Geoinformatica (2017) 21:365–388
DOI 10.1007/s10707-016-0258-x

Efficient online extraction of keywords for localized events in twitter

Hamed Abdelhaq[1] · Michael Gertz[2] · Ayser Armiti[1]

Received: 20 January 2016 / Accepted: 12 April 2016 / Published online: 30 April 2016

Abstract Messages published via social media sites, such as Twitter, Facebook, and Foursquare hide a considerable amount of information about real world events. The timely identification of such events from this huge, unstructured, and noisy user-generated content plays an important role in increasing situation awareness and in supporting useful applications such as recommendation systems. Interestingly, a large number of these messages are enriched with location information, due to the recent advancements of today's location acquisition techniques. This, in turn, enables location-aware event mining, i.e., the detection and tracking of localized events such as sport events, demonstrations, or traffic jams, to name but a few. The main building blocks of a localized event are local keywords that exhibit a surge in usage at the event location. In this paper, we propose an approach that aims at extracting local keywords from a stream of Twitter messages by (1) identifying local keywords, and (2) estimating the central location of each keyword. This extraction procedure is performed in an online fashion using a sliding window over the Twitter stream. Additionally, we address the problem of spatial outliers that adversely affect a sound identification of local keywords. Spatial outliers occur when people far away from the location of an event use related keywords in their Tweets. We handle this problem by adjusting the spatial distribution of keywords based on their co-occurrence with place names that may refer to the location of an event. To ensure scalability, we utilize a hierarchical spatial index to gradually prune the geographic space and thus to efficiently perform complex spatial computations. Extensive comparative experiments are conducted using Twitter data. The analysis of the experimental results demonstrates the superiority of our approach over existing methods in terms of efficiency and precision of the obtained results.

✉ Michael Gertz
gertz@informatik.uni-heidelberg.de

1 moovel Group GmbH, Stuttgart, Germany

2 Heidelberg University, Heidelberg, Germany

Keywords Local keywords · Localized event · Event detection · Social media

1 Introduction

The recent and rapid advancements of social media services such as Twitter and Facebook have provided virtual and easy-to-use channels for users to interact, to report their thoughts, and to share what is happening. To date, a large number of such social media services are being accessed by hundreds of millions of users who publish content on a daily basis. Microblogging services such as Twitter are among the leading services causing this dramatic increase in user-generated content. Users constantly publish brief textual updates, called "tweets", describing their daily activities and surrounding phenomena. As of December 2015, about 320 million monthly active users publish tweets using Twitter.[1] As a consequence, the detection and analysis of events from Twitter data is an active research area as it provides event information faster than via traditional news agencies [2, 17, 27, 32].

For example, Chunara and colleagues proved that the trends of the Cholera epidemic that occurred in Haiti in 2010 were observed in tweets two weeks earlier than respective information from official sources [12]. Therefore, the timely extraction of up-to-date event information from Twitter data helps increasing the situation awareness, supporting recommendation systems, and providing useful sentiment information about products [30]. During the time of an event, a number of event-related words typically exhibit an increase in usage. These words are referred to as bursty words (*keywords* for short) and are considered the main building blocks of detected events.

The introduction of GPS technology and the availability of GPS-enabled mobile devices contribute to increasing the number of geo-tagged tweets, providing almost precise information about the location from which tweets have been published. Currently, more than 3 % of the published tweets are geo-tagged, enabling the detection of another type of events, namely, *localized events*. A localized event, e.g., a music festival, a soccer game, or a traffic jam, is a real-world event where people (willingly or unwillingly) gather at a specific location for a limited period of time and observe what is going on. As a localized event takes place within a specific geographic region, its related keywords show an increase in usage at the event location. These keywords pertaining to localized events are referred to as *local keywords*. In this paper, we propose an approach to extract local keywords from a Twitter data stream, which includes (1) identifying them among other non-localized (key)words, and (2) estimating the central location of each local keyword in an online fashion.

However, a proper extraction of local keywords faces the following challenges:

1. Dynamics of Twitter content: tweets have a high arrival rate, which requires a scalable approach that can consume a huge stream of tweets and promptly update the state of each previously extracted local keyword.
2. Spatial outliers: a local keyword may exhibit a non-local behavior when people use it far away from the event location. A method is needed to adjust the spatial distribution for such a keyword in order to determine its actual local characteristics.
3. When two localized events on the same topic, e.g., two music concerts, take place at the same time, extracted keywords can be erroneously recognized as global keywords due to their occurrence at different locations.

[1] https://about.twitter.com/company, accessed Dec. 2015.

In this paper, we present a novel framework to efficiently and effectively extract local keywords. What distinguishes this framework from related approaches is that it performs a real-time extraction and tracking of local keywords at a fine-grained spatio-temporal resolution, and handles spatial outliers to obtain a reliable keyword locality decision. For this, a sliding window approach is employed to incrementally process incoming tweets. Therefore, the timeline is split into fixed-length snapshots such that the most recent c snapshots are covered by the time window. Once a new snapshot t elapses, the window slides and the following steps are performed:

1. Keyword extraction: a set of event-related words (called *keywords*) are extracted from the tweets published during the current snapshot t.
2. Geo-filter generation: for each keyword k, a probability distribution over the place names co-occurring with k is estimated and used to eliminate spatial outliers.
3. Spatial update: the statistics describing the spatial distribution of keywords are updated by both inserting the content of tweets published during the current snapshot t and removing the content of the expired snapshot $t - c$.
4. Local keyword extraction: geo-filters are utilized to adjust the spatial distributions of keywords. These distributions are then analyzed to test the locality of keywords and to compute the central location of each.

The contributions of this work are summarized as follows:

– Handling spatial outliers: a novel spatial regularization technique is proposed to adjust the spatial distribution of keywords.
– A hierarchical space-partitioning index structure is employed to ensure a fast pruning of the geographic space while checking the locality of keywords and estimating their spatial focus.
– An extensive experimental evaluation is conducted to demonstrate the effectiveness and efficiency of our approach in extracting local keywords and tracking their spatial evolution at a fine-grained spatio-temporal resolution.

The remainder of this paper is organized as follows. We first review related work in Section 2. The main concepts, notations, and the problem statement are given in Section 3. In Section 4, we describe the steps required to extract keywords from a batch of recent tweets. Then, in Section 5, we introduce the main mechanism used to check the locality of keywords, and we show how to generate geo-filters that can handle spatial outliers. In Section 6, we detail the steps to extract local keywords. The experimental evaluation of our framework is presented in Section 7. Finally, we conclude the paper and address ongoing work in Section 8.

2 Related work

An increasing interest in detecting and tracking real-world events from social media has recently been witnessed [11, 20, 21, 24, 30]. Event detection itself is a rather old research area encouraged by the Topic Detection and Tracking (TDT) initiative that aims at identifying events from text streams [5, 31]. Event detection techniques can be classified according to two main detection paradigms: (1) *document-pivot*: events are represented as clusters of semantically similar documents [9, 20, 27, 30]. (2) *feature-pivot*: representative features, e.g., bursty words, are extracted, and then clustered to form events [11, 13]. While some approaches detect events from already collected data (called *retrospective detection* [11, 28, 30]), others incrementally process a stream of messages in real time, i.e., they perform an *online detection* [3, 15, 22].

In order to study these approaches that have problem settings and objectives similar to ours, we narrow down the space of related work and focus only on feature-pivot based approaches that operate on social media data in an online fashion. First, in Section 2.1, we discuss efforts aimed at extracting event information using only the temporal dimension. Then, approaches that consider both spatio-temporal dimensions are discussed in Section 2.2.

2.1 Events over time

Li et al. [17] introduced Twevent, a framework to identify bursty word segments from tweets and to cluster them using a k-nearest neighbor graph. For this, they exploit Wikipedia to distinguish actual event clusters from noisy ones. The approach $BursT$ proposed by Lee et al. [16] scores words dynamically based on frequency statistics collected during a time window. The main disadvantage of $BursT$ is that it assigns high scores to regularly mentioned (familiar) words, e.g., “job”, “evening”, or “lol”. Approaches employing the discrepancy-based paradigm [2, 15] try to avoid such a problem by measuring the deviation between the observed frequency of a word and its expected usage baseline.

