GeoBurst+: Effective and Real-Time Local Event Detection in Geo-Tagged Tweet Streams

CHAO ZHANG, University of Illinois at Urbana-Champaign
DONGMING LEI, University of Illinois at Urbana-Champaign
QUAN YUAN, University of Illinois at Urbana-Champaign
HONGLEI ZHUANG, University of Illinois at Urbana-Champaign
LANCE KAPLAN, U.S. Army Research Laboratory
SHAOWEN WANG, University of Illinois at Urbana-Champaign
JIAWEI HAN, University of Illinois at Urbana-Champaign

The real-time discovery of local events (*e.g.*, protests, disasters) has been widely recognized as a fundamental socioeconomic task. Recent studies have demonstrated that the geo-tagged tweet stream serves as an unprecedentedly valuable source for local event detection. Nevertheless, how to effectively extract local events from massive geo-tagged tweet streams in real time remains challenging. To bridge the gap, we propose a method for effective and real-time local event detection from geo-tagged tweet streams. Our method, named GeoBurst+, first leverages a novel cross-modal authority measure to identify several pivots in the query window. Such pivots reveal different geo-topical activities and naturally attract similar tweets to form candidate events. GeoBurst+ further summarizes the continuous stream and compares the candidates against the historical summaries to pinpoint truly interesting local events. Better still, as the query window shifts, GeoBurst+ is capable of updating the event list with little time cost, thus achieving continuous monitoring of the stream. We used crowdsourcing to evaluate GeoBurst+ on two million-scale data sets, and found it significantly more effective than existing methods while being orders of magnitude faster.

CCS Concepts: • **Information systems → Data management systems**; **Spatial-temporal systems**; **Data mining**; **Web mining**; **Information retrieval**;

Additional Key Words and Phrases: Event detection, local event, location-based service, data stream, social media, spatiotemporal data mining

ACM Reference format:
Chao Zhang, Dongming Lei, Quan Yuan, Honglei Zhuang, Lance Kaplan, Shaowen Wang, and Jiawei Han. 2017. GeoBurst+: Effective and Real-Time Local Event Detection in Geo-Tagged Tweet Streams. *ACM Trans. Intell. Syst. Technol.* 1, 1, Article 1 (March 2017), 23 pages.
DOI: 0000001.0000001

1 INTRODUCTION

1.1 Motivation

A local event (*e.g.*, protest, crime, disaster, sport game) is an unusual activity bursted in a local area and within specific duration while engaging a considerable number of participants. The real-time discovery of local events has been recognized as an important task for a wide spectrum

DOI: 0000001.0000001

of applications. Consider disaster control as an example. By detecting emergent disasters (*e.g.*, earthquakes, fires) in real time, we can send alarms to the populace at the very first moment when these disasters outbreak. Such real-time alarms are expected to be much faster than traditional reports [8], [29], [45] and thus allow for timely response to avoid huge life and economic losses. As another example, with an intelligent detector that continuously extracts interesting local events, it is feasible to achieve effective personalized activity recommendation in the urban space. Suppose a user is interested in sport games and music festivals, the detector can easily identify related events with a few filtering keywords, and continuously feed the user with events-of-interest.

While the real-time detection of local events was nearly impossible years ago due to the lack of reliable data sources, the explosive growth of geo-tagged tweet data brings new opportunities to it. With the ubiquitous connectivity of wireless networks and the wide proliferation of mobile devices, more than 10 million geo-tagged tweets are created in the Twitterverse every day. Each geo-tagged tweet, which contains a text message, a timestamp, and a geo-location, provides a unified 3W (*what-when-where*) view of the user's activity. For example, when the tragic 2011 Tohoku Earthquake hit Japan on March 11th 2011, thousands of related geo-tagged tweets were created instantly; and when the Baltimore Riot took place in April 2015, many people posted geo-tagged tweets to broadcast it right on the spot. Its sheer size, multi-faceted information, and real-time nature make the geo-tagged tweet stream an unprecedentedly valuable source for detecting local events.

1.2 Challenges

Our goal is to achieve *real-time* and *effective* local event detection from geo-tagged tweet streams. The challenge of this problem is three-fold:

- *Integrating diverse types of data.* The geo-tagged tweet stream involves three different data types: location, time, and text. Considering the totally different representations of those data types and the complicated cross-modal interactions among them, how to effectively integrate them for local event detection is challenging.
- *Capturing the semantics of short text.* Since every tweet is limited to 140 characters, the semantics of the user's activity is expressed through short and sparse text messages. Compared with traditional documents (*e.g.*, news), it is much more difficult to capture the semantics of short tweet messages and extract high-quality local events.
- *On-line and real-time detection.* When a local event outbreaks, it is key to report the event instantly to allow for timely actions. As massive geo-tagged tweets stream in, the detector should work in an on-line and real-time manner instead of a batch-wise and inefficient one. Such a requirement is the third challenge of our problem.

Recently, there has been increasing interest in leveraging social media for modeling people's spatiotemporal activities in the physical world, addressing tasks like event detection [1, 3, 6, 13, 14, 19, 21, 33], geographical topic discovery [9, 17, 18, 31, 39], and mobility modeling [36, 37, 40]. Among them, [3, 14, 21, 33] are very related to our problem as they also aim to extract interesting events on Twitter, but they are all designed to detect global events instead of local events. Unlike global events that are bursty in the entire stream, local events are "bursty" in a small geographical region and involve much fewer tweets. Such local bursts cannot be readily captured by global event detection methods. A few methods tailored for local event detection [1, 6, 13, 19] have been introduced. Nevertheless, most of them process the geo-tagged tweet data in a batch manner, and none of them can support *real-time* local event detection from geo-tagged tweet streams.

1.3 Contributions

We propose an effective and real-time local event detector called GeoBurst+. Our insight behind the design of GeoBurst+ is that, as a local event outbreaks, there are usually a considerable number of geo-tagged tweets around the occurring place (*e.g.*, many participants of a protest may post tweets on the spot). As such tweets are geographically close and semantically coherent, they form a geo-topical cluster and serve as a potential local event. However, not necessarily does every geo-topical cluster correspond to a local event. First, *the cluster may not be spatiotemporally unusual.* A geo-topical cluster could be just a routine regional activity (*e.g.*, many shopping-related tweets are posted on the Fifth Avenue in New York every day), or geographically scattered discussions (*e.g.*, a popular TV show may result in several geo-topic clusters in different regions). Second, *the cluster may not be spatiotemporally bursty.* A cluster that contains a limited number of tweets may be just random babbles from users instead of interesting local events. Therefore, we claim that a geo-topical cluster should be spatiotemporally unusual and bursty to form a local event, and it is necessary to carefully judge each candidate to pinpoint true local events.

Motivated by the above, GeoBurst+ first finds all geo-topical clusters in the query window based on a novel authority measure. The measure quantifies a tweet's geo-topical authority by combining the geographical and semantic contributions from its similar tweets, where the geographical side is captured with a kernel function, and the semantic side is captured with random walk on a keyword co-occurrence graph. With the authority measure, we design an authority ascent procedure to identify all pivot tweets, which are essentially authority maxima in the geo-topical space. Such pivots reflect different representative activities in the query window and naturally attract similar tweets to form geo-topical clusters as candidate events.

To judge whether each candidate is indeed an interesting local event, GeoBurst+ consists of a summarization module that summarizes the continuous geo-tagged tweet stream. The obtained summaries not only encode the typical activities in different geographical regions, but also captures the subtle semantics of different keywords and tweets by embedding them into a latent space. Relying upon the summaries, we compare each candidate event against the routine activities to extract a set of discriminative features, which allow us to train a classifier to accurately determine whether each candidate is a true local event.

Better still, as the query window shifts, GeoBurst+ does not need to extract new local events from scratch. Instead, it features an updating module that updates the event list continuously as new geo-tagged tweets stream in. The updating incurs little time cost because authority computation, which is the most time-consuming operation in GeoBurst+, can be completed by subtracting the contributions of the outdated tweets and emphasizing the contributions of the new ones. Such an updating module enables effective monitoring of the tweet stream to report local events in a real-time and continuous manner.

The major contributions of this work are summarized as follows:

(1) We design GeoBurst+ for local event detection in geo-tagged tweet streams. The effectiveness of GeoBurst+ is underpinned by a novel cross-modal authority measure that generates candidate events, along with a module that summarizes the continuous tweet stream to accurately pinpoint true local events.
(2) With the additive property of the authority score, we design an updating module for GeoBurst+. It fast updates the event list when the query window shifts, and thus enables real-time and continuous local event detection. To the best of our knowledge, GeoBurst+ is the first method that can achieve real-time local event detection from geo-tagged tweet streams.

(3) We performed extensive experiments on millions of geo-tagged tweets in two different cities, and evaluated the results using a crowdsourcing platform. Our results demonstrate that GeoBurst+ significantly outperforms state-of-the-art methods in effectiveness, and runs orders of magnitude faster.

A preliminary version of GeoBurst+ has been presented in [41]. Compared with the preliminary version, our GeoBurst+ method employs a new supervised framework for selecting the true local events, while the previous GeoBurst method ranks all the candidates and selects the top-K bursty ones. In addition, GeoBurst+ performs keyword embedding to capture the subtle semantics of tweet messages, which is also a new component. The major advantage of the GeoBurst+ method over its preliminary version is two-fold: (1) It frees us from manually designing ranking functions and removes the inflexibility of rigid top-K selection for every query window; and (2) It can easily incorporate other signals (*e.g.*, embedding-based features) that can help characterize true local events to achiever better effectiveness. Our experiments verify that both the supervised framework and the keyword embedding technique are useful in improving the detection effectiveness considerably.

2 PRELIMINARIES

In this section, we formulate the real-time local event detection problem, and then explore several of its characteristics, which motivate the design of GeoBurst+.

