events happening, we apply a simple heuristic for suppressing the negative candidates. It removes grid cells having too few users (i.e., less than 5) or having a topical coherence score less than 0.8, which is a suggested lower bound for tweet clustering using Tweet2Vec [48, 49]. For fairness, we also similarly filter out the event candidates with less than 5 users for the baseline approaches as well in the evaluation.

EvenTweet takes the same space partition as in DeLLe and similarly selects the top K local event candidates. Since each event in EvenTweet is a cluster of keywords instead of tweets, we use the implementation in [28] to retrieve the top k representative tweets.

Eyewitness exhaustively sweeps through a set of different space and time discretization and is unsuitable for processing live tweet streams. We ease its settings by using the same space and time discretizations in the batch mode of DeLLe. To select the top K local event candidates, we rank them by the prediction error divided by the standard deviation of the error of its regression function, which has shown to be an important feature in classifying events to be postivie or negative [21]. After that, each local event is represented by by choosing k = 5 tweets with the highest frequency words.

For GeoBurst and TrioVecEvent, we adopt their default parameter settings and implementations in [28] and [24], respectively. Since both methods require an input time window to query the occurrence of local events, we set it as a list of disjoint Δt-size windows like the time discretization in DeLLe's batch mode and choose the top K candidates for comparison. Note that TrioVecEvent also classifies an event candidate to be true or false, we therefore use the spatial deviation (i.e., lat/long deviations, which are the two most important features in their classifier) to rank local events.

6.2 Illustrative Cases

We select several positive and negative examples of local event detection and present them in Figure 7 and Figure 8, respectively. Each example is described by 5 representative tweets with locations plotted as red circles in the accompanying maps. Ahead of each

(a) A baseball game of Yankees-Mariners at Safeco Field (2017-07-20 7:00 PM).

(b) NYC Pride March traversing down Fifth Avenue (2017-06-25 10:30 AM).

Figure 7: Examples of true local events. The left is in Seattle, WA, and the right is in New York City, respectively.

(a) People talking about food near the Space Needle (2017-07-22 4:30 PM).

(b) People waiting for 4th of July fireworks at East River (2017-07-04 5:00 PM).

Figure 8: Examples of false local events. The left is in Seattle, WA, and the right is in New York City, respectively.

tweet is its publisher's username. Note that multiple tweets may reside at the same location causing overlapping and dark red circles.

Figure 7 illustrates two positive local events reported in DeLLe. Figure 7a is about a baseball game between the Yankees and the Mariners held at the Safeco Field in Seattle. Figure 7b is about NYC Pride March 2017 traversing southward down Fifth Avenue in New York City. Those two events are very demonstrative as examples of local events because they have exhibited the necessary properties DeLLe wants to capture: spatiotemporal unusualness regarding the number of tweets at a local place and topical coherence regarding the content of aggregated tweets. The tweets selected to describe the events are also representative to convey the necessary information. It is worth mentioning that the tweets in Figure 7a fall closely to the common border of two neighboring grid cells. The expander module in DeLLe effectively captures this case by connecting spatiotemporally adjacent grid cells sharing similar content. These two examples also appeared in the baseline approaches.

Figure 8 presents two cases of negative local events reported in EvenTweet and Eyewitness, respectively. Figure 8a refers to an activity about people talking food near the Space Needle in Seattle, WA. EvenTweet reported this activity as a local event since it finds some spatiotemporal bursty keywords like "Bite". This is because, when the day comes around dinner time, that area seems to be a popular place for people to eat and thereby aggregates tweets with similar keywords about food. Likewise, GeoBurst also falsely reported a related geo-topic cluster because it groups together tweets mentioning similar keywords on the topic and locating geographically closely. Although such an activity may attract enough tweets at a high rate at certain time (e.g., dinner time in the example), it usually follows a periodic daily pattern and does not reflect any unusual event. Neither Eyewitness or our method DeLLe reported this activity because both of them take routine patterns into consideration. Similarly, TriVecEvent classified it as a non-local event too. This is because its multimodal embedding model also addresses the effect of time in tweets and unveils typical words in different regions and time periods.