Aggarwal and Subbian [4] leverage both the underlying network structure of social media and the content of tweets to provide a more effective clustering scheme and to improve upon the content-only similarity metric. Furthermore, they use a sketch-based technique to efficiently compute the structural similarity between clusters. Cataldi et al. [10] model the life cycle of terms using a novel ageing theory and connect related emerging keywords after navigating a topic graph. For these approaches and similar contributions [6, 8], the spatial dimension is not considered, and hence, the extracted event-related keywords are typically of a global nature, thus related to large regions.

2.2 Events over time and space

The approach by Sakaki et al. [22] distinguishes event-related tweets from other tweets using a SVM classifier. Event tweets are modeled over time using a Poisson process to detect the occurrence of an event. Then, they apply Particle and Kalman filters to estimate the location of detected events. As for detecting a new type of event, the classifier is trained using a different set of event keywords, which is a limitation that stands against applying their approach on a broad range of events. Mathioudakis et al. [28] identify geographic locations showing a burst in a regular grid by minimizing a cost function to distinguish between bursty and non-bursty grid cells. Backstrom et al. [7] exploit the geo-coordinates (latitude and longitude) associated with Yahoo! query log entries to manifest and quantify the spatial variation in search queries using a generative probabilistic framework. However, these approaches are based on batch processing and do not scale well for the large volume and high arrival rate of new data.

A framework to analyze the spatio-temporal characteristics of keywords has been proposed by Abdelhaq and colleagues in [2]. Using this framework, a graph-based regularization procedure is used to estimate a reliable spatial distribution for each keyword after handling spatial noise and sparsity problems. Skovsgaard et al. [25] propose a scalable framework in support of processing top-k most popular term queries on a stream of geo-tagged and timestamped tweets in a user-specified spatio-temporal range. They also use an extension to frequent item counting techniques to adaptively maintain exact counts for terms

Table 1 Main notations used throughout this paper for extracting local keywords and estimating their central locations

Notation	Description
t	current snapshot
M^t	tweets published at t, m is a tweet in M^t
$\mathcal{W}$	sliding window
$\mathcal{W}^t$	set of words mentioned at snapshot t
$\mathcal{W}^t_w$	tweets published at t and contains word w
c	sliding window size (number of snapshots in $\mathcal{W}$)
v	history length
G	geographic space
K^t	keywords extracted at snapshot t
K^t_{local}	local keywords at t, $K^t_{local} \in K^t$
ϕ	locality threshold
F_k	spatial focus (central location) of keyword k
Ω	keyword-georeference map
$\mathcal{T}$	spatial index (pyramid space-partitioning index)

at various spatio-temporal granularities. Magdy et al. [18] introduce $Mercury$, a system to support real-time top-k spatio-temporal queries on tweets within a memory-constrained environment. $Mercury$ can help retrieve tweets related to a certain event provided that the event location and time is known in advance.

The approach described by Boettcher and Lee in [9] extracts a set of words W published during a time window from Twitter streams. Word subsets of length 1, 2, and 3 out of the power set of W are built. Then, the DBScan clustering algorithm is employed to find the subsets having limited spatial extent based on the geo-coordinates associated with the tweets of each potential event cluster. Finally, they train a classifier to filter out non-event clusters. Watanabe et al. [29] estimate the location of non-geotagged tweets to increase the chance of finding local keywords. Then, they search for place names and count the number of key terms that co-occur with each place name. However, their method fails to find localized events when no place names indicating the location of an event are given.

It is important to note that our framework not only explicitly considers the spatial and temporal dimensions to perform real-time extraction of localized keywords at fine spatio-temporal granularity, but also tracks their spatial evolution over time and handles their respective spatial outliers. In this paper, we extend our work in [1] by employing a space-partitioning index structure towards an efficient extraction of local keywords. We also cope with the difficulty in extracting local keywords when two or more events on the topic are taking place at the same time.

3 Background and problem statement

In this section, we first illustrate the temporal organization of tweets along with the notations used to access these tweets and the contained word statistics. A summary of these notations is given in Table 1. We conclude this section by presenting the problem statement.

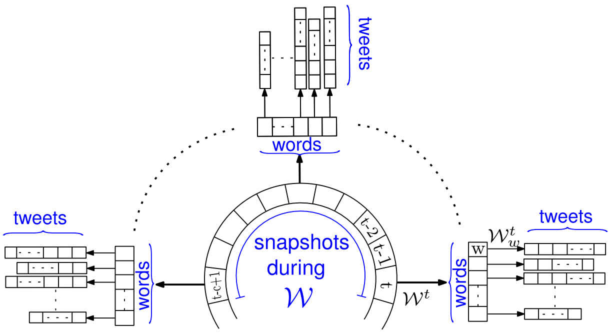


Fig. 1 The temporal index maintaining tweets published during sliding window $\mathcal{W}$. The key $\mathcal{W}^t$ of the inverted index on $\mathcal{W}$ points to a set of words. These words are also keys for another inverted index. The postings of the key $\mathcal{W}^t_w$ are the ids of tweets published at snapshots t and containing word w

3.1 Temporal organization of tweets

A tweet m arriving from a Twitter stream consists of a set of words S, a tweet id id, a creation time $time$, and a geo-coordinate $loc = (lat, lon)$, given as latitude/longitude pair. We assume that the components $(id, S, time)$ exist for each tweet, while only a small subset of the tweets are geo-tagged with loc. This is because the majority of tweets are published without geographic location due to privacy reasons or the lack of underlying positioning infrastructures. Although both geo- and non geo-tagged tweets are used to extract keywords, only geo-tagged tweets are used to determine the locality of a keyword and its central location. We assume that each geo-tagged tweet originates from within some geographic space G.

The time dimension is inherent and essential when dealing with data streams. To cope with the huge amount of incoming tweets, our approach exploits a sliding window on the stream of tweets and incrementally processes the most recently posted tweets. The timeline is split into fixed-length time intervals called snapshots $(\cdots, t-2, t-1, t)$, where t is the current snapshot. The last c snapshots form the sliding window $\mathcal{W}$ holding the most recent tweets to be processed.

Each tweet is indexed by the snapshot during which it was posted. That is, to access the tweets published during the current snapshot t, the key $\mathcal{W}^t$ is used. For efficiency reasons, we add another level of indexing where the tweets posted during snapshot t are further organized as an inverted index based on the words mentioned at t.

As illustrated in Fig. 1, the key $\mathcal{W}^t$ can be viewed as an index whose keys are the set of words mentioned in tweets M^t that are published at snapshot t, i.e.,

$$\mathcal{W}^t := \{w | m \in M^t \wedge w \in m.S\}.$$

The postings of key $\mathcal{W}^t_w$ in that index are the tweets published at t and contain word w. Formally,

$$\mathcal{W}^t_w := \{m | m \in M^t \wedge w \in m.S\}.$$

Using this two-level indexing, one can directly retrieve tweets published at a certain snapshot and containing a specific word.

3.2 Problem statement

Assume a set of keywords K^t has been extracted for snapshot t, the aim is to *identify the local keywords* K^t_{local} *and compute the focus* F_k *of each keyword* $k \in K^t_{local}$ at a fine-grained spatial resolution. The focus of a keyword k represents the geographic location at which k has the highest density within the geographic space G.

The sparse distribution of tweets over space, especially when a fine spatial granularity is used, forces us to aggregate the spatial statistics of keywords over a time window $\mathcal{W}$ in order to obtain statistically significant results. For this, an update mechanism is needed (1) to aggregate sufficient spatial statistics to identify local keywords and estimate their focus, and (2) to ensure an up-to-date state of these statistics. The update mechanism includes the insertion of statistics from the current snapshot t and the deletion of others from the expired snapshot $t - c$.

The fact that users may publish tweets containing a keyword far away from the location of an event leads to erroneously considering this keyword as non-local. We handle this issue based on the co-occurrence of such keywords with place names (georeferences) referring to the event location.