2.1 Problem Description

Let $\mathcal{D} = (d_1, d_2, \ldots, d_n, \ldots)$ be a continuous stream of geo-tagged tweets that arrive in chronological order. Each tweet d is a tuple $\langle t_d, l_d, E_d \rangle$, where t_d is its post time, l_d is its geo-location, and E_d is a bag of keywords. For each tweet, we use an off-the-shelf tool [12] to extract verbs and nouns as its keywords. Note that such preprocessing does not affect the generality of our method, and one can also represent each tweet message as a bag of uni-grams for simplicity.

Consider a query time window $Q = [t_s, t_e]$ where t_s and t_e are the start and end timestamps satisfying $t_{d_1} \leq t_s < t_e \leq t_{d_n}$. The local event detection problem consists of two sub-tasks: (1) extract from $\mathcal{D}$ all the local events that occur during Q; and (2) monitor the continuous stream $\mathcal{D}$ and update the local event list in real time as Q shifts continuously.

2.2 GeoBurst+ Overview

We provide the following insights about the key factors that characterize a local event:

- *A local event often results in a group of relevant tweets around its occurring location.* Take Figure 1 as an example. Suppose a protest occurs on the 5th Avenue in New York, many participants may post tweets on the spot to express their attitude, with keywords such as "protest" and "rights". We call such a set of tweets a *geo-topical cluster* as they are geographically close and semantically coherent.
- *A local event is spatiotemporally unusual.* Not necessarily does every geo-topical cluster correspond to a local event. Continue with the example in Figure 1. During almost any hour, we can observe many shopping-related tweets on the 5th Avenue. Although such tweets also form a geo-topical cluster, they do not reflect any unusual activities. Meanwhile, the cluster may correspond to a global event instead of a local one. For instance, when a popular TV show like "Game of Thrones" goes online, we can observe geo-topical clusters discussing about it in different regions. Such geo-topical clusters do not correspond to local events as well.

- *A local event is spatiotemporally bursty.* Even if the cluster is spatiotemporal unusual, it may not be an interesting event if it has a small size. Previous research has shown that about 40% tweets are just user babbles. As such, the geo-topical clusters that are not spatiotemporally bursty may be just uninteresting babbles from a few users instead of meaningful local events.

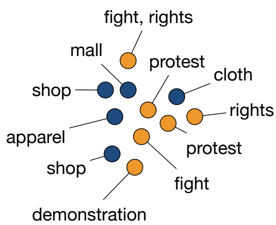


Fig. 1. Example geo-topical clusters.

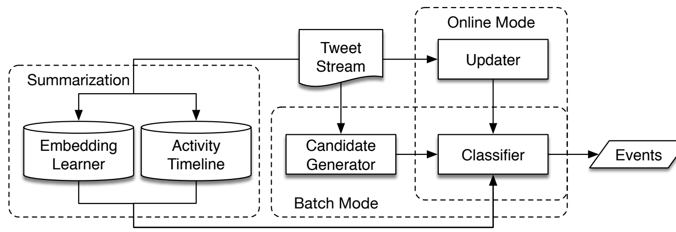


Fig. 2. The framework of GeoBurst+.

We claim that *a local event is a geo-topical cluster that is spatiotemporally* ***unusual*** *and shows clear spatiotemporal* ***burstiness***. Based on the above insights, we design the framework of GeoBurst+ in Figure 2. As shown, there are three key modules: 1) the candidate generator; 2) the summarization module; and 3) the on-line updater. First, the *candidate generator* detects all geo-topical clusters in the query window and regards them as candidates — this step ensures high coverage of the underlying local events. The discovery of geo-topical clusters relies on a novel authority measure that captures the cross-modal correlations among the geo-tagged tweets, as well as a novel non-parametric procedure for detecting all the authority maxima. Second, the *summarization module* performs continuous summarization of the stream and extracts background knowledge to classify the candidate events. It consists of: 1) an activity timeline that stores the typical activities in different regions; and 2) an embedding learner that derives low-dimensional embeddings for any ad-hoc tweets. The activity timeline allows for quantifying the spatiotemporal burstiness of each candidate event, while the embedding learner capture the intrinsic semantics of the short tweets to measure unusualness. Those two components collectively enable us to extract a set of discriminative features for each candidate event and thus select out true local events. Third, the *online updater* can update the result list in real time as the query window shifts. It will be shown shortly that, the authority score satisfies an additive property. Hence, instead of finding new candidates from scratch when the query window shifts, we can identify them by simply updating the authority scores and then performing fast authority ascent.

3 THE CANDIDATE GENERATOR

In this section, we describe the candidate generator of GeoBurst+. Given a query window Q and the set D_Q of tweets falling in Q, the candidate generator is to divide D_Q into several geo-topical clusters, such that the tweets in each cluster are geographically close and semantically coherent. The clustering of D_Q, however, poses several challenges: how to combine the geographical and semantic similarities in a reasonable way? how to capture the correlations between different keywords? and how to generate quality clusters without knowing the suitable number of clusters in advance?

To address these challenges, we perform a novel pivot seeking process to identify the centers of geo-topical clusters. Our key insight is that: the spot where the event occurs acts as a pivot that produces relevant tweets around it; the closer we are to the pivot, the more likely we observe relevant tweets. Therefore, we define a geo-topical authority score for each tweet, where the geographical influence among tweets is captured by a kernel function, and the semantic influence by random walk on a keyword co-occurrence graph. With this authority measure, we develop an authority ascent procedure to retrieve authority maxima as pivots; and each pivot naturally attracts similar tweets to form a quality geo-topical cluster. Below, we first introduce our geo-topical authority measure to define pivot tweets, and then develop an authority ascent procedure for pivot seeking.

3.1 Pivot Tweet

3.1.1 Geographical Proximity. Given two tweets d and d', we measure the geographical proximity of d' to d as $G(d' \rightarrow d) = K(\|l_d - l_{d'}\|/h)$, where $K(\cdot)$ is a kernel function, $\|l_d - l_{d'}\|$ is the geographical distance between d and d', and h is the kernel bandwidth. While various kernel functions can be used, we choose the Epanechnikov kernel here due to its simplicity and optimality in terms of bias-variance tradeoff [7]. With the Epanechnikov kernel, $G(d' \rightarrow d)$ becomes

$$G(d' \rightarrow d) = \begin{cases} c(1 - \|l_d - l_{d'}\|^2/h^2) & \text{if } \|l_d - l_{d'}\| < h \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where c is a scaling constant of the Epanechnikov kernel.

3.1.2 Semantic Proximity. As each tweet message is represented by a bag of keywords, a very straightforward idea for measuring semantic proximity is to compute the vector similarity between two tweet messages. Nevertheless, the effectiveness of vector similarity is limited, not only because tweets are short in nature, but also that the dimensions (keywords) are correlated instead of independent. To overcome these drawbacks, we propose a random-walk-based approach to capture semantic proximity more effectively.

Definition 3.1 (Keyword Co-occurrence Graph). The keyword co-occurrence graph for D_Q is an undirected graph $G = (V, E)$ where: (1) V is the set of all keywords in D_Q; and (2) E is the set of edges between keywords, and the weight of an edge (e_i, e_j) is the number of tweets in which e_i and e_j co-occur.

The keyword co-occurrence graph can be easily built from D_Q. With such a graph, we employ *random walk with restart (RWR)* to define keyword similarity as it uses the holistic graph structure to capture node correlations. Consider a surfer who starts RWR from the keyword $x_0 = u$. Suppose the surfer is at keyword $x_t = i$ at step t, she returns to u with probability α $(0 < \alpha < 1)$ and continues surfing with probability $1 - \alpha$. If continuing, she randomly moves to i's neighbor j with probability $\mathbf{P}_{ij}$, where $\mathbf{P}$ is the transition matrix of the graph. The stationary distribution of such a process defines the RWR scores from u to all the keywords in V, and the score from u to keyword v, denoted as $r_{u \rightarrow v}$, is the probability that the surfer resides on v. Given two tweets d and d', we start RWR from the keywords of d', and define the semantic proximity of d' to d as the average probability that the random walk resides on d. Formally, let $E_d = \{e_1, e_2, \ldots, e_m\}$ be the keyword set of d, and $E_{d'} = \{e'_1, e'_2, \ldots, e'_n\}$ the keyword set of d', then the semantic proximity from d' to d is

$$S(d' \rightarrow d) = \frac{1}{mn} \sum_{e \in E_d} \sum_{e' \in E_{d'}} r_{e' \rightarrow e}. \tag{2}$$

3.1.3 geo-topical authority. Based on the geographical and semantic proximities, we measure the *geo-topical authority* of a tweet as follows.

Definition 3.2 (Neighbor). Given a tweet d, we say d' is a neighbor of d if d' satisfies $G(d' \rightarrow d) > 0$ and $S(d' \rightarrow d) > \delta$, where $0 < \delta < 1$ is a pre-specified threshold.

Definition 3.3 (Authority). Given a tweet $d \in D_Q$, let $N(d)$ be the set of d's neighbors in D_Q. The authority of d is $A(d) = \sum_{d' \in N(d)} G(d' \rightarrow d) \cdot S(d' \rightarrow d)$.

Given a tweet d, d' is a neighbor of d if it resembles d both geographically and semantically. The set of all neighbors in D_Q form d's neighborhood and contribute to d's authority. We could interpret Definition 3.3 as follows: an amount of $G(d' \to d)$ energy is distributed from d' to d through random walk on the graph, $G(d' \to d) \cdot S(d' \to d)$ is the amount that successfully reaches d; and d's authority is the total amount of energy that d receives from its neighbors [38]. The authority score is analogous to kernel density in the task of non-parametric kernel density estimation [7]. In kernel density estimation, the density of any point x in the Euclidean space is contributed mainly by the observed points that are close enough to x. As such, the density maxima can be defined in a non-parametric manner. Analogously, in our problem, the geo-topic authority of any tweet d is contributed by the observed tweets that are similar to d both geographically and semantically. As a result, the salient tweets for different activities can be selected in the geo-topical space.