Figure 8b is an example of negative local event reported in Eyewitness. It is about people waiting for the 4th of July fireworks show at East River Ferry Dock in New York City. This is more like a national event in the United States because fireworks show on Independence Day may happen at different places in a nationwide scale. When it comes close to the evening, one may expect that tweets about fireworks suddenly increase all over the country. Both GeoBurst and TrioVecEvent reported this nationwide event too. This is because such an event is also geographically compact and more importantly, semantically coherent. In contrast, we do not find the occurring grid cell of this event ranked in the top unusual grid cell candidates in our method DeLLe. There are three reasons behind this. First, the likelihood of unusualness in this grid cell was not high considering that other places were experiencing similar burstiness in tweet volume. Second, the spatial burstiness was not strong either because similar keywords were being used everywhere. Third, the topical coherence in this grid cell deteriorated due to the presence of lots of other tweets like "@511NY Cleared: Incident on #ServiceBus at Midtown". EvenTweet did not report this event either because the keyword like "fireworks" and "july4th" appeared adequately in other regions too and thus was considered not to be local to this event's occurring site.

6.3 Quantitative Analysis

Table 2: Comparison results using Precision, Recall and F-Score.

Method	Seattle, WA				NYC			
	#	P	R	F	#	P	R	F
EvenTweet	354	0.391	0.390	0.390	1665	0.146	0.131	0.138
Eyewitness	273	0.769	0.593	0.670	1204	0.614	0.398	0.483
GeoBurst	354	0.517	0.517	0.517	1665	0.203	0.182	0.192
TrioVecEvent	240	0.858	0.582	0.694	1214	0.704	0.461	0.557
DeLLe	269	**0.862**	**0.655**	**0.745**	1128	**0.741**	**0.450**	**0.560**

6.3.1 Effectiveness. We first evaluate the different local event detection methods using *precision*, *recall* and *f-score*. For precision, we recruited 3 volunteers to individually judge the detected events

and collect the results using the strategy of majority votes[2]. In lack of groundtruth on the set of events happened in the real world, we build a pseudo groundtruth by assembling a set of distinct true positive local events reported in different methods to calculate the recall and f-score. The comparison results are listed in Table 2. It shows that DeLLe outperforms baseline approaches in most cases. In particular, a significant improvement is observed over EvenTweet and GeoBurst. DeLLe also achieves comparatively better results to Eyewitness, showing the effectiveness of its unusualness detection and consideration of topical coherence. We notice that TrioVecEvent outperforms all other methods except for the proposed one, showing its effectiveness of multimodal embedding of location, time and text information in tweets.

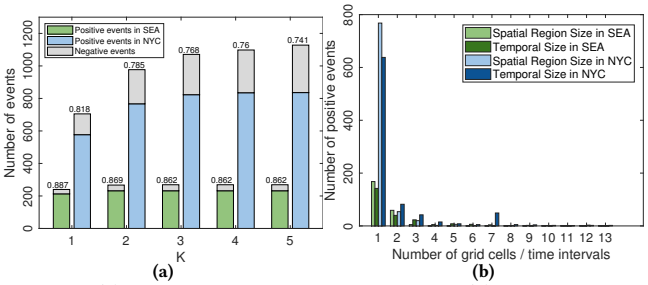


Figure 9: (a) Precision with different K values. (b) Temporal span and spatial region size of positive local events in DeLLe.

To evaluate the sensitivity of K, Figure 9a illustrates the number of positive local events out of the total detected ones in DeLLe when K varies. The decimal number above each bar represents the precision. In general, the precision decreases as K increases because a larger K likely outputs more negative local events, even thought it may also give more positive ones. We notice that the precision and the number of positive local events nearly maintain the same in SEA after $K = 2$ and NYC after $K = 3$.

Figure 9b plots the distributions of positive local events in DeLLe regarding the temporal span (i.e., number of time intervals) and spatial region size (i.e., number of grid cells). The results show that majority of the events fall within one single time interval and one grid cell. This validates our settings in the time and space discretization.

6.3.2 Efficiency. To investigate the efficiency, after each time interval ends, we record the time spent in processing the tweets aggregated during that interval for the 5 different methods. The results are reported on the NYC dataset as it contains relatively more tweets. The total number of time intervals is 1,488.