4 Keyword extraction

Keyword extraction from a stream of messages has recently been studied extensively due to its role in detecting real-world events from the content of social media (see, e.g., [13, 14, 30]). Since the task of keyword extraction is out of the scope of this paper, we only give a brief outline of the approach presented by Abdelhaq and colleagues in [3], which relies on a variation of the discrepancy paradigm. The discrepancy paradigm [14, 15] has been shown to efficiently extract keywords by measuring the deviation between the observed usage frequency of a word and its expected usage baseline. The higher the deviation, the more likely this word is considered bursty (and thus is a keyword). During snapshot t, the number of published tweets containing word w is denoted by $|\mathcal{W}^t_w|$. In addition, we have

$$history[w] := (|\mathcal{W}^1_w|, |\mathcal{W}^2_w|, ..., |\mathcal{W}^v_w|) \tag{1}$$

as a fixed historical sequence of usage frequencies for w collected before the current snapshot t, such that $v < t$. These frequencies are considered a baseline describing the normal (non-bursty) usage of w over previous snapshots.

The word usage baseline $b(w)$ is assumed to be constant and estimated from $history[w]$. The values of $history[w]$ are drawn from a Gaussian distribution with mean $b(w).\mu$ and standard deviation $b(w).\sigma$. The mean and the standard deviation of $b(w)$ are estimated using maximum likelihood on the usage values $history[w]$ of word w. Then, the z-score is used to measure the burstiness of a word w. Formally,

$$burst(w, t) := \frac{|\mathcal{W}^t_w| - \mu_w}{\sigma_w}. \tag{2}$$

As a result, each word $w \in \mathcal{W}^t$ whose burstiness is at least two standard deviations above the mean, i.e., $burst(w, t) \geq 2$, is considered a keyword. Since the region $\mu_w \pm 2\sigma_w$ contains 95 % of the data under the assumption of a normal distribution, the probability that $|\mathcal{W}^t_w|$ is generated from the normal distribution when $burst(w, t) \geq 2$ is $(1 - 0.95)/2 = 0.025$ % and, therefore, w can be identified as bursty.

The set of all keywords published during snapshot t is denoted K^t. If w is a new word that has been not observed in history, i.e., having $\mu_w = 0$ and $\sigma_w = 0$, we assume a prior noise level of $\sigma_w = |\mathcal{W}_w^t|/f$. That is, the initial noise level is $1/f$-th of the observed word signal. In our work, we set $f = \log_2(|\mathcal{W}_t^w| + 1)$, so that word w should be observed at least 3 times at snapshot t in order to treat it as a bursty word. This restriction on the number of occurrences of a new word is useful to mitigate the impact of the out-of-vocabulary words appearing frequently in social media.

A keyword is considered *local* if it appears in small parts of the geographic space. In this case, it is more likely that this keyword corresponds to a localized event. We denote the set of local keywords published at snapshot t as K_{local}^t. We will elaborate more on the locality of keywords in the following section.

5 Locality of keywords

In this section, we show how the geographic space G is partitioned and the spatial distribution of each extracted keyword $k \in K^t$ is computed. As we will show in Section 5.1, this spatial distribution is important to decide about the locality of keywords. A keyword-georeference map is then introduced in Section 5.2 and used to mitigate the problem of spatial outliers that adversely affect the expected distribution of keywords.

5.1 Entropy for locality check

It is assumed that local keywords have a limited spatial extent and appear only in small parts of the geographic space. To identify these keywords among others, we compute their distribution over space. For this, a given space G is partitioned using a regular grid with a particular cell width. The cell width is treated as a configurable parameter to be chosen according to the level of granularity one needs to reach. For example, if the aim is to check the locality of keywords at the country level, a relatively large cell width is chosen. However, if one is interested in extracting local keywords related to localized events at the city level, a small cell width is considered, e.g., a cell width of 1km or less. Next, the probability distribution for observing keyword k over each cell is calculated. To test its locality, the entropy measure is used [2]. It measures the minimum number of bits needed to represent the geographic spread of a keyword. If a certain keyword is equally distributed over n cells, then the entropy is maximized and has the value $log_2(n)$. However, if it appears in exactly one cell, then the entropy will be 0. Assume that N_k^g refers to the number of observations of keyword k at cell g. Then, the density of this keyword at cell g is $\mathrm{p}_k^g = \frac{N_k^g}{\sum_{g' \in G} N_k^{g'}}$. The entropy of the probability distribution of keyword k over the entire space G is determined as follows:

$$H(\mathrm{p}_k) = -\sum_{g \in G} \mathrm{p}_k^g \times log_2(\mathrm{p}_k^g). \tag{3}$$

Based on this measure, a keyword $k \in K^t$ is considered *local* if and only if $H(\mathrm{p}_k) \leq \phi$. The value of the threshold ϕ is chosen based on the locality level one needs to obtain. The smaller the threshold, the more strict decision is established. For example, when ϕ is set to 0, then only those keywords occurring at exactly one cell are considered local.

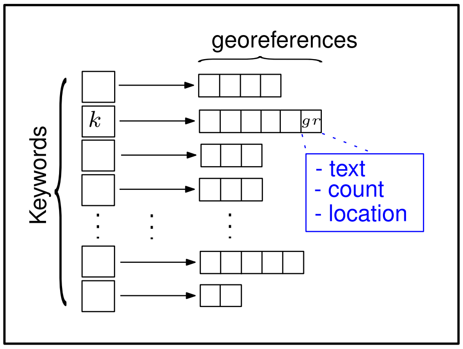


Fig. 2 Keyword-georeference map

5.2 Geo-filter generation

For some popular localized events, people who are not attending such events are stimulated to post relevant tweets. This implies that some related keywords will be mentioned far away from the event location. This type of keyword occurrences is referred to as *spatial outliers*. Spatial outliers contribute to the distortion of the spatial distribution we wish to obtain for local keywords. Thus, they will increase the entropy of the underlying distribution, which might lead to erroneously considering local keywords as non-local. Furthermore, having a localized event occurring in a low-populated cell exaggerates this problem, because in this case the distribution is more sensitive to outliers. Consequently, such a distribution tends to approach faster to the uniform distribution.

Let us assume that a keyword k related to a localized event occurring at cell g suffers from spatial outliers. Our approach to handle this problem is to boost the density p_k^g and to lower the densities at other cells. As a result, this will decrease the entropy $H(\mathrm{p}_k)$, hoping that it reaches the desired locality threshold. For this, we have to answer the following questions: 1) which cells are candidates for the localized event corresponding to keyword k?, and 2) how likely does each candidate cell correspond to that localized event? The answers to these two questions are given in the following two sections.

5.2.1 Keyword-georeference map

We now show how candidate cells of an event corresponding to a certain keyword are identified. For this, we assume that if keyword $k \in K^t$ relates to a localized event, then k tends to co-occur more frequently with georeferences referring to the event location than with other georeferences.

A georeference (place name) gr is a phrase embedded in the content of a tweet and refers to a certain location. For example, if a soccer match takes place at location gr during snapshot t, then there will be more tweets containing keyword “soccer” together with gr than with other georeferences during that time. This is because people tend to mention the location at which they observe or attend an event.

Suppose that the set of georeferences co-occurring with keyword k during t is denoted $\Omega_k = \{gr_1, gr_2, \cdots, gr_v\}$ s.t. each $\Omega_k[gr_i] = (text, count, loc)$ is a tuple reflecting the textual phrase (place name), the number of times gr_i co-occurs with k, and the geo-coordinates (lat, lon) of that georeference, respectively. Thus, Ω denotes the set of keywords published during snapshot t and co-occurring with at least one georeference. The set Ω is referred to as *keyword-georeference map*. Due to the sparsity of this mapping, as only a subset of the extracted keywords map to one or more georeferences, it is realized as an inverted index, as shown in Fig. 2.