3.1.4 Pivot. With Definition 3.3, we define a *pivot* as an authority maximum.

Definition 3.4 (Pivot). Given a tweet $d \in D_Q$ and its neighborhood $N(d)$, d is a pivot if $A(d) = \max_{d' \in N(d)} A(d')$.

Consider a local event that occurs at location l. If d is a tweet discussing about that event at l, then d is likely to be surrounded by relevant tweets to become the pivot for that event. The notion of neighborhood plays an important role in Definition 3.4: it ensures the supporting tweets are both geographically close and semantically relevant. This property leads to different pivots that can distinguish different-semantics events happening at the same location, as well as same-semantics events happening at different locations.

3.2 Authority Ascent for Detecting Geo-Topical Clusters

Now our task is to find all pivots in D_Q and assign each tweet to its corresponding pivot. We develop an authority ascent procedure for this purpose. As shown in Figure 3, starting from a tweet d_1 as the initial center, we perform step-by-step center shifting. Assuming the center at step t is tweet d_t, we find d_t's neighborhood $N(d_t)$, and the *local pivot* $l(d_t)$ — the tweet having the largest authority in $N(d_t)$. Then we regard $l(d_t)$ as our new center, *i.e.*, $d_{t+1} = l(d_t)$. As we continue such an authority ascent process, the center is guaranteed to converge to an authority maximum. It is because every shift operation increases the authority of the current center, and the authority is upper bounded (there are only a finite number of tweets in D_Q).

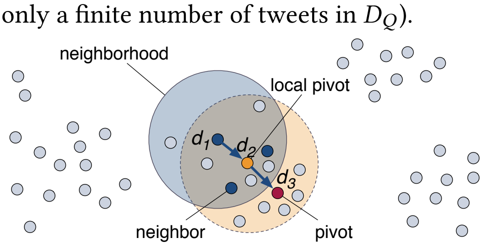


Fig. 3. An illustration of the authority ascent process.

Algorithm 1 depicts the process of finding the pivot for every tweet in D_Q. As shown, we first compute the neighborhood for each tweet $d \in D_Q$ (lines 1-2). Subsequently, we compute the authority of each tweet (lines 3-4), and obtain its local pivot (lines 5-6). So long as the local pivots are obtained, we perform authority ascent to identify the pivot each tweet belongs to. Finally, the tweets having the same pivot are grouped into one geo-topical cluster and returned as a candidate event.

The geographical kernel bandwidth, the geographical threshold, and the semantic threshold play an important role in constraining the neighbor set and guaranteeing the coherence of the

ALGORITHM 1: Pivot seeking.

Input: The tweet set D_Q, the kernel bandwidth h, the semantic threshold δ.
Output: The pivot for each tweet in D_Q.

`// Neighborhood computation.`
foreach $d \in D_Q$ **do**
  $N(d) \leftarrow \{d' | d' \in D_Q, G(d' \to d) > 0, S(d' \to d) > \delta\}$;

`// Authority computation.`
foreach $d \in D_Q$ **do**
  $A(d) \leftarrow d$'s authority score computed from $N(d)$;

`// Find local pivot for each tweet.`
foreach $d \in D_Q$ **do**
  $l(d) \leftarrow \arg\max_{d' \in N(d)} A(d')$;

`// Authority ascent.`
foreach $d \in D_Q$ **do**
  Perform authority ascent to find the pivot for d;

final geo-topical clusters. Specifically: (1) with the Equation 1 and the geographical threshold set to 0, only the tweets that are close enough to d can fall in d's neighborhood, thus ensuring the geographical compactness of the result clusters; (2) with the semantic threshold δ, only the tweets that are semantically similar enough can fall in d's neighborhood, thus ensuring the semantic coherence of the result clusters.

In Algorithm 1, while it is easy to compute geographical proximity based on tweet location, the challenge is how to compute semantic proximity efficiently. A naïve idea is to obtain the RWR score between any two keywords, but such an idea is not efficient as the keyword co-occurrence graph can be large. To address this challenge, we leverage the locality of RWR: given a keyword q, we observe that only a limited number of keywords falling in q's vicinity have large values, while most keywords have extremely small RWR scores. We thus introduce the concept of *keyword vicinity*, which keeps only large enough RWR scores by exploring a small neighborhood around q. Below, we demonstrate how to fast compute the keyword vicinity based on the *Decomposition Theorem* [16].

Theorem 3.5. *For a keyword u, let O_u be the set of u's out-neighbors in G. Given a keyword q, the RWR from u to q satisfies*

$$r_{u\to q} = \begin{cases} (1-\alpha)\sum_{v\in O_u} \mathbf{P}_{uv} r_{v\to q} & \text{if } u \neq q \\ (1-\alpha)\sum_{v\in O_u} \mathbf{P}_{uv} r_{v\to q} + \alpha & \text{if } u = q. \end{cases} \tag{3}$$

Theorem 3.5 says that, the RWR from u to q can be derived by linearly combining the RWR scores of u's out-neighbors, with extra emphasis on q itself. With this theorem, we use a local computation algorithm [23] to obtain q's vicinity. Starting from an initial vicinity, we gradually expand the vicinity and propagate RWR scores among the keywords falling inside. The RWR approximation becomes tighter and tigher as the vicinity expansion continues, and terminates when an error bound ϵ ($0 < \epsilon \ll \delta$) is guaranteed. Algorithm 2 depicts the detailed vicinity computation process. To compute q's vicinity, we maintain two quantities for any keyword u: (1) $s(u)$ is the current RWR score from u to q; and (2) $p(u)$ is the score that needs to be propagated. We use a priority queue to keep $p(u)$ for all the keywords. Every time we pop the keyword u that has the largest to-propagate score, and update the score and to-propagate score for each in-neighbor of u. After that, we set

ALGORITHM 2: Approximate RWR score computation.

Input: The keyword co-occurrence graph G, a keyword q, the restart probability α, an error bound ϵ.
Output: q's vicinity V_q.

// $p(u)$ is the score of node u that needs to be propagated.
$s(q) \leftarrow \alpha, p(q) \leftarrow \alpha, V_q \leftarrow \phi$;
$Q \leftarrow$ a priority queue that keeps $p(u)$ for the keywords in G;
while *Q.peek()* $\geq \alpha\epsilon$ **do**
    $u \leftarrow$ Q.pop();
    for $v \in I(u)$ **do**
        $\Delta s(v) = (1-\alpha) p_{vu} p(u)$;
        $s(v) \leftarrow s(v) + \Delta s(v)$;
        $V_q[v] \leftarrow s(v)$;
        Q.update($v, p(v) + \Delta s(v)$);
    $p(u) \leftarrow 0$;
return V_q;

$p(u)$ to zero to avoid redundant propagation. The algorithm terminates when the max element in the priority queue is less than $\alpha\epsilon$, and returns all the keywords that have non-zero RWR scores as q's vicinity. Any keyword u not in q's vicinity must satisfy $r_{u\rightarrow q} < \epsilon$.

Theorem 3.6. *Let $\hat{r}_{u\rightarrow q}$ be the approximate RWR score computed by Algorithm 2, then $\hat{r}_{u\rightarrow q}$ satisfies $|r_{u\rightarrow q} - \hat{r}_{u\rightarrow q}| \leq \epsilon$. The time complexity of Algorithm 2 is $O(D_q/\alpha \log 1/(\epsilon\alpha))$, where $D_q = \sum_{u: s_{u\rightarrow q} > \alpha\epsilon} (|I(u)| + \log |V|)$.*

Proof. See [23] for details. □

With Theorem 3.6, we further analyze the complexity of Algorithm 1 as follows. First, for each keyword, we need to compute its vicinity using Algorithm 2. Assume the total number of keywords is M, then the complexity of this part is $O(MD_q/\alpha \log 1/(\epsilon\alpha))$, where $D_q = \sum_{u: s_{u\rightarrow q} > \alpha\epsilon} (|I(u)| + \log |V|)$. Second, based on the obtained keyword vicinities, we need to perform the pivot seeking process for every tweet in the query window. Assume the maximum number of tweets in the query window is N, then the time complexity of the pivot seeking process is $O(N^2)$. Therefore, the overall complexity of the candidate generation step is $O(MD_q/\alpha \log 1/(\epsilon\alpha) + N^2)$.

4 CANDIDATE EVENT CLASSIFICATION

Up to now, we have obtained a set of geo-topical clusters in the query window as candidate events. Nevertheless, as aforementioned, not necessarily does every candidate correspond to a local event. In this section, we describe the module for candidate event classification. The foundation of our classification is the summarization module, which learns word embedding to capture the semantics of short tweet messages and meanwhile constructs the activity timeline to reveal routine regional activities. In what follows, we describe embedding learning and activity timeline construction in Section 4.1 and 4.2, respectively; and then present the classifier in Section 4.3.

4.1 Learning Embeddings from the Stream

The embedding learner aims at capturing the semantics of short text by jointly mapping the tweet messages and keywords into the same low-dimensional space. If two tweets (keywords) are semantically similar, they are forced to have close embedding vectors in the latent space. The learner continuously consumes a massive amount of tweets from the input stream and learns to preserve their intrinsic semantics. As such, it can generate fixed-length vectors for any text pieces (*e.g.*, the candidate event and the background activity), which serve as high-quality features to discriminate whether a candidate event is indeed a local event or not.