[2]The instructions given to the judges are summarized at http://www.cs.umd.edu/~hyw/instructions-local-events.txt

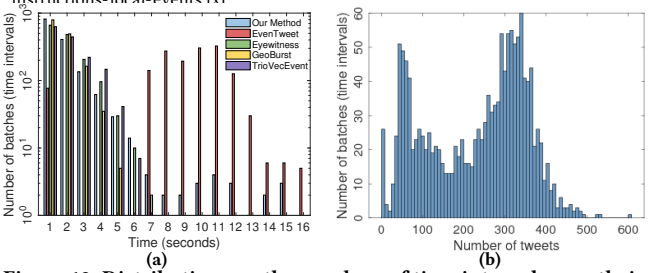


Figure 10: Distributions on the numbers of time intervals over their processing times in (a), and over their number of tweets in (b).

Figure 10a presents the distributions of time intervals over their processing time in different methods. To have an idea of the number of tweets in each time interval, we plot its histogram in Figure 10b. Among the three methods that exploit space partition strategy (i.e., EvenTweet, Eyewitness and DeLLe), Eyewitness in general is the most efficient method because it does not require sophisticated tweet text processing except when summarizing its detected event. DeLLe has achieved similar efficiency with Eyewitness in majority cases, even though a few of the cases sometimes take as long as 15 seconds. The major overhead lies in computing topical coherence in the ranker module as well as content similarity in the expander module. These steps, however, are only necessary when an unusual grid cell appears. Simply running the seeker module to identity potential local event candidates is very fast and takes 0.06 seconds on the average. EvenTweet is less efficient than the other methods due to its calculation of spatial entropy to identify spatially local keywords and then performing clustering. Although GeoBurst and TrioVecEvent have excellent efficiency as well, their implementations [24, 28] require certain preprocessing steps on the tweets like extracting keywords and keyword co-occurrence relation, which would take considerably more time.

6.4 Online Modifications

The batch mode of DeLLe divides the temporal dimension into disjoint time intervals, i.e., $\{\cdots[t-2\Delta t, t-\Delta t), [t-\Delta t, t)\}$. In practice, some local events may fall across these interval boundaries. We made online modifications in Section 5 for handling this issue. In this section, we investigate the effectiveness and efficiency of these modifications on the NYC dataset.

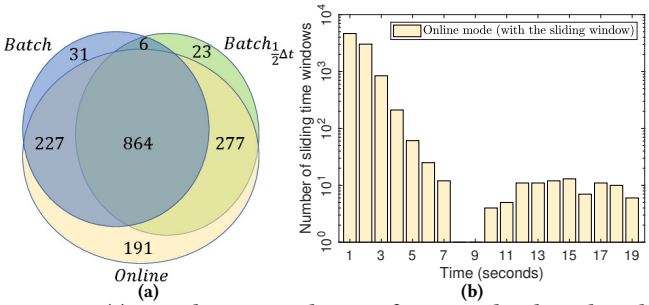


Figure 11: (a) Venn diagram on the sets of events in batch mode and online mode. (b) Distribution of time spent in online mode.

Effectiveness is evaluated by examining how many local events detected in the batch mode are also detected in the online processing and meanwhile how many local events the batch mode misses. For comparison, we here claim that two local event candidates refer to the same occurrence if i) their content similarity is greater than 0.7; ii) their time centroids (i.e., the average publish time of the tweets) are within $2\Delta t$ (i.e., one hour); iii) they come from the same grid cell. Figure 11a shows the Venn diagram of different sets of local event candidates generated in batch and online mode. For comparison, we also include one variant of the batch mode, *called* $Batch_{\frac{1}{2}\Delta t}$, which offsets the disjoin time intervals by $\frac{1}{2}\Delta t$, i.e, $\{\cdots[t-\frac{5}{2}\Delta t, t-\frac{3}{2}\Delta t), [t-\frac{3}{2}\Delta t, t-\frac{1}{2}\Delta t)\}$. The Venn diagram shows that the online mode, with help of the flexible sliding time window, has chances to screen different interval settings on the temporal dimension and indeed discovers more local events. We

also found that $Batch_{\frac{1}{2}\Delta t}$ has slightly more events than the original batch mode (i.e., dividing an integral hour into two time intervals). This is reasonable in the sense that although the latter time division fits more with the habits of people for planning events, people are likely to post tweets before an event starts when they have chances.

For evaluating the efficiency, we similarly record the processing time for each step in the sliding moving window. The results are presented in Figure 11b. Generally, the online processing shows a similar trend with the batch mode except with more cases falling after 10 seconds. After analyzing, we found that the major overhead lies in the frequent invocation of the expansion procedure to connect temporally adjacent cells that are semantically similar. Even so, the worst case takes less than 20 seconds in general and is likely acceptable for many applications.