Algorithm 1 Building Keyword-Georeference Map

```
   Input: Keywords $K^t$, tweets $\mathcal{W}^t$
   Output: $\Omega$ map
1  $\Omega = \{\}$
   // processing only geo-tagged tweets
2  foreach $tw_i \in \mathcal{W}^t \wedge loc_i \neq \{\}$ do
       // extracting georeferences using OSM data source
3      $gr \leftarrow extractGeoreference(W_i)$
4      if $(gr \neq \{\}) \wedge (dist(gr.loc, loc_i) < \rho)$ then
5          foreach $k \in (W_i - gr.text) \wedge k \in K^t$ do
6              if $(k, gr) \notin \Omega$ then
7                  $\Omega.insert(k, gr)$
8                  $\Omega_k[gr].count = 1$
9              else
10                 $\Omega_k[gr].count \mathrel{+}= 1$
```

Algorithm 1 lists the steps needed to build the map Ω at snapshot t. At Line 2, the tweets posted during t are processed to extract keyword-georeference pairs and to update Ω accordingly. For each tweet, the embedded georeference, if any, is extracted assuming that each tweet contains at most one georeference (Line 2). OpenStreetMap[2] (OSM for short) data is utilized to extract the georeferences. Although there are several Web services allowing for this type of georeference extraction, e.g., Open MapQuest,[3] we chose to materialize OSM data for the space G and index it to provide efficient and offline *geocoding*. Using this OSM-based georeference extraction procedure, we found that about 5.4 % of the tweets have georeferences. Furthermore, a geo-tagged tweet tw_i contributes to build Ω only if tw_i originates within a circular buffer whose center and radius are $gr.location$ and ρ, respectively (Line 4). The parameter ρ is set to 500m to avoid place ambiguity, and the Haversive formula[4] is used to compute the distance between the tweet and the georeference. Then, in Lines 5–10, each keyword k in tweet tw_i (excluding those extracted as a georeference $gr.text$) is mapped to gr, and Ω is updated to include the (k, gr) pair.

5.2.2 *Keyword geo-filter*

In this section, we answer the question of how each georeference contributes to alter the density distribution p_k of keyword k over space G. In other words, for each keyword k, a probability distribution over the georeferences co-occurring with k is to be estimated. For this, we exploit $\Omega_k[gr].count$, the number of times keyword k co-occurs with a certain georeference gr. We define the confidence $\alpha_k[c]$ as the ratio between the number of times

[2] http://www.openstreetmap.org/.

[3] http://open.mapquestapi.com/nominatim.

[4] http://www.movable-type.co.uk/scripts/latlong.html.

keyword k co-occurs with georeferences in cell c and the total number of times k co-occurs with georeferences over the entire space G at snapshot t. More formally, we have

$$\alpha_k[c] := \mathrm{p}(c|k) = \frac{\left(\sum\limits_{gr \in c} \Omega_k[gr].count\right) + 1}{\left(\sum\limits_{c \in G} \sum\limits_{gr \in c} \Omega_k[gr].count\right) + |G|} \tag{4}$$

where Laplacian Smoothing is used to mitigate sparsity and to eliminate zero probabilities. This confidence value, as will be discussed in Section 6, captures how likely the density p_k^c of keyword k at cell c will be affected in order to get the mode of the distribution close to the candidate event cell. As a result, if k belongs to a localized event and co-occurs with relevant georeferences, the entropy will decrease. Finally, the set of confidence measures, denoted α_k, of keyword k over G is referred to as the *geo-filter* of keyword k.

After that, the spatial distribution of k over G is adjusted by multiplying the density p_k^g for each cell $g \in G$ by its corresponding confidence in α_k and then by normalizing by the sum of the updated densities over the entire space, i.e.,

$$\hat{\mathrm{p}}_k^g := \frac{\mathrm{p}_k^g \times \alpha_k^g}{\sum\limits_{g' \in G} \mathrm{p}_k^{g'} \times \alpha_k^{g'}} \qquad \forall g \in G \tag{5}$$

where $\hat{\mathrm{p}}_k^g$ refers to the adjusted version of the spatial density of keyword k at cell g.

6 Local keyword extraction

After adjusting the spatial distribution of keywords as discussed above, the next tasks to be performed towards extracting local keywords are checking the locality of each keyword $k \in K^t$, and then, estimating the spatial focus (central location) of local keywords.

In Section 6.1, we define the concept of spatial focus and list the main steps required to estimate the spatial focus of local keywords. Then, in Section 6.2, we describe the spatial index employed to efficiently extract local keywords. Finally, in Section 6.3, we show how to exploit this index for locality check and focus estimation.

6.1 Spatial focus

Recall that one of the main objectives of our approach is to estimate the spatial focus of local keywords at a fine-grained resolution. This entails exploiting the spatial metadata associated with tweets, namely geo-coordinates, in order to compute the spatial focus of local keywords. The spatial focus $F_k = (lat, lon)$ of keyword k is the geographic midpoint of a set of geo-coordinates associated with the geo-tagged tweets that contain k. The geographic midpoint[5] is favored over the normal mean of points, because it accounts for the spherical shape of the Earth. Although the difference between the focus estimated using the geographic midpoint and that estimated using the normal mean can be neglected in the case of localized events, we choose to apply the geographic midpoint to generalize our detection

[5] http://www.geomidpoint.com/.

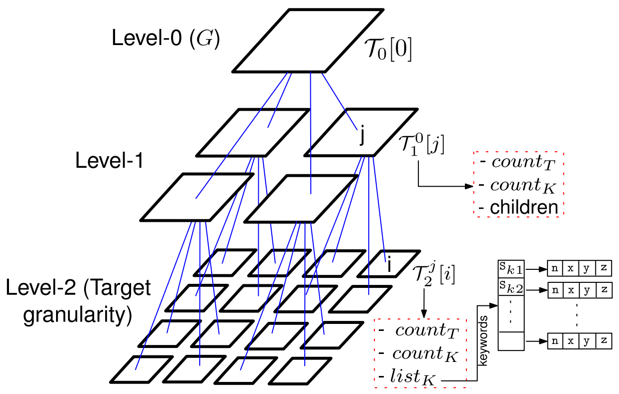


Fig. 3 Dynamic pyramid to spatially index the tweets published during sliding window $\mathcal{W}$. Internal cells hold count statistics required for locality checks, while leaf cells contain, in addition to count statistics, location information needed for focus estimation

approach. That is, when events of a larger spatial extent are to be detected, e.g., events at the country level, a grid of coarse-grained resolution (10km by 10km cells or even larger) is utilized. The computation of the focus of a set of points using the geographic midpoint is done as follows (see Section 6.2 for more details):

1. the lat/lon coordinates of each point is converted into Cartesian (x, y, z) coordinates, then
2. the mean of these coordinates is computed to get the midpoint in the Cartesian 3D space, and finally,
3. this midpoint is converted back to lat/lon coordinates.

Checking the locality of each keyword $k \in K^t$ and computing the spatial focus of local ones are computationally-expensive processes, especially when using a fine-grained spatial resolution and a relatively large widow size. To cope with this, in the following section, we present a space-partitioning index structure where count and location information is maintained at multiple levels of a hierarchy, providing efficient and incremental updates of such information.

6.2 Hierarchical space-partitioning index

In order to efficiently process tweets in the context of high-arrival rates, we employ a hierarchical space-partitioning index structure [23]. This type of spatial indexing recursively partitions the space into non-overlapping cells, starting at the root cell until a certain capacity threshold or a target resolution is reached. These partitions are organized in a hierarchical structure with the root cell referring to the entire space G and the last level referring to the finest (target) resolution as illustrated in Fig. 3. Quadtrees and pyramid index structures are typical examples for hierarchical indexing [26].

In this work, we adopt the pyramid index, because it allows for maintaining statistics (synopses), e.g., keyword counts, in internal nodes, which summarize the information stored in the leaf nodes of such an index. Maintaining synopses in internal nodes, as will be discussed later, helps expedite the process of updating the index structure, deciding about the locality of keywords, and estimating their spatial focus.