The objective of the embedding learner is to reconstruct the observed tweets as much as possible. Specifically, given a tweet d and a set of keywords $w_1, w_2, \ldots, w_n$ that appear in d, we model the probability of observing the keyword $w_i (1 \leq i \leq n)$ as $p(w_i|d_{-i}) = \exp(s(w_i, d_{-i}))/\sum_{w_j \in V} \exp(s(w_j, d_{-i}))$, where d_{-i} is the set of all the units in d except w_i, $s(w_i, d_{-i})$ is the similarity score between w_i and d_{-i} based on their embeddings, and V is the keyword vocabulary. The key is how to define $s(w_i, d_{-i})$. Inspired by the success of the Paragraph Vector model [20] for capturing the semantics of sentences and documents, we assume both the keywords and the tweet itself have latent representations in the common space. Such a joint embedding strategy can lead to more discriminative representations for the tweet compared to learning keywords' embeddings alone and computing the average as the tweet embedding. Hence, we define $s(w_i, d_{-i})$ as

$$s(w_i, d_{-i}) = \mathbf{v}_i^{\mathrm{T}} \frac{\mathbf{v}_d + \sum_{w_j \in d_{-i}} \mathbf{v}_j}{|d_{-i}| + 1},$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are the latent embeddings for word w_i and w_j, respectively; and $\mathbf{v}_d$ is the latent embedding for the tweet d.

Ideally, the embeddings of the tweets and keywords should be learnt to maximize the likelihood of observing all the tweets seen so far. Nevertheless, as the embedding learner runs in a stream setting, it is infeasible to store all the seen tweets and iterate through them for multiple epochs — as done in previous works. To tackle this issue, we maintain a fixed-size cache for storing the incoming tweets. Once the cache is saturated, we randomly shuffle the stored tweets and use them to update the embeddings of the keywords, and then empty the cache to accommodate future tweets from the stream. More specifically, let C be the collection of tweets in the current cache, we define the objective function as observing all the units in C, namely

$$O = -\sum_{d \in C} \sum_{w_i \in d} \log p(w_i|d_{-i}). \tag{4}$$

To efficiently optimize the above objective, we use stochastic gradient descent (SGD) and negative sampling [25]. At each time, we use SGD to sample a tweet d and a word $w_i \in d$. With negative sampling, we randomly select K negative words that do not appear in d, then the loss function for the selected samples becomes:

$$L = -\log \sigma(s(w_i, d_{-i})) - \sum_{k=1}^{K} \log \sigma(-s(w_k, d_{-i})),$$

where $\sigma(\cdot)$ is the sigmoid function. Letting $\mathbf{h}_i = (\mathbf{v}_d + \sum_{w_j \in d_{-i}} \mathbf{v}_j)/(|d_{-i}| + 1)$, then the updating rules for $\mathbf{v}_i$, $\mathbf{v}_k$, and $\mathbf{h}_i$ can be obtained based on the following derivatives:

$$\frac{\partial L}{\partial \mathbf{v}_i} = -\sigma(-s(w_i, d_{-i}))\mathbf{h}_i; \ \frac{\partial L}{\partial \mathbf{v}_k} = \sigma(s(w_i, d_{-i}))\mathbf{h}_i;$$

$$\frac{\partial L}{\partial \mathbf{h}_i} = \sum_{k=1}^{K} \sigma(s(w_k, d_{-i}))\mathbf{v}_k - \sigma(-s(w_i, d_{-i}))\mathbf{v}_i.$$

For any unit j in $\mathbf{h}_i$ (can be the tweet d or any keyword $w \in d_{-i}$), we have $\partial L/\partial \mathbf{v}_j = \partial L/\partial \mathbf{h}_i \cdot \partial \mathbf{h}_i/\partial \mathbf{v}_j$, as $\mathbf{h}_i$ is linear in j, the item $\partial \mathbf{h}_i/\partial \mathbf{v}_j$ is straightforward to obtain.

Relying on the tweet caching strategy and the SGD optimization procedure, the embedding learner continuously consumes the geo-tagged tweet stream and keeps updating the embeddings for different keywords and tweets. With the learnt keyword embeddings, the embedding of any ad-hoc text piece can be easily derived with SGD. As we will illustrate shortly, such a property enables us to quantify the spatiotemporal unusualness of each candidate event and extract highly discriminative features to pinpoint true local events.

4.2 Activity Timeline Construction

The activity timeline aims at unveiling the typical activities in different regions during different time periods. For this purpose, we design a structure called *tweet cluster* (TC) and extend the CluStream algorithm [2]. Let S be a set of tweets that are geographically close, its TC maintains the following statistics:

1) $n = |S|$: the number of tweets.
2) $m_l = \sum_{d \in S} l_d$: the sum vector of locations.
3) $m_{l^2} = \sum_{d \in S} l_d \circ l_d$: the squared sum vector of locations.
4) $m_t = \sum_{d \in S} t_d$: the sum of timestamps.
5) $m_{t^2} = \sum_{d \in S} t_d^2$: the squared sum of timestamps.
6) $m_e = \sum_{d \in S} E_d$: the sum dictionary of keywords.

The TC essentially provides a concise where-when-what summary for S: (1) *where*: with n, m_l, and m_{l^2}, one can easily compute the location mean and variance for S; (2) *when*: with n, m_t, and m_{t^2}, one can compute the mean time and temporal variance for S; and (3) *what*: m_e keeps the number of occurrences for each keyword.

These fields in a TC S enable us to estimate the number of keyword occurrences at any location. First, the quantities n, m_l, and m_{l^2} allow us to compute the center location of the TC S. Second, the m_e tracks the number of occurrences for different keywords around the centered location of S. With either spatial interpolation or kernel density estimation, one can estimate the occurrences of keyword k at any ad-hoc location based on the distance to the center location of S.

Moreover, TC satisfies the additive property, *i.e.*, the fields can be easily incremented if a new tweet is absorbed. Based on this property, we adapt CluStream to continuously clusters the stream into a set of TCs. When a new tweet d arrives, it finds the TC m that is geographically closest to d. If d is within m's boundary (computed from n, m_l, and m_{l^2}, see [2] for details), it absorbs d into m and updates its fields; otherwise it creates a new TC for d. Meanwhile, we employ two strategies to limit the maximum number of TCs: (1) deleting the TCs that are too old and contain few tweets; and (2) merging closest TC pairs until the number of remaining TCs is small enough. We cluster the continuous stream and store the clustering snapshots at different timestamps. Since storing the snapshot of every timestamp is unrealistic, we use the pyramid time frame (PTF) structure [2] to achieve both good space efficiency and high coverage of the stream history.

4.3 The Classifier

The learnt embeddings and the activity timeline serve as useful background knowledge for classifying candidate events. Based on them, we extract the following set of discriminative features to characterize each candidate event:

Temporal unusualness. The temporal unusualness measures how unusual a candidate C is at its pivot location l_C. To quantify C's temporal unusualness, our idea is to leverage the embedding learner to obtain low-dimensional vectors for both the candidate C as well as the the background activity at l_C to compare them.

We compute the temporal unusualness measure as follows.

(1) For the candidate C, we form a pseudo tweet of C by selecting the top K keywords based on TF-IDF weights. Once the pseudo tweet is obtained, we process it with the learnt keyword embeddings to derive its textual embedding, denoted as $\mathbf{v}_C$.
(2) To obtain the embedding for the background activity, we examine the most recent snapshot from the activity timeline and retrieve the closest cluster with l_C. Such a cluster, denoted as T_l, encodes the typical activities around location l_C. Based on the statistics stored in T_l, we again form a pseudo tweet for T_l by selecting the top K keywords, and then use keyword embeddings to obtain the embedding of T_l, denoted as $\mathbf{v}_T$.
(3) After computing the two vectors $\mathbf{v}_C$ and $\mathbf{v}_T$, we compute temporal unusualness as the cosine distance between them, namely

$$f_T(C) = \cos(\mathbf{v}_C, \mathbf{v}_T).$$

Spatial unusualness. The spatial unusualness captures how spatially unique the candidate C is compared to other candidate events in the query window. We quantify the spatial unusualness as follows:

We compute the spatial unusualness measure as follows.

(1) For the candidate C, we still form a pseudo tweet of C by selecting the top K keywords based on TF-IDF weights, and derive its embedding $\mathbf{v}_C$.
(2) Given the tweet corpora D_Q in the query window, we select the top K keywords from D_Q based on TF-IDF weights, and derive its embedding $\mathbf{v}_Q$.
(3) We compute spatial unusualness as the cosine distance between the two vectors

$$f_T(C) = \cos(\mathbf{v}_C, \mathbf{v}_Q).$$

Temporal burstiness. To measure how temporally bursty a candidate event C is, we quantify the temporal burstiness of each keyword in C, and then aggregate the burstiness of all the keywords. As shown in Figure 4, we retrieve the snapshots in a reference time window R that right precedes the query window Q. Each pair of consecutive snapshots in R corresponds to a *historical activity*, defined as follows.

Definition 4.1 (Historical activity). Let s_1 and s_2 be two snapshots at timestamp t_{s_1} and t_{s_2} ($t_{s_1} < t_{s_2}$). The historical activity during the time interval $[t_{s_1}, t_{s_2}]$ is the set of TCs obtained by subtracting s_1 from s_2.

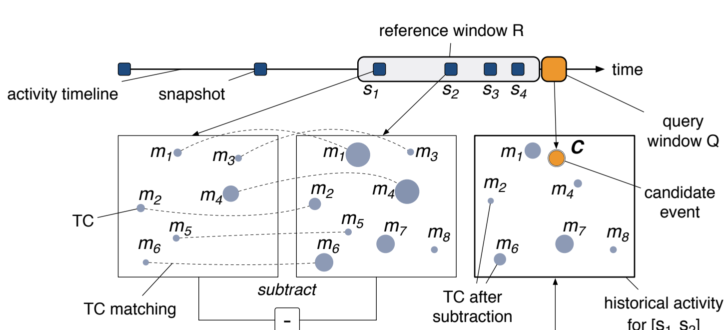


Fig. 4. Retrieving historical activities from activity timeline.