7 CONCLUSIONS

In this paper, we presented DeLLe for detecting latest local events in geotagged tweet streams. In essence, DeLLe first identifies spatiotemporal unusualness using a novel prediction-based anomaly detection approach, and subsequently ranks them to identify potential local events, by addressing both spatiotemporal burstiness and topical coherence. Afterwards, DeLLe monitors the impact range for an ongoing local event in space and time by tracking its movement with content similarity, and meanwhile selects influential tweets for summarization. The evaluation results on two selected cities show that DeLLe outperforms competitive baselines in most cases, showing the effectiveness of the proposed method.

The human evaluation yields a groundtruth of local events, and therefore enables the exploration of learning to classify spatiotemporal unusualness into true/false local events using features like burstiness and topical coherence. We leave this for our future work.

8 ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under grant IIS-1816889.

REFERENCES

[1] N. Gramsky and H. Samet. Seeder Finder: Identifying Additional Needles in the Twitter Haystack. LBSN '13.
[2] J. Sankaranarayanan, H. Samet, B. E. Teitler, et al. TwitterStand: News in Tweets. SIGSPATIAL '09.
[3] H. Samet, J. Sankaranarayanan, M. D. Lieberman, et al. Reading News with Maps by Exploiting Spatial Synonyms. *Commun. ACM*, 2014.
[4] H. Samet, M. D. Adelfio, B. C. Fruin, et al. Porting a Web-based Mapping Application to a Smartphone App. SIGSPATIAL '11.
[5] H. Samet, B. E. Teitler, M. D. Adelfio, et al. Adapting a Map Query Interface for a Gesturing Touch Screen Interface. WWW '11.
[6] M. D. Lieberman, H. Samet, and J. Sankaranayananan. Geotagging: Using Proximity, Sibling, and Prominence Clues to Understand Comma Groups. GIR '10.
[7] M. D. Lieberman and H. Samet. Multifaceted Toponym Recognition for Streaming News. SIGIR '11.
[8] M. D. Lieberman and H. Samet. Adaptive Context Features for Toponym Resolution in Streaming News. SIGIR '12.
[9] H. Wei, J. Sankaranarayanan, and H. Samet. Finding and Tracking Local Twitter Users for News Detection. SIGSPATIAL '17.
[10] H. Wei, H. Zhou, J. Sankaranarayanan, et al. Residual Convolutional LSTM for Tweet Count Prediction. WWW '18 Companion.
[11] H. Wei, H. Zhou, J. Sankaranarayanan, et al. Detecting Latest Local Events from Geotagged Tweet Streams. SIGSPATIAL '18.
[12] H. Wei, J. Sankaranarayanan, and H. Samet. Enhancing Local Live Tweet Stream to Detect News. ACM SIGSPATIAL LENS '18.
[13] G. Quercini, H. Samet, J. Sankaranarayanan, et al. Determining the Spatial Reader Scopes of News Sources Using Local Lexicons. GIS '10.
[14] A. Jackoway, H. Samet, and J. Sankaranarayanan. Identification of Live News Events Using Twitter. LBSN '11.
[15] M. Mathioudakis and N. Koudas. TwitterMonitor: Trend Detection over the Twitter Stream. SIGMOD '10.
[16] H. Abdelhaq, C. Sengstock, and M. Gertz. EvenTweet: Online Localized Event Detection from Twitter. PVLDB '13.
[17] T. Lappas, M. R. Vieira, D. Gunopulos, et al. On the Spatiotemporal Burstiness of Terms. PVLDB '12.
[18] Q. He, K. Chang, and E.-P. Lim. Analyzing Feature Trajectories for Event Detection. SIGIR '07.
[19] H. Abdelhaq, M. Gertz, and C. Sengstock. Spatio-temporal Characteristics of Bursty Words in Twitter Streams. SIGSPATIAL'13.
[20] X. Shi, Z. Chen, H. Wang, et al. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. NIPS '15.
[21] J. Krumm and E. Horvitz. Eyewitness: Identifying Local Events via Space-time Signals in Twitter Feeds. SIGSPATIAL '15.