6.2.1 *Dynamic pyramid structure*

The pyramid index structure is viewed as a tree $\mathcal{T}$ of height h. The nodes of the tree are divided into levels $(0, 1, \cdots, h-1)$ where level 0 and $h-1$ hold the root and the leaves, respectively. Each level corresponds to a regular grid partitioning the space into non-overlapping cells, such that each node in that level is mapped to exactly one geographic cell. The higher the level, the finer the spatial granularity used to partition space G. Let us denote the node j at level i in $\mathcal{T}$ as $\mathcal{T}_i^p[j]$, where p refers to the parent node at level $i-1$. The children of the node $\mathcal{T}_i[j]$ are denoted by $\mathcal{T}_{i+1}^j$ and represent the cells covered by the bounding box of the parent cell $\mathcal{T}_i[j]$. The root node $\mathcal{T}_0[0]$ corresponds to a bounding box covering the space G, and the leaves $\mathcal{T}_{h-1}$ reflect the finest spatial resolution to be targeted.

As can be seen in Fig. 3, each internal node $\mathcal{T}[j]$ in the tree contains:

1. $count_T$: the number of tweets published from $\mathcal{T}[j]$ during the sliding window $\mathcal{W}$, which is used to both estimate the spatial signal at $\mathcal{T}[j]$ and accelerate updating the tree.
2. $count_k$: the number of tweets containing a keyword and originating from within $\mathcal{T}[j]$ during $\mathcal{W}$. $count_k$ is maintained to check the locality of keywords. There is only one instance of $count_k$ at each cell, and therefore, once a new keyword is examined, the value of $count_k$ is reset and initialized with respective count statistics accumulated from leaf cells.

Each leaf cell contains, in addition to the previous statistics, a list $list_K = (\mathtt{S}_{k_1}, \mathtt{S}_{k_2}, \cdots)$ containing synopses required to check the locality of keywords and compute their spatial focus. Each $\mathtt{S}_k = (\mathtt{n}, \mathtt{x}, \mathtt{y}, \mathtt{z})$ maintained in the leaf node $\mathcal{T}_{h-1}[i]$ has the following components: (a) $\mathtt{S}_k.\mathtt{n}$: the number of tweets originating from $\mathcal{T}_{h-1}[i]$ and containing keyword k, and (b) the values $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$ as the sum of the Cartesian coordinates of each tweet containing k and mapped to that cell. These synopses are sufficient to estimate the spatial focus of keywords. Next, we will show how to update their values as new tweets are inserted at snapshot t and others are eliminated when their snapshots expire.

6.2.2 *Spatial index update*

Updating the tree $\mathcal{T}$ is essential to ensure an up-to-date state of the maintained synopses. The update procedure is triggered once a new snapshot t elapses and involves the *insertion* and the *deletion* of the tweets published during t and $t-c$, respectively.

Insertion For each tweet m_i containing at least one keyword, i.e., $m_i \in \mathcal{W}_k^t \wedge k \in K^t$, the tree is traversed from the root to the corresponding leaf node based on the geo-coordinates $m_i.loc_i$. The value $count_T$ of each traversed node is incremented by 1. When the leaf node is reached, $\mathtt{S}_k.\mathtt{n}$ is incremented by 1 and the remaining elements $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$ are updated as follows

$$\mathtt{S}_k.\mathtt{x} \mathrel{+}= \cos\left(loc_i.lat \times \frac{\pi}{180}\right) \times \cos\left(loc_i.lon \times \frac{\pi}{180}\right) \tag{6}$$

$$\mathtt{S}_k.\mathtt{y} \mathrel{+}= \sin\left(loc_i.lat \times \frac{\pi}{180}\right) \times \cos\left(loc_i.lon \times \frac{\pi}{180}\right) \tag{7}$$

$$\mathtt{S}_k.\mathtt{z} \mathrel{+}= \sin\left(loc_i.lon \times \frac{\pi}{180}\right) \tag{8}$$

These equations convert the lat/lon coordinates of tweet m_i into the Cartesian coordinate system and accumulate the results into their respective elements $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$.

However, tweets pertaining to a certain event might contain more than one keyword. Hence, the tree will be traversed more than once to insert the keywords of one tweet. To overcome this, we build a temporary inverted index having the tweet ids as keys and the keywords as postings, which helps to reduce the insertion time.

Deletion Similarly, the tweets $\{\mathcal{W}_k^{t-c}\}_{\forall k \in K^t}$ of the expired snapshot $t - c$, which contain at least one keyword, are used to traverse the tree and to remove the corresponding statistics. This time, however, the statistics of these tweets are subtracted from the maintained synopses. For the same efficiency reason, a temporary inverted index is created so that the tree is traversed once for each tweet. To improve the efficiency of the deletion process, each time a new node is visited, $count_T$ is decremented by 1. If it becomes 0, the current node and all its children are removed from the tree.

6.3 Focus estimation

In this section, we use the pyramid structure along with the maintained synopses in order to incrementally identify local keywords K_{local}^t and to estimate the focus F_k of each $k \in K_{local}^t$. Algorithm 1 details the steps needed to accomplish these tasks for each $k \in K^t$. These steps can be categorized into: (1) count initialization (Line 3), (2) count update and locality check (Lines 4–21), and (3) focus estimation (Lines 22–24).

Algorithm 2 Locality Check and Focus Estimation

```
    **Input**: Keywords $K^t$ , tree $\mathcal{T}$, entropy threshold $\phi$
    **Output**: Local keywords $K_{local}^t$, focus $F_k$ for each $k \in K_{local}^t$
1   $K_{local}^t \leftarrow \{\}$
2   **foreach** $k \in K^t$ **do**
3       $initializeCounts(k)$
        // index of root node
4       $n \leftarrow 0$
5       $isLocal \leftarrow true$
6       **for** $l = 1$ *to* $h - 1$ **do**
7           $sum \leftarrow 0$
8           **foreach** $g \in \mathcal{T}_l^n$ **do**
9               $\text{p}_k^g := \frac{g.count_k}{g.count_T}$ // spatial density estimation
10              $sum \leftarrow sum + \text{p}_k^g$
11          $\text{p}_k \leftarrow \text{p}_k / sum$
12          $\hat{\text{p}}_k \leftarrow spatialAdjustment(\text{p}_k)$ // see Section 5.2
13          $ent \leftarrow 0$
14          **foreach** $(g \in \mathcal{T}_l^n) \wedge (\hat{\text{p}}_k^g > 0)$ **do**
15              $ent \mathrel{+}= \hat{\text{p}}_k^g \times \log_2(\hat{\text{p}}_k^g)$
            // the entropy of keyword $k$ at level $l$
16          $entropy \leftarrow -1 \times ent$
17          **if** $entropy \leq \phi$ **then**
18              $n \leftarrow \arg\max_{g' \in \mathcal{T}_l^n} (\hat{\text{p}}_k^{g'})$
19          **else**
                // in case keyword $k$ is identified as non-local
20              $isLocal \leftarrow false$
21              $break$
22      **if** $isLocal == true$ **then**
            // if keyword $k$ is local, it will be added to $K_{local}^t$
23          $K_{local}^t \leftarrow K_{local}^t \cup k$
24          $F_k \leftarrow geomid(k, \mathcal{T}_{h-1}[n])$
```

Count initialization Recall that each node in the tree maintains the field $count_k$ to store the number of tweets containing keyword k. In this step, a value is assigned to $count_k$ of each node in $\mathcal{T}$. For leaf nodes, these counts can simply be initialized as $count_k = \mathtt{S}_k.\mathtt{n}$. Then, the tree $\mathcal{T}$ is traversed recursively to accumulate these counts and propagate the sums up the tree hierarchy until the root node is reached. We end up having the field $count_k$ of each internal node holding the sum of the corresponding fields of its children, i.e.,

$$\mathcal{T}_i[j].count_k = \sum_{g \in \mathcal{T}_{i+1}^{j}} g.count_k \qquad \forall i \in \{0, \ldots, h-2\} \tag{9}$$

Count update and locality check In Line 4, n refers to the index of the cell that is assumed to be a candidate location for the localized event described by keyword k. The index n is initialized to 0 referring to the root at level 0. Then, the tree is traversed top-down and level-wise starting from the coarsest level (i.e., level 1) to the finest one (level $h-1$) to locate the leaf node (cell) with the highest density in case keyword k is identified as local. In order to consider keyword k a local keyword, it should pass the locality check at each level. Before checking the locality of k at level l using the entropy-based method discussed in Section 5, the spatial distribution of keyword k is estimated, normalized, and adjusted, as illustrated in Lines 7–12. Then, the entropy of the adjusted distribution over the entire set of cells $\mathcal{T}_l^n$ is computed (Lines 13–16). If it falls in the range $[0, \phi]$, then keyword k is considered local at level l and the index of the cell having the highest density is returned and stored in n. This will prune the spatial space, because only the tree branch of this cell will be handled in the next iteration (at level $l+1$).