Let us use an example in Figure 4 to illustrate how we acquire historical activities in the reference window R. As shown, the snapshots s_1, s_2, s_3, s_4 fall in R. For each pair of consecutive snapshots, *i.e.*,$[s_1, s_2]$, $[s_2, s_3]$, $[s_3, s_4]$, we perform snapshot subtraction to obtain the historical activity during the respective time interval. For instance, for the snapshot pair $[s_1, s_2]$, we subtract s_1 from s_2 and obtain the historical activity, represented as a set of TCs: $\{m_1, m_2, m_4, m_6, m_7, m_8\}$. Note that the subtraction of two snapshots can be easily done by matching TC ids and subtracting the fields. With each historical activity, we can use kernel density estimation to infer k's occurrences at location l_C. As R contains multiple historical activities, and each can generate an estimation of keyword k's occurrences at location l_C, we obtain a set of estimations, denoted as $\Omega_t = \{\hat{N}_1(k), \hat{N}_2(k), \ldots, \hat{N}_c(k)\}$. Then we use z-score to quantify k's temporal burstiness:

$$z_t(k) = (N(k) - \mu_{\Omega_t})/\sigma_{\Omega_t},$$

where $N(k)$ is k's actual number of occurrences in C, and μ_{Ω_t} and σ_{Ω_t} are the mean and standard deviation of Ω_t.

Spatial burstiness. To measure spatial burstiness, we horizontally compare all the candidates in Q. The rationale is that, among the spatially scattered candidates, a keyword k in candidate C is spatially bursty if k's proportion in C is significantly higher than in other candidates. Given n candidate events $C_1, C_2, \ldots, C_n$, let P_i denote the keyword probability distribution of candidate C_i. With $\Omega_s = \{P_1(k), P_2(k), \ldots, P_n(k)\}$, we compute the spatial burstiness of keyword k in candidate C_i as:

$$z_s(k) = (P_i(k) - \mu_{\Omega_s})/\sigma_{\Omega_s},$$

where μ_{Ω_s} and σ_{Ω_s} are the mean and standard deviation of Ω_s. The underlying assumption of computing the z-score as the spatial burstiness (as well as the temporal burstiness) is that, the fraction of any keyword across different regions (days) follows a normal distribution. Such an assumption is reasonable given the regularity and periodicity underlying people's everyday activities. Under such an assumption, a large z-score typically reflects certain unusual burst of the keyword, and could be good indicators for local events.

Static features. For each candidate C, we also extract the following static features:

(1) $|C|$: the total number of tweets in C.
(2) $STD_t|C|$: the standard deviation of the timestamps of the tweets in C.
(3) $STD_{lat}|C|$: the standard deviation of the latitudes of the tweets in C.
(4) $STD_{lng}|C|$: the standard deviation of the longitudes of the tweets in C.

The classification procedure. With the above features, we use logistic regression to train a binary classifier and judge whether each candidate is indeed a local event. We choose logistic regression because of its robustness when there is only a limited amount of training data. While we have also tried using other classifiers like Random Forest and SVM, we find that the logistic regression classifier produces the best result in our experiments. The labeled instances for the classifier are collected through a large-scale experiment on a popular crowdsourcing platform. We will shortly detail the annotation process in Section 6.

We analyze the complexity of the candidate classification step as follows. As the prediction time of logistic regression is linear in the number of features and has $O(1)$ complexity, the time cost is dominated by the feature extraction process. Let N_C be the maximum number of tweets in each candidate, and M be the keyword vocabulary size, D be the latent embedding dimension, and N_Q be the number of tweets in the query window. We need to extract the features for all the candidates in the query window. The time costs for extracting different features for each candidate event are analzyed as follows: (1) For the temporal unusualness measure, its time complexity is $O(M + N_A + D)$ where N_A is the maximum number of TCs in one snapshot of the activity timeline; (2) For the spatial unusualness measure, its time complexity is $O(M + N_Q + D)$; (3) For the temporal burstiness measure, its time complexity is $O(MN_A)$; (4) For the spatial burstiness measure, its time complexity is $O(MN_C)$; (5) For the static features, the total time complexity is $O(N_C)$.

5 THE ONLINE UPDATER

In this section, we present the online updater of GeoBurst+. Consider a query window Q, let Q' be the new query window after Q shifts. Instead of finding the local events in Q' from scratch, the online updater leverages the results in Q and updates the event list with little cost.

If one runs the batch detection algorithm in the updated window Q', the candidate generation step will dominate the total time cost in the two-step detection process, while the candidate classification step is very efficient. Hence, our focus for supporting efficient online detection is to develop algorithms that can fast update the geo-topical clustering results when the query window shifts from Q to Q'.

To guarantee generating the correct clustering results in Q', the key is to find the new pivots in the new window Q' based on the previous results in Q. Let D_Q be the tweets falling in Q and D'_Q be the tweets in Q'. We denote by R_Q the tweets removed from D_Q, *i.e.*, $R_Q = D_Q - D'_Q$; and by I_Q the tweets inserted into D_Q, *i.e.*, $I_Q = D'_Q - D_Q$. In the sequel, we design a strategy that finds pivots in D'_Q by just processing R_Q and I_Q. Recall that, the pivot seeking process first computes the local pivot for each tweet and then performs authority ascent via a path of local pivots. So long as the local pivot information is correctly maintained for each tweet, the authority ascent can be fast completed. The major idea for avoiding finding pivots from scratch is that, as D_Q is changed to D'_Q, only a number of tweets have their local pivots changed. We call them mutated tweets, defined as follows.

Definition 5.1 (Mutated Tweet). A tweet $d \in D'_Q$ is a mutated tweet if d's local pivot in D'_Q is different from its local pivot in D_Q.

Now the questions is, how do we fast identify the mutated tweets by analyzing the influence of R_Q and I_Q? Our observation is that, for any tweet, it can become a mutated tweet only if at least one of its neighbors has authority change. Therefore, we take a *reverse search* strategy to find mutated tweets: (1) First, we identify in D'_Q all the tweets whose authorities have changed. (2) Second, for each authority-changed tweet t, we retrieve the tweets that regard t as its neighbor, and update their local pivots. Hence, the remaining issue is just to find the authority-changed tweets. In what follows, we handle R_Q and I_Q to this end.

Handling deletions. The deletion of a tweet $d \in R_Q$ can cause authority change in two ways. First, for the tweets having d as a neighbor in D_Q, their authorities decrease. Second, the keyword co-occurrence graph may evolve because of deleting d. As a result, the vicinities of certain keywords need to be recomputed and the authorities of corresponding tweets may change. The first case can be easily handled due to the additive property of authority. When d is deleted, we simply retrieve the tweets having d as a neighbor in D_Q. For each of those tweets, we subtract d's contribution from the authority score. For the second case, the key is to identify the keywords that need vicinity recomputation. Let us look at an example in Figure 5. If d contains two keywords e_1 and e_2, deleting d would decrease the weight of the edge $[e_1, e_2]$. For any other keywords having e_1 or e_2 in their old vicinities (e_3 and e_4 in this example), we mark them as to-recompute keywords. However, we defer the computation of their vicinities until I_Q is handled to identify the complete set of to-recompute keywords.

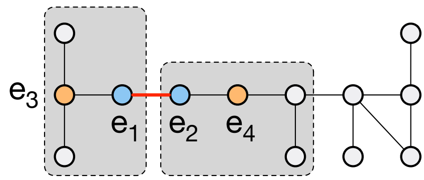


Fig. 5. Updating the keyword co-occurrence graph and keyword vicinities.

Handling insertions. A new tweet $d \in I_Q$ can also cause authority changes in two ways: (1) increasing the authority of the tweets that regard d as a neighbor; and (2) making the keyword co-occurrence graph evolve. Here, we need to first deal with the second case to ensure authority computation in the first case is based on the updated keyword vicinities. Similarly, we identify the keywords whose attaching edges have weight change, and mark other keywords that include such keywords in their vicinities. After all the to-recompute keywords are identified, we call Algorithm 2 to obtain their new vicinities. Once the keyword vicinities are updated, we retrieve the affected tweet pairs and update the corresponding authority scores. For the second case, now that the keyword vicinities have already been updated, for the inserted tweet d, we simply find which other tweets having d as their neighbor, and then add d's contribution to their authorities.

6 EXPERIMENTS

6.1 Experimental Settings

Compared methods. We compare GeoBurst+ with the following methods:

- EvenTweet [1] extracts bursty and localized keywords as features, and then clusters those features based on spatial distributions.
- Wavelet [6] uses wavelet transform to identify spatiotemporally bursty keywords and then clusters them by considering both co-occurrence and spatiotemporal distribution.
- GeoBurst [41] is a preliminary version of GeoBurst+. It neither uses embedding to capture textual semantics, nor has the classification module for accurate event identification. Instead, it heuristically ranks all the candidates by the weighted combination of the spatial burstiness and temporal burstiness.
- GeoBurst* is an adapted version of GeoBurst+, which does not use the features generated by the embedding learner (*i.e.*, the temporal unusualness and spatial unusualness) for candidate event classification.

Data Sets and Ground Truth. Our experiments are based on two real-life data sets, both of which are crawled using Twitter Streaming API during 2014.08.01 — 2014.11.30. The first data set, referred to as NY, consists of 6.41 million geo-tagged tweets in New York (after removing the tweets that do not have any verbs or nouns). The second data set, referred to as LA, consists of 5.53 million geo-tagged tweets in Los Angeles.

To evaluate the performance of different local event detection methods, we randomly generate 160 query time windows that are non-overlapping. We generated those queries with four different lengths: 3-hour, 4-hour, 5-hour, and 6-hour; and there are thus 40 queries for each query length. As all the methods require a reference window, we use a one-week reference window right preceding each query.