[22] F. Atefeh and W. Khreich. A Survey of Techniques for Event Detection in Twitter. *Comput. Intell.*, 31(1):132–164, February 2015.
[23] H. Abdelhaq. *Localized Events in Social Media Streams: Detection, Tracking, and Recommendation.* PhD thesis, Heidelberg University, November 2015.
[24] C. Zhang, L. Liu, D. Lei, et al. TrioVecEvent: Embedding-Based Online Local Event Detection in Geo-Tagged Tweet Streams. KDD '17.
[25] L. Hong, A. Ahmed, S. Gurumurthy, et al. Discovering Geographical Topics in the Twitter Stream. WWW '12.
[26] X. Zhou and L. Chen. Event Detection over Twitter Social Media Streams. *VLDB*, 23(3):381–400, June 2014.
[27] W. Wei, K. Joseph, W. Lo, et al. A Bayesian Graphical Model to Discover Latent Events from Twitter. ICWSM '15.
[28] C. Zhang, G. Zhou, Q. Yuan, et al. GeoBurst: Real-Time Local Event Detection in Geo-Tagged Tweet Streams. SIGIR '16.
[29] C. Zhang, D. Lei, Q. Yuan, et al. GeoBurst+: Effective and Real-Time Local Event Detection in Geo-Tagged Tweet Streams. *ACM TIST '18.*
[30] M. Walther and M. Kaisser. Geo-spatial Event Detection in the Twitter Stream. ECIR '13.
[31] A. Boettcher and D. Lee. EventRadar: A Real-Time Local Event Detection Scheme Using Twitter Stream. GreenCom '12.
[32] A. Magdy, M. F. Mokbel, S. Elnikety, et al. Mercury: A Memory-Constrained Spatio-temporal Real-time Search on Microblogs. ICDE '14.
[33] A. Magdy, A. M. Aly, M. F. Mokbel, et al. GeoTrend: Spatial Trending Queries on Real-time Microblogs. SIGSPATIAL '16.
[34] R. Lee and K. Sumiya. Measuring Geographical Regularities of Crowd Behaviors for Twitter-based Geo-social Event Detection. LBSN '10.
[35] Z. Liu, Y. Huang, and J. R. Trampier. LEDS: Local Event Discovery and Summarization from Tweets. SIGSPATIAL '16.
[36] A. Marcus, M. S. Bernstein, O. Badar, et al. Twitinfo: Aggregating and Visualizing Microblogs for Event Exploration. CHI '11.
[37] J. Weng and B.-S. Lee. Event Detection in Twitter. ICWSM '11.
[38] W. Kang, A. K. H. Tung, F. Zhao, et al. Interactive hierarchical tag clouds for summarizing spatiotemporal social contents. ICDE '14.
[39] A. Skovsgaard, D. Sidlauskas, and C. S. Jensen. Scalable top-k spatio-temporal term querying. ICDE '14.
[40] K. Y. Kamath, J. Caverlee, K. Lee, et al. Spatio-temporal Dynamics of Online Memes: A Study of Geo-tagged Tweets. WWW '13.
[41] K. Watanabe, M. Ochi, M. Okabe, et al. Jasmine: A Real-time Local-event Detection System Based on Geolocation Information Propagated to Microblogs. CIKM '11.
[42] C. Jonathan, A. Magdy, M. F. Mokbel, et al. GARNET: A holistic system approach for trending queries in microblogs. ICDE '16.
[43] J. Zhang, Y. Zheng, D. Qi, et al. DNN-based Prediction Model for Spatio-temporal Data. SIGSPATIAL '16.
[44] J. Zhang, Y. Zheng, and D. Qi. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. AAAI '17.
[45] A. Graves. Generating Sequences With Recurrent Neural Networks. *CoRR '14.*
[46] K. He, X. Zhang, S. Ren, et al. Deep Residual Learning for Image Recognition. *CVPR '16.*
[47] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Deep Image Prior. *CVPR '17.*
[48] B. Dhingra, Z. Zhou, D. Fitzpatrick, et al. Tweet2Vec: Character-Based Distributed Representations for Social Media. ACL '16.
[49] S. Vakulenko, L. Nixon, and M. Lupu. Character-based Neural Embeddings for Tweet Clustering. SocialNLP '17.
[50] L. Page, S. Brin, R. Motwani, et al. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.
[51] T. Landauer, P. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25:259–284, 1998.
[52] M. Yoon, W. Jin, and U. Kang. Fast and Accurate Random Walk with Restart on Dynamic Graphs with Guarantees. WWW '18.
[53] F. Chollet et al. Keras. https://github.com/fchollet/keras, 2015.