Focus estimation Once keyword k is recognized as local, it will be added to K_{local}^t (Line 23), and the focus of k (Line 24) is computed as follows

$$F_k.lat = atan2\left(\frac{\mathtt{S}_k.\mathtt{z}}{\mathtt{S}_k.\mathtt{n}}, \sqrt{\left(\frac{\mathtt{S}_k.\mathtt{x}}{\mathtt{S}_k.\mathtt{n}}\right)^2 + \left(\frac{\mathtt{S}_k.\mathtt{y}}{\mathtt{S}_k.\mathtt{n}}\right)^2}\right) \times \frac{180}{\pi} \tag{10}$$

$$F_k.lon = atan2\left(\frac{\mathtt{S}_k.\mathtt{y}}{\mathtt{S}_k.\mathtt{n}}, \frac{\mathtt{S}_k.\mathtt{x}}{\mathtt{S}_k.\mathtt{n}}\right) \times \frac{180}{\pi} \tag{11}$$

where the middle spatial point of the Cartesian coordinates accumulated in $(\mathtt{S}_k.\mathtt{x}, \mathtt{S}_k.\mathtt{y}, \mathtt{S}_k.\mathtt{z})$ is computed and converted back into lat/lon coordinates. At snapshot t, each local keyword $k \in K_{local}^t$, along with its focus F_k, is used to form an *event entity* (entity for short), denoted by $ee = (k, F_k, t)$. These entities are appended to a queue called *Event Queue* ($EQueue$) that holds entities generated during the sliding window $\mathcal{W}$.

One last issue we consider is when multiple events on the same topic are taking place at the same time over G. In this case, some shared keywords, referred to as *topical keywords*, are distributed broadly over space, and hence, are mistakenly identified as non-local. For example, when two soccer matches occur simultaneously, topical keywords like “match” and “goal” will be mentioned at the location of both events. To recover the locality of such keywords, we exploit the spatial co-occurrence between topical and local keywords K_{local}^t. The intuition here is that a keyword identified as non-local might pertain to one or more localized events if it shows a certain level of co-location with at least one of the events' local keywords. For this, we employ the Pearson correlation coefficient as detailed in Algorithm 3.

Algorithm 3 Topical-to-local keyword recovery procedure.

```
Input: Keywords $K^t$, local keywords $K^t_{local}$, spatial distributions $\hat{p}_k$ $\forall k \in K^t$, and a
       threshold $\gamma$.
Output: Updated $EQueue$
1  $T \leftarrow K^t - K^t_{local}$ // set of topical keywords
2  $cells \leftarrow \{\}$
   // only considering geo-tagged tweets
3  foreach $tk \in T$ do
4      foreach $lk \in K^t_{local}$ do
5          if $cells \cap cellOf(F_{lk}) == \{\}$ then
6              if $\rho_{tk,lk} \geq \gamma$ then
7                  $ee \leftarrow (tk, F_{lk}, t)$
8                  $EQueue.append(ee)$
9                  $cells \leftarrow cells \cup cellOf(F_{lk})$
10     $cells \leftarrow \{\}$
```

In Line 1, the notation T refers to the set of all possible topical keywords. They are traversed to extract those that can be recovered as local keywords (Lines 3–10). The correlation between the two probability distributing $\hat{p}_{tk}$ and $\hat{p}_{lk}$ is measured using the Pearson correlation coefficient, defined as

$$\rho_{tk,lk} := \frac{\text{cov}(\hat{p}_{tk}, \hat{p}_{lk})}{\sigma_{\hat{p}_{tk}} \times \sigma_{\hat{p}_{lk}}}, \tag{12}$$

where $\text{cov}(\hat{p}_{tk}, \hat{p}_{lk})$ is the covariance that is estimated as $E[(\hat{p}_{tk} - \mu_{\hat{p}_{tk}})(\hat{p}_{tl} - \mu_{\hat{p}_{tl}})]$ and $\sigma_{\hat{p}_{tk}}, \sigma_{\hat{p}_{lk}}$ are the standard deviation of $\hat{p}_{tk}$ and $\hat{p}_{lk}$, respectively. The correlation coefficient yields a value in the range $[-1, 1]$, where 1 implies a perfect positive correlation and -1 means a perfect negative correlation. If the correlation between the distributions of the topical keyword tk and the local one lk is above or equal to a certain threshold γ, tk is considered local, and thus, a new event entity ee is generated and appended to $EQueue$ (Lines 5–8). The set $cells$ holds the geographic cells that the topical keyword has been assigned to so far, which prevents generating multiple entities that have a focus in same cell. This also significantly improves efficiency as only a subset of the local keywords are compared against the topical keyword tk.

7 Experimental evaluation

In this section, we test the performance of our framework in reliably and efficiently extracting local keywords. First, in Section 7.1, we discuss the dataset and platform used for the evaluation. Then, the identification and focus estimation tasks are evaluated in Sections 7.2 and 7.3, respectively.

7.1 Experimental setup

Dataset A Twitter dataset of about 5.5 million tweets published from densely-populated regions of New York and New Jersey is utilized to evaluate our framework. The tweets are

Table 2 Localized events occurred in NY during Nov, 2013. They are used as case studies for evaluation purposes

ID	Event description	Keywords
$E1$	The Victoria's Secret Fashion Show	fashion, secret, victoria, vsfashionshow, victoriassecret
$E2$	Argentina vs. Equador	argentina, equador, argentinavsecuador
$E3$	New York Giants vs. packers	giants, packers, giantsnation
$E4$	Krewella with Gareth Emery	krewella, gareth, emery, pieroffear
$E5$	Dallas Cowboys vs. New York Giants	giants, dallas, giantsnation, dallascowboys, hakeem, cruz, andre, victor
$E6$	Carolina Hurricanes vs. New York Rangers	hurricanes, carolina, carolinahurricanes, rangers

crawled using a Twitter API[6] and are filtered by their geo-coordinates, where the bounding box [lat:40.50, lon:-74.63, lat:40.97, lon:-73.63], covering an area of ca. 4418.58 km^2, is passed on to the crawling API. The timestamps of the collected tweets fall within the time interval (2013/10/20 00:00 - 2013/11/30 23:59). The first 12 days are reserved as a history from which the baseline parameters (see Section 4) are estimated. Based on the findings by Morstatter et al. [19], this location-based filtering allows for retrieving almost a complete sample of geo-tagged tweets originating from within the specified bounding box. The tweets are ordered by their timestamps and sent to the framework as a stream of geo-tagged messages, imitating the streaming nature of Twitter. To cope with the noisy Twitter content, we eliminated special characters (e.g., #, ?, !, "), stop words, words of length less than 3, and URLs. Moreover, duplicate tweets published from one user at a certain snapshot are combined into one tweet. This is essential to cope with tweets that are republished frequently as advertisements, news distribution, spam etc.

To perform a focused evaluation based on domain knowledge, we utilized several Web sources to find some featured localized events in the dataset as study cases for Sections 7.2 and 7.3. Table 2 lists these events along with some related keywords to be considered in the experiments.

Platform The experimental evaluation is performed using a platform having an Intel Core i7 (3.40 GHz) and 16 GB main memory. We ran our framework under Ubuntu 12.04 (with Java 7 framework).

7.2 Keyword locality evaluation

In this section, we test the performance of our framework in providing a reliable decision about the locality of keywords and in exhibiting a fast response for incoming tweets that might change the locality behavior of the contained keywords. We set the cell width of the used grid to 1.1 km. The locality threshold ϕ for this and the following experiments is set as follows: we plot the percentage of keywords recognized as local over a number of locality thresholds. As can be seen in Fig. 4, a significant change (about 17 %) in the percentage of local keywords is observed when varying ϕ from 0.5 to 1.0, and thus, we decided to assign the value 0.5 to ϕ as a cutoff locality threshold.