Now we describe the process for collecting groundtruth local events on NY and LA using a crowdsourcing platform. For every query, we run the methods to retrieve local events on the two data sets, and upload the results to CrowdFlower[1], a popular crowdsourcing platform, for evaluation. For GeoBurst+ and its variants, we ran both the batch mode and online mode to detect local events in the query window, and found these two modes produce exactly the same results. Thus, we only upload the results produced by the online mode and report its effectiveness. On CrowdFlower, we represent each event with 5 most representative tweets as well as 10 representative keywords, and ask three CrowdFlower workers to judge whether the event is indeed a local event or not. To ensure the quality of the workers, we label 20 queries for groundtruth judgments on each data set, such that only the workers who can achieve no less than 80% accuracy on the groundtruth can submit their answers. Finally, we use majority voting to aggregate the workers' answers. The representative tweets and keywords are selected as follows: (1) For GeoBurst+ and its variants, each event is a cluster of tweets, we select the 5 tweets having the largest authority scores, and the 10 keywords having the largest TF-IDF weights. (2) EvenTweet represents each event as a group of keywords. We select top-10 keywords in each event. Then we regard the group of keywords as a query to retrieve the top-5 most similar tweets using the BM25 retrieval model. (3) Wavelet represents an event with both keywords and matching tweets. We simply select the top-5 tweets and the top-10 keywords.

As both GeoBurst+ and GeoBurst* are supervised methods, we need to obtain training data for the candidate classifiers. The process for collecting the ground-truth events is described as follows: after gathering judgments from CrowdFlower, we rank the 160 query windows in chronological order. We train the candidate event classifiers for GeoBurst+ and GeoBurst* using the labeled candidates from the first 80 queries, and used the labeled data from the remaining 80 queries for evaluating all the methods.

Parameters. There are three major parameters in GeoBurst+: (1) the kernel bandwidth h; (2) the restart probability α; and (3) the RWR similarity threshold δ. We set $h = 0.01$, $\alpha = 0.2$, and $\delta = 0.02$. We have tuned these parameters and finally set them to these values because of the following reasons: (1) α specifies the restart probability during the random walk with restart process. To ensure good performance of the RWR measure, it is common to set it to the range [0.1, 0.3]. After tuning it on our data, we find that $\alpha = 0.2$ produces quality geo-topical clusters; (2) h controls the spatial granularity of geo-topical clustering process. With $h = 0.01$, we find that the geo-topic clusters are geographically compact enough; and (3) δ controls the semantic coherence of the results clusters. We observe that setting δ into the range $[0.01, 0.025]$ produces clusters that are of high quality. A too large δ imposes a too strong constraint that could split relevant tweets into different clusters; while a too small δ could make the clusters too coarse-grained such that the tweets about different activities are grouped into the same cluster.

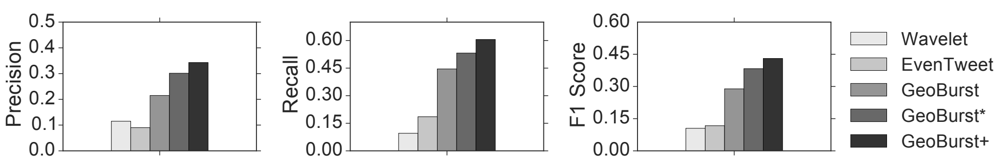
EvenTweet partitions the whole space into $N \times N$ small grids. We find N is EvenTweet's most sensitive parameter, and set $N = 50$ after tuning. For Wavelet, the most sensitive parameters are the granularities for constructing the spatiotemporal signal. After tuning, we set the space partitioning granularity to $\delta_x = 0.1, \delta_y = 0.1$; and the time granularity to $\delta_t = 3$ hours. For GeoBurst, it shares the three parameters with GeoBurst+, but has one more parameter η balancing the spatial and temporal burstiness in the ranking module. By default, we set $\eta = 0.5$.

6.2 Effectiveness Study

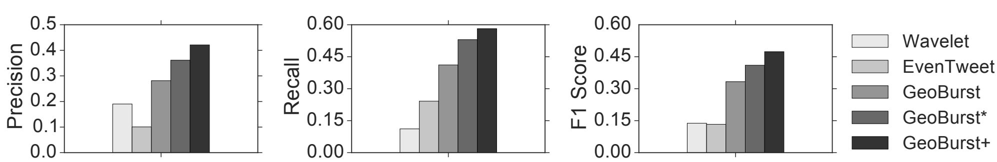
6.2.1 Quantitative Comparison. As aforementioned, after generating the 160 queries, we use the labeled data in the last 80 query windows for evaluation. To quantify the performance of all the methods, we report the following metrics:

[1] http://www.crowdflower.com/

(1) The detection precision is computed as $P = N_{\text{true}}/N_{\text{report}}$, where N_{true} is the number of true local events in the result list and N_{report} is the number of reported events.
(2) While the precision is easy to compute, the detection recall is hard to obtain due to the lack of the comprehensive set of local events in a given query window. We thus propose to measure the pseudo recall for each method. Specifically, for each query window, we aggregate all the true local events detected by different methods. Let N_{total} be the total number of the distinct local events detected by all the methods; we compute the pseudo recall of each method as $R = N_{\text{true}}/N_{\text{total}}$.
(3) Finally, we also report the F1 score of each method, which is simply computed as $F1 = 2 * P * R/(P + R)$.



(a) Performance comparison on NY.

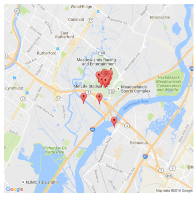


(b) Performance comparison on LA.

Fig. 6. Comparing the detection precision, recall, and F1 score of different methods on NY and LA.

Figure 6 shows the precisions, recalls, and F1-scores of all the methods on NY and LA. Comparing the five methods, we find that GeoBurst+ significantly outperforms the baseline methods on both data sets. The huge improvements indicate the superiority of GeoBurst+'s two-step scheme: (1) the candidate generation step ensures a good coverage of all potential local events; and (2) the classification step effectively pinpoints the true local events based on the features that captures the burstiness and unusualness of each candidate event.

Comparing the performance of GeoBurst+ and its variants, we find that GeoBurst+ outperforms GeoBurst by as much as 42.3% percent. Such a performance gap demonstrates that the features (*i.e.*, temporal unusualness and spatial unusualness) extracted from the embedding module indeed capture the characteristics of local events. Meanwhile, the classification procedure effectively leverages the extracted features to pinpoint true local events from the candidate set. GeoBurst* has better performance than GeoBurst, but is outperformed by GeoBurst+ considerably. This phenomenon further suggests that the two features generated by the embedding module play an important role in the classification process. Overall, Wavelet and EvenTweet perform much poorer than GeoBurst+ and its variants. For Wavelet, it is more suitable for detecting local events in a long time span. When the query windows are short, Wavelet fails to extract the less bursty but still interesting keywords. For EvenTweet, it deals with the text part by simply considering each keyword as an independent item, and thus fails to capture the intrinsic correlations among the keywords.

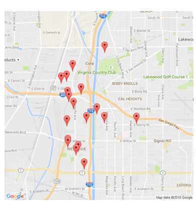


- **Giants** vs. **Patriots** #metlife #nfl @ **MetLife Stadium**
- **GIANTS** WIN? #**giants** #football #metlife #winning @ **MetLife Stadium** http://t.co/3MqBIROTHJ
- **Giants** v. **Patriots** pre-season @ **MetLife Stadium** http://t.co/dZUhLVrjGD
- **Giants** vs **Patriots** who do I root for...... #firstprofootballgame @ MetLife??? http://t.co/EcAw2GCVRn
- Enjoying the game with my fav girl. Let's go **Pats!!**

(a) NY local event I: the football game between the Giants and the Patriots.

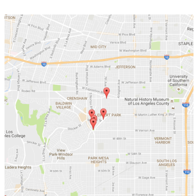
- **Electric Zoo** Sunday before it got cancelled again @ **Electric Zoo Festival**, Randalls Island Park http://t.co/UCQh83SmHf
- **EZoo Day** 2! #EZOO6 @ **Electric Zoo Festival**, Randalls Island Park
- Wonderful and kind couple. You guys are the bomb! #Ezoo @ **Electric Zoo Festival**, Randalls Island Park http://t.co/IYtylgnY9u
- A blast in a glass ! #ezoo #confetti #ezoonyc @ **Electric Zoo Festival**
- Thanking you for another great weekend @ **Electric Zoo Festival**

(b) NY local event II: the Electric Zoo Festival at the Randalls Island Park.



- Whoa....**earthquake**? Anyone feel it!?
- Ok I def felt an **earthquake**
- Shit **earthquake**..
- I usually never feel **earthquakes** but I felt that one. In San Perdo, CA
- I think I'm tripping ... i thought I felt an **earthquake** haha

(c) LA local event I: a level 3.3 earthquake near San Pedro.



- At The Stop The **Violence Rally** at **Leimert Park** #mikebrow
- #justiceformikebrown @ **Leimert Park** 2014 http://t.co/KT6OpOkTRU
- **#MikeBrown** #Ferguson #WeStandWithYou #LAprotest @ **Leimert Plaza Park** http://t.co/jQc3H3wnIZ
- Angelenos are ready to rise up. #NMOS14 #DontShoot #HandsUp #JusticeForMikeBrown @ **Leimert Plaza Park**
- Although the **rally** was bullshit. @ilusttv and I look good. @ **Leimert Park**

(d) LA local event II: a protesting rally at the Leimert Park.

Fig. 7. Example local events detected by GeoBurst+ on the NY and LA data sets. For each event, we plot the locations of the member tweets and show the top five tweets that have the largest authority scores.

6.2.2 Case Studies. In this subsection, we illustrate the example local events detected by GeoBurst+ on NY and LA. For each event, we plot the locations of the member tweets and select the top five tweets that have the largest authority scores. Figure 7(a) and 7(b) show two local events detected on NY: 1) the football game between the Giants and the Patriots; and 2) the Electric Zoo Festival. Examining the detected local events, one can see the generated geo-topical clusters are of high quality: the tweets in each cluster are both geographically compact and semantically coherent. Interestingly, GeoBurst+ can group the tweets that discuss about the topic using different keywords (*e.g.*, "Pats" and "Patriots"). This is because the RWR measure effectively captures the subtle semantic correlations between keywords. Another observation is that, the pivot tweets of each cluster are highly interpretable. This is because such high-quality tweets mention most important keywords about the topic and locate closely to the occurring spot, thereby receiving high authority scores.

Figure 7(c) and 7(d) show two local events detected on the LA data set. The first is an earthquake occurred in the San Pedro area; and the second is a protesting rally held at the Leimert Park to fight for Mike Brown. Again, we can see the representative tweets are highly interpretable. Meanwhile, the locations of the earthquake event are more scattered, while the locations of the protest event are very concentrated around the Leimert Park area. Such a phenomenon is explained by the fact that the earthquake influences a much larger geographical scope than the protest event, and GeoBurst+ can robustly detected the local events that have different scopes.

6.3 Efficiency Study

6.3.1 Running time comparison. We first compare the running time of different methods, by generating 500 random queries with different lengths and reporting the running time of each method. As the running time of GeoBurst+ and GeoBurst* are almost the same, we omit the results for GeoBurst*. We run GeoBurst+ in both batch mode and online mode. Given a query window Q, the batch mode performs candidate generation and classification in Q; the online mode considers a window Q' that precedes Q by 10 minutes, and finds local events in Q by updating the results in Q'.

Figure 8 shows the running time of all the methods on NY and LA. We observe that GeoBurst+ is much more efficient than EvenTweet and Wavelet even when in the batch mode. This phenomenon is explained by two facts. First, in the candidate generation step, the approximate RWR computation strategy can effectively speed up the pivot seeking process. Second, in the classification step, GeoBurst+ just uses a number of historical activities to extract the feature set, which is very efficient. Meanwhile, the online mode is even much faster than the batch mode. This is expected as the online mode does not need to find pivots from scratch in the time-consuming candidate generation step, but just needs to process the updated tweets and can achieve excellent efficiency. The batch mode of GeoBurst is a bit more efficient than GeoBurst+, because GeoBurst+ needs to extract embedding-based features in addition to the spatial and temporal burstiness and thus incurs extra overhead. Nevertheless, the marginal efficiency overhead of GeoBurst+ brings about large improvements in detection effectiveness and is thus cost-effective.

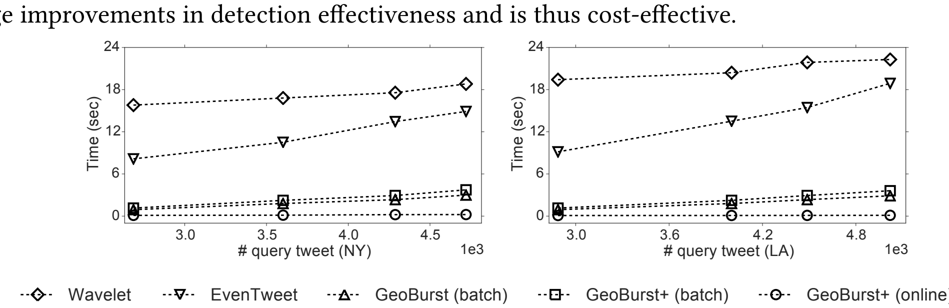
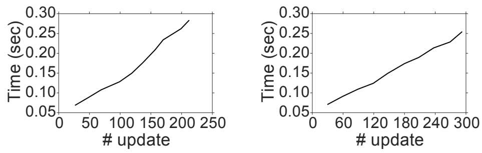


Fig. 8. Running time v.s. # tweets in the query window.

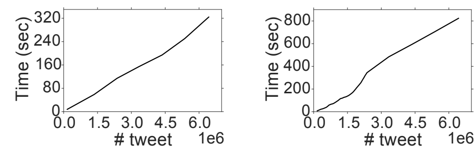
The major overhead of EvenTweet and Wavelet is due to their space partitioning strategy. Specifically, EvenTweet needs to compute spatial entropy to select localized keywords and perform clustering based on keyword spatial distributions; Wavelet needs to perform wavelet transform on the spatiotemporal signal and compute the spatiotemporal KL-divergence between keywords. One may propose to partition the space at a coarser granularity to improve the running time of the two methods, but that comes with the price of being much less effective.

6.3.2 Throughput Study. In Figure 9, we report the scalability of GeoBurst+'s online mode in terms of the number of updates: $N_{\text{update}} = N_{\text{delete}} + N_{\text{inserte}}$. To this end, we choose a 3-hour query



(a) Time v.s. N_{update} (NY). (b) Time v.s. N_{update} (LA).

Fig. 9. Throughput of GeoBurst+'s online mode.



(a) Embedding learning. (b) Actiity timeline.

Fig. 10. Time cost of stream summarization (NY).

window Q. Then we use a window Q' that precedes Q by 1, 2, . . ., 10 minutes, respectively, and update the results in Q'. One can observe that the running time of the online mode shows good scalability with the number of updates. For example, when there are as many as 212 updates, the online mode takes just 0.337 second to finish on the NY data set. Such performance suggests that, GeoBurst+ is capable of continuously monitoring the stream and realizing real-time detection.

To study the throughput of GeoBurst+'s summarization module, we apply it to process the continuous streams of NY and LA, and periodically record the number of tweets processed so far and the time for summarization. As the summarization consists of embedding learning and activity timeline construction, we report the time cost for each of them *w.r.t* the number of processed tweets. With the NY data set, Figure 10(a) and 10(b) show the scalability for embedding learning and activity timeline construction, respectively. One can observe that, for the three-month tweets in New York, GeoBurst+ learnt the embeddings in 330.82 seconds and constructed the activity timeline in 831.85 seconds, and both operations scale well with the number of tweets. The results and trends are similar on LA, we omit them to save space.

7 RELATED WORK

7.1 Global Event Detection

Global event detection aims at extracting events that are bursty and unusual in the entire tweet stream. Existing approaches to this end can be classified into two categories: *document-based* and *feature-based*. Document-based approaches consider each document as a basic unit and group similar documents to form events. Allan *et al.* [4] perform single-pass clustering of the stream, and use a similarity threshold to determine whether a new document should form a new topic or be merged into an existing one. Aggarwal *et al.* [3] also detect events by continuously clustering the tweet stream, but their similarity measure considers both tweet content relevance and user proximity. Sankaranarayanan *et al.* [30] train a Naïve Bayes filter to identify news-related tweets, and cluster them based on TF-IDF similarity. They also enrich each piece of news with location information by extracting geo-entities. Feature-based approaches [11], [14], [24], [33], [21] identify a set of bursty features (*e.g.*, keywords) from the stream and cluster them into events. Fung *et al.* [11] model feature occurrences with binomial distribution to extract bursty features. He *et al.* [14] construct the time series for each feature and perform Fourier Transform to identify bursts. Weng *et al.* [33] use wavelet transform and auto-correlation to measure word energy and extract high-energy words. Li *et al.* [21] segment each tweet into meaningful phrases and extract bursty phrases based on frequency, which are clustered into candidate events and further filtered using Wikipedia. The above methods are all designed for detecting global events that are bursty in the entire stream. As aforementioned, a local event is usually bursty in a small geographical region instead of the entire stream. Hence, directly applying these methods to the geo-tagged tweet stream would miss many local events. There has also been work [29], [27], [22] on detecting specific types of events. Sakaki *et al.* [29] investigate real-time earthquake detection. A classifier is trained to judge whether an incoming tweet is related to earthquake or not, and an alarm is released when the number of earthquake-related tweets is large. Li *et al.* [22] detect crime and disaster events (CDE) with a self-adaptive crawler that dynamically retrieves CDE-related tweets. Different from those studies, we aim to detect all kinds of local events from the stream.

7.2 Local Topic and Event Detection

There have been quite a few studies that model the topics/activities in different regions with geo-tagged social media. Specifically, Sizov *et al.* [31] extends LDA [5] by assuming each latent topic has a multinomial distribution over text, and two Gaussians over latitudes and longitudes. They later extend the model to find topics that have complex and non-Gaussian distributions [18]. Yin *et al.* [34] extend PLSA by assuming each region has a normal distribution that generates locations, as well as a multinomial distribution over the latent topics that generate text. Guo *et al.* [13] uses Dirichlet Process to extract activities that freely span several regions and peaks multiple times. Zhang *et al.* [39] propose a cross-modal embedding framework for uncovering the typical activities in different geographical regions and time periods. While the above models are designed to detect macro-level geographical topics, Hong *et al.* [15] and Yuan *et al.* [35] introduce the user factor in the modeling process such that micro-level user preferences can be inferred. There is a clear difference between geographical topic modeling and local event detection. The former attempts to summarize the *typical* activities in different regions, whereas the latter aims at extracting *unusual* activities bursted in local areas.

Watanabe *et al.* [32] and Quezada *et al.* [28] study location-aware events in the social media, but their major focus is on geo-locating tweets/events, whereas we aim to automatically extract local events from raw geo-tagged tweets. Chen *et al.* [6] extract events from geo-tagged Flickr photos. By converting the spatiotemporal distribution of each tag into a 3-dimensional signal, they perform wavelet transform to extract spatiotemporally bursty tags, and clusters those tags into events based on co-occurrence as well as spatiotemporal distributions. Such a method, however, can only detect local events in batch manner. Krumm *et al.* [19] propose the detection of spatiotemporal spikes in the tweet stream as local events. Nevertheless, their approach can only detect events for pre-defined rigid time windows (*e.g.*, 3-6 pm, 6-9 pm), because it discretizes time and compares the number of tweets in the same bin across different days. It supports neither ad-hoc query windows nor real-time detection. Abdelhaq *et al.* [1] propose EvenTweet, which first extracts bursty and localized keywords and then clusters such keywords based on their spatial distributions. Unfortunately, EvenTweet suffers from two drawbacks. First, the clustering of localized keywords is merely based on spatial distribution without considering tweet content. It results in irrelevant keywords in the same cluster, and cannot distinguish different events that occur at the same location. Second, although EvenTweet is an online method, it is incapable of detecting local events in real time, as the detection is triggered only when the current window is saturated. A preliminary version of GeoBurst+ is introduced in [41]. However, the GeoBurst method proposed in [41] does not leverage embedding learning to capture short-text semantics and is meanwhile unsupervised. The embedding learner, the classification procedure, and the more systematic evaluations are all new in this paper.