[6]http://dev.twitter.com/pages/streaming_api.

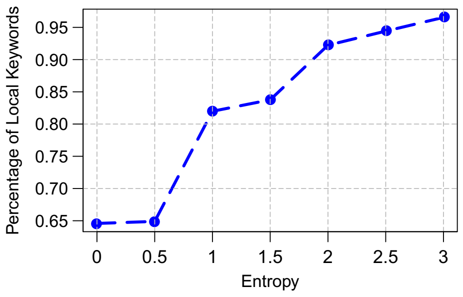


Fig. 4 The impact of ϕ on local keywords

We selected a number of events from Table 2 and collected statistics pertaining to the considered keywords before and after dealing with spatial outliers. In Table 3, the second and third column show how the entropy mean of extracted keywords decreases drastically after eliminating spatial outliers, which, of course, increases the chance of identifying them as local keywords. The forth column shows the percentage of *spatially-adjusted keywords* that become local and have a focus at the event location after the adjustment. The more important and popular the localized event, the higher is the percentage. The fifth column illustrates the precision of the adjusted keywords, i.e., these keywords that actually refer to the corresponding event. For example, the total number of keywords that have a focus at the location of event $E1$ after spatial outlier elimination is 40 where 34 of them are actually pertaining to $E1$, yielding a precision of 0.85 as illustrated in Table 3. Consequently, the majority of the adjusted keywords that have a focus at the event location relate to that event.

In Fig. 5, the keyword "krewella" of event $E4$ is considered and the temporal profile of the keyword's entropies during 10 1-hour snapshots is plotted. We notice that the entropy values increase over time due to the fact that more and more people will be aware of the occurrence of the event by time through TV, friends etc. After the elimination of outliers (discussed in Section 5.2), these values become smaller (below ϕ most of the time), and thus, the impact of these outliers on the spatial distribution of keyword is minimized.

Furthermore, we evaluate another important aspect of our approach, which is how much information is needed to handle spatial outliers and regain the expected spatial distribution of keywords. In other words, the aim is to be able to report the locality of a keyword using just a few tweets. For this, a number of tweets are injected into the dataset at snapshot (11/13 03:00), having the geo-coordinates (lon: -73.983, lat: 40.738) of an assumed event location. These tweets contain the words: "fire", "shooting", and the georeference "pizza pub" that is close to the event location. In Fig. 6, we plot two diagrams for both keywords, and show how the entropy decreases as the number of such tweets increases. To highlight the value added by our approach that performs Spatial Outlier Elimination (denoted as `SOE`), we compare its performance against a baseline that does not perform Spatial Outlier elimination, denoted as `SO`. In Fig. 6a, it is observed that 120 tweets are required to make the keyword "fire" local

Table 3 Performance of the framework before and after eliminating spatial outliers

EID	(before)	(after)	Spatially-adjusted	Precision
$E1$	1.345	0.807	33.6 %	$\frac{34}{40} = 0.85$
$E2$	0.478	0.027	41.4 %	$\frac{41}{56} = 0.732$
$E3$	0.803	0.1603	56.2 %	$\frac{70}{104} = 0.67$
$E4$	0.49	0.028	39.6 %	$\frac{16}{20} = 0.8$

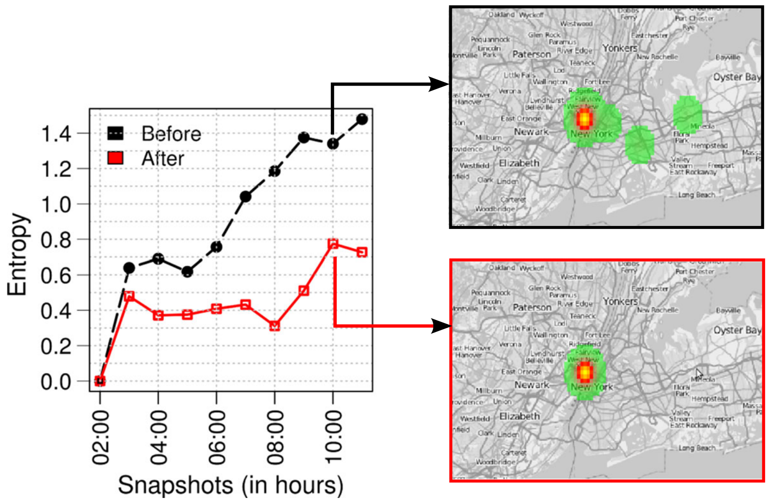


Fig. 5 The impact of outlier elimination for the keyword "krewella" of event $E4$ during 10 1-hour snapshots

using SO. This is because this word is usually mentioned in different contexts, and thus, it has a broad spatial coverage. However, SOE entails only 8 tweets to reach the locality threshold ϕ. On the other hand, and as can be seen in Fig. 6b, the keyword "shooting" has reached ϕ only with 3 tweets using SOE since it is sparsely used over space.

In addition, we compare SOE against the model of Backstrom et al. [7], which we denote as SV. SV is a probabilistic model that captures the spatial variation of geo-tagged search queries and gives an estimation of the central location of a query and its spatial dispersion. The dispersion indicates whether the query has a local interest or broader regional or national appeal. It is quantified by estimating a value for the dispersion parameter α. The larger α, the faster the query decays away from the center and the more localized the query

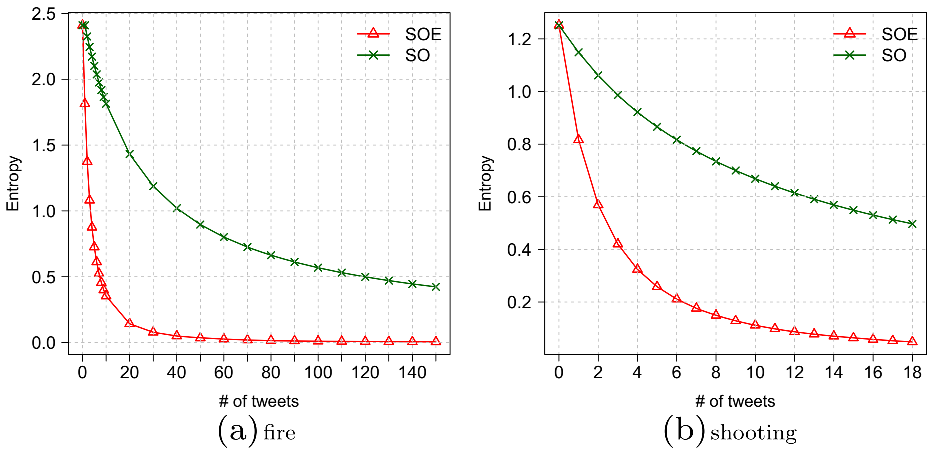


Fig. 6 Impact of increasing the number of a keyword's occurrences on entropy using the methods SOE and SO

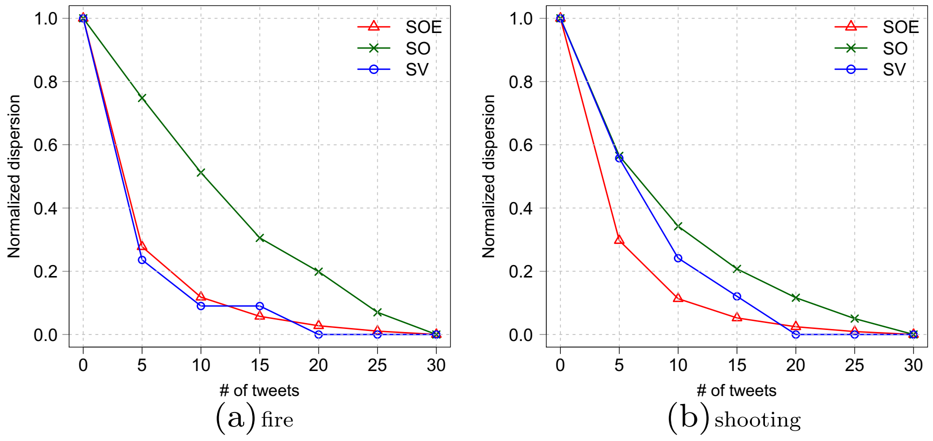


Fig. 7 Effect of number of tweets on normalized dispersion measures from `SOE`, `SO`, and `SV`

is. In this context, we consider the preprocessed content of a tweet as a query in order to be able to utilize `SV` and compute the dispersion of a particular keyword. Figure 7 shows the effects of varying the number of injected tweets from 0 to 30 on a normalized versions of $-\alpha$, the entropy from `SOE`, and the entropy from `SO`. In Fig. 7a, both `SOE` and `SV` exhibit a similar and fast response to the increase in the number of tweets containing the keyword “fire”. Both approaches outperform `SO`, because `SO` relies only on a word’s spatial distribution whose entropy decreases slowly in case of widely-used keywords such as “fire”. Figure 7b shows that `SO` has a better response to the increase in tweets containing the keyword “shooting” due to its sparse spatio-temporal usage. However, `SOE` outperforms both `SO` and `SV`, which highlights its effectiveness in providing a rapid locality detection for a sparsely-used keyword.