7.3 Local Event Forecasting

Local event forecasting is another line of research that is related to our problem. Foley *et al.* [10] use distant supervision to extract future local events from Web pages, but the proposed method can only extract local events that are well advertised in advance on the Web. Muthiah *et al.* [26]

and Zhao *et al.* [43], [44], [42] have developed a bunch of methods and the EMBERS system for forcasting local events. They formulate local event forecasting as a binary prediction problem, *i.e.*, predicting whether a specific type of event (*e.g.*, civil unrest) will occur on a given day. Their methods combine social media with other data sources (*e.g.*, gold standard report, news articles) to train reliable predictors. Our problem is orthogonal to their studies in that, instead of performing binary prediction for a specific event type, we attempt to extract all types of local events from the geo-tagged tweet data alone.

8 CONCLUSION

We studied the problem of real-time local event detection in geo-tagged tweet streams. We proposed the GeoBurst+ detector. To the best of our knowledge, GeoBurst+ is the first method that is capable of extracting highly interpretable local events in real time. GeoBurst+ first generates candidate events based on a novel pivot seeking process, and then leverages the continuous summarization of the stream as background knowledge to classify the candidates. Our extensive experiments have demonstrated that GeoBurst+ is highly effective and efficient. The usage of GeoBurst+ is not limited to Twitter. Rather, any geo-textual social media stream (*e.g.*, Instagram photo tags, Facebook posts) can use GeoBurst+ to extract interesting local events as well.

9 ACKNOWLEDGEMENTS

This work was sponsored in part by the U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), National Science Foundation IIS-1017362, IIS-1320617, and IIS-1354329, HDTRA1-10-1-0120, NSFC (Grant No. 61572488), and Grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov), and MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Hamed Abdelhaq, Christian Sengstock, and Michael Gertz. 2013. Eventweet: Online localized event detection from twitter. *PVLDB* 6, 12 (2013), 1326–1329.

[2] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. 2003. A framework for clustering evolving data streams. In *VLDB*. 81–92.

[3] Charu C Aggarwal and Karthik Subbian. 2012. Event Detection in Social Streams.. In *SDM*. 624–635.

[4] James Allan, Ron Papka, and Victor Lavrenko. 1998. On-Line New Event Detection and Tracking. In *SIGIR*. 37–45.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3, 1 (2003), 993–1022.

[6] Ling Chen and Abhishek Roy. 2009. Event detection from flickr data through wavelet-based spatial analysis. In *CIKM*. 523–532.

[7] Dorin Comaniciu and Peter Meer. 1999. Mean shift analysis and applications. In *ICCV*. 1197–1203.

[8] Son Doan, Bao-Khanh Ho Vo, and Nigel Collier. 2012. An analysis of Twitter messages in the 2011 Tohoku Earthquake. In *Electronic Healthcare*. Springer, 58–66.

[9] Wei Feng, Chao Zhang, Wei Zhang, Jiawei Han, Jianyong Wang, Charu Aggarwal, and Jianbin Huang. 2015. STREAMCUBE: Hierarchical spatio-temporal hashtag clustering for event exploration over the Twitter stream. In *ICDE*. 1561–1572.

[10] John Foley, Michael Bendersky, and Vanja Josifovski. 2015. Learning to Extract Local Events from the Web. In *SIGIR*. 423–432.

[11] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Philip S Yu, and Hongjun Lu. 2005. Parameter free bursty events detection in text streams. In *VLDB*. 181–192.

[12] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *ACL*. 42–47.
[13] Jinjin Guo and Zhiguo Gong. 2016. A Nonparametric Model for Event Discovery in the Geospatial-Temporal Space. In *CIKM*. 499–508.
[14] Qi He, Kuiyu Chang, and Ee-Peng Lim. 2007. Analyzing feature trajectories for event detection. In *SIGIR*. 207–214.
[15] Liangjie Hong, Amr Ahmed, Siva Gurumurthy, Alexander J Smola, and Kostas Tsioutsiouliklis. 2012. Discovering geographical topics in the twitter stream. In *WWW*. 769–778.
[16] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *WWW*. 271–279.
[17] Wei Kang, Anthony K. H. Tung, Wei Chen, Xinyu Li, Qiyue Song, Chao Zhang, Feng Zhao, and Xiajuan Zhou. 2014. Trendspedia: An Internet observatory for analyzing and visualizing the evolving web. In *ICDE*. 1206–1209.
[18] Christoph Carl Kling, Jérôme Kunegis, Sergej Sizov, and Steffen Staab. 2014. Detecting non-gaussian geographical topics in tagged photo collections. In *WSDM*. 603–612.
[19] John Krumm and Eric Horvitz. 2015. Eyewitness: Identifying Local Events via Space-Time Signals in Twitter Feeds. In *SIGSPATIAL*.
[20] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.
[21] Chenliang Li, Aixin Sun, and Anwitaman Datta. 2012. Twevent: segment-based event detection from tweets. In *CIKM*. 155–164.
[22] Rui Li, Kin Hou Lei, Ravi Khadiwala, and KC-C Chang. 2012. Tedas: A twitter-based event detection and analysis system. In *ICDE*. 1273–1276.
[23] Peter Lofgren and Ashish Goel. 2013. Personalized pagerank to a target node. *arXiv:1304.4658* (2013).
[24] Michael Mathioudakis and Nick Koudas. 2010. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*. 1155–1158.
[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[26] Sathappan Muthiah, Patrick Butler, Rupinder Paul Khandpur, Parang Saraf, Nathan Self, Alla Rozovskaya, Liang Zhao, Jose Cadena, Chang-Tien Lu, Anil Vullikanti, Achla Marathe, Kristen Maria Summers, Graham Katz, Andy Doyle, Jaime Arredondo, Dipak K. Gupta, David Mares, and Naren Ramakrishnan. 2016. EMBERS at 4 years: Experiences operating an Open Source Indicators Forecasting System. In *KDD*. 205–214.
[27] Swit Phuvipadawat and Tsuyoshi Murata. 2010. Breaking news detection and tracking in twitter. In *WI-IAT*. 120–123.
[28] Mauricio Quezada, Vanessa Peña-Araya, and Barbara Poblete. 2015. Location-aware model for news events in social media. In *SIGIR*. 935–938.
[29] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. In *WWW*. 851–860.
[30] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. 2009. Twitterstand: news in tweets. In *GIS*. 42–51.
[31] Sergej Sizov. 2010. GeoFolk: latent spatial semantics in web 2.0 social media. In *WSDM*. 281–290.
[32] Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. 2011. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM*. 2541–2544.
[33] Jianshu Weng and Bu-Sung Lee. 2011. Event Detection in Twitter.. In *ICWSM*. 401–408.
[34] Zhijun Yin, Liangliang Cao, Jiawei Han, Chengxiang Zhai, and Thomas S. Huang. 2011. Geographical topic discovery and comparison. In *WWW*. 247–256.
[35] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. 2013. Who, where, when and what: discover spatio-temporal topics for twitter users. In *KDD*. 605–613.
[36] Quan Yuan, Wei Zhang, Chao Zhang, Xinhe Geng, Gao Cong, and Jiawei Han. 2017. PRED: Periodic Region Detection for Mobility Modeling of Social Media Users. In *WSDM*. 263–272.
[37] Chao Zhang, Jiawei Han, Lidan Shou, Jiajun Lu, and Thomas F. La Porta. 2014. Splitter: Mining Fine-Grained Sequential Patterns in Semantic Trajectories. *PVLDB* 7, 9 (2014), 769–780.
[38] Chao Zhang, Shan Jiang, Yucheng Chen, Yidan Sun, and Jiawei Han. 2015. Fast Inbound Top-K Query for Random Walk with Restart. In *ECML/PKDD*. 608–624.
[39] Chao Zhang, Keyang Zhang, Quan Yuan, Haoruo Peng, Yu Zheng, Tim Hanratty, Shaowen Wang, and Jiawei Han. 2017. Regions, Periods, Activities: Uncovering Urban Dynamics via Cross-Modal Representation Learning. In *WWW*.
[40] Chao Zhang, Keyang Zhang, Quan Yuan, Luming Zhang, Tim Hanratty, and Jiawei Han. 2016. GMove: Group-Level Mobility Modeling Using Geo-Tagged Social Media. In *KDD*. 1305–1314.

[41] Chao Zhang, Guangyu Zhou, Quan Yuan, Honglei Zhuang, Yu Zheng, Lance M. Kaplan, Shaowen Wang, and Jiawei Han. 2016. GeoBurst: Real-Time Local Event Detection in Geo-Tagged Tweet Streams. In *SIGIR*. 513–522.

[42] Liang Zhao, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2016. Multi-resolution Spatial Event Forecasting in Social Media. In *KDD*.

[43] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2015. Multi-Task Learning for Spatio-Temporal Event Forecasting. In *KDD*. 1503–1512.

[44] Liang Zhao, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2016. Hierarchical Incomplete Multi-source Feature Learning for Spatiotemporal Event Forecasting. In *KDD*. 2085–2094.

[45] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban Computing: Concepts, Methodologies, and Applications. *ACM TIST* 5, 3 (2014), 38:1–38:55.