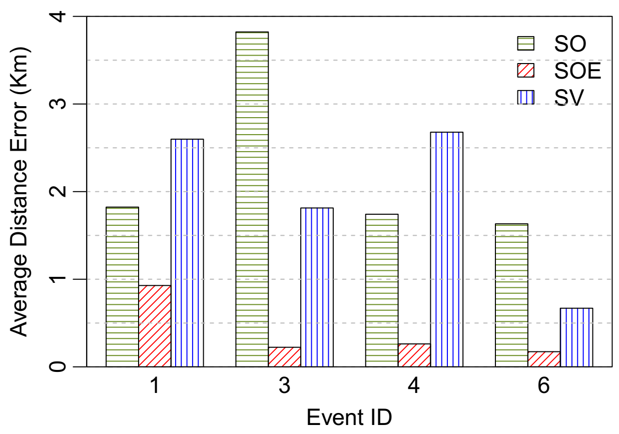


Fig. 8 Average error distance for a number of events

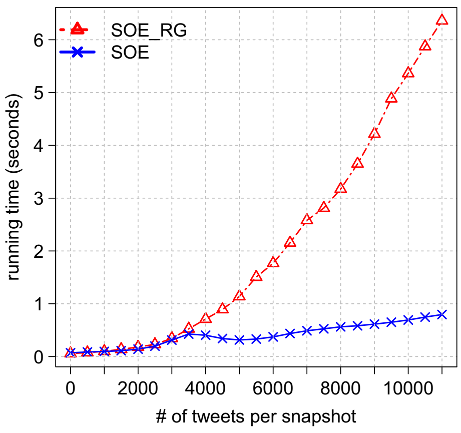


Fig. 9 Running time (in seconds) for different numbers of tweets using our approach (SOE) and a baseline variation (SOE_RG) without spatial indexing

7.3 Evaluation of focus estimation

In this section, we compare the accuracy of SOE in computing the spatial focus of local keywords against SO and SV. For this, we chose some events from Table 2, computed the focus of each keyword using the three approaches, and plotted the average error distance (see Fig. 8).

The focus of a keyword k obtained from the baseline approach SO is estimated from the tweets containing k by computing the weighted center of gravity of their geo-coordinates. To calculate the error distance for a particular keyword, we manually specify a location for each localized event as the central geographic point of the event venue, and then apply the Haversine formula to determine the distance between the keyword focus and the event's central point. Figure 8 shows that SOE outperforms the other approaches in estimating an accurate focus for all events. This is explained by the fact that SOE not only eliminates outliers, but also preserves a location summary for each keyword inside each cell. The approaches SO and SV exhibit an almost similar behavior, and, of course, spatial outliers lead to relatively large error distances.

After running our framework over the entire dataset, 4.37 minutes are needed to process 3.8 million tweets. This time includes the preprocessing of tweets, keyword extraction, index update, locality check, and focus estimation. Hence, on average, about 0.069 ms is required to process each tweet. In Fig. 9, we compare the SOE approach with SOE_RG, a baseline version without spatial indexing. SOE outperforms SOE_RG due to its ability to perform fast pruning of the geographic space until the cell with the summarized location information is reached.

Another aspect that supports the scalability of SOE is that it can easily be parallelized. Fortunately, each word is processed separately, which makes it easy to split the set of words $\mathcal{W}^t$ published during snapshot t among multiple processing threads. Each thread processes a subset of these words. Then, the generated entities from all threads are appended to a synchronized version of the event queue $EQueue$.

8 Conclusions and ongoing work

In this paper, we presented a framework to extract local keywords, i.e., bursty words that have a limited spatial extent, from a stream of Twitter messages and estimate their spatial focus. Such keywords are essential in building systems that support the online detection of localized events. The framework incrementally consume incoming tweets published during a time window and aggregates their location information to obtain synopses that are sufficient to check the locality of keywords and to estimate their geographic focus. Moreover, the framework handles the undesirable spatial behavior of a local keyword, e.g., when people mention it far away from the event location, a problem we call spatial outliers. It adversely affects both a proper computation of the locality of a keyword and an accurate estimation of its focus. On the other hand, we dealt with cases when a keyword is mentioned in tweets pertaining to several events, namely, events having the same type, e.g., two soccer matches.

As an ongoing work, we experiment with a density-based clustering algorithm to form potential event clusters, each of which corresponds to one real-world event. Due to the high levels of noise in Twitter content, one expects a large number of generated clusters. For this, the next step is to use a machine learning solution in order to distinguish actual event clusters from noisy ones.

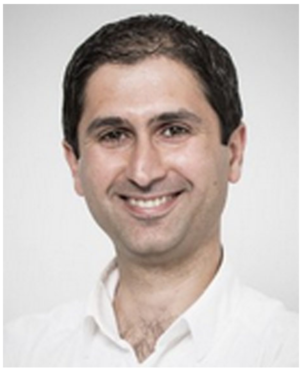
References

1. Abdelhaq H, Gertz M (2014) On the locality of keywords in Twitter streams. In: IWGS ’14, pp 12–20
2. Abdelhaq H, Gertz M, Sengstock C (2013) Spatio-temporal characteristics of bursty words in Twitter streams. In: SIGSPATIAL ’13, pp 149–158
3. Abdelhaq H, Sengstock C, Gertz M (2013) EvenTweet: online localized event detection from Twitter. Proc VLDB Endow 6(12):1326–1329
4. Aggarwal CC, Subbian K (2012) Event detection in social streams. In: SDM ’12, pp 624–635
5. Allan J (ed) (2002) Topic detection and tracking: event-based information organization. Kluwer Academic Publishers, Norwell
6. Alvanaki F, Michel S, Ramamritham K, Weikum G (2012) See what’s enBlogue: real-time emergent topic identification in social media. In: EDBT ’12, pp 336–347
7. Backstrom L, Kleinberg J, Kumar R, Novak J (2008) Spatial variation in search engine queries. In: WWW ’08, pp 357–366
8. Becker H, Naaman M, Gravano L (2011) Beyond trending topics: real-world event identification on Twitter. In: ICWSM ’11
9. Boettcher A, Lee D (2012) EventRadar: a real-time local event detection scheme using Twitter stream. In: GreenCom ’12, pp 358–367
10. Cataldi M, Di Caro L, Schifanella C (2010) Emerging topic detection on Twitter based on temporal and social terms evaluation. In: MDMKDD ’10, pp 4:1–4:10
11. Chen L, Roy A (2009) Event detection from Flickr data through wavelet-based spatial analysis. In: CIKM ’09, pp 523–532
12. Chunara R, Andrews JR, Brownstein JS (2012) Social and news media enable estimation of epidemiological patterns early in the 2010 Haitian cholera outbreak. Am J Trop Med Hyg 86(1):39–45
13. Kleinberg J (2003) Bursty and hierarchical structure in streams. Data Min Knowl Discov 4:373–397
14. Lappas T, Arai B, Platakis M, Kotsakos D, Gunopulos D (2009) On burstiness-aware search for document sequences. In: KDD ’09, pp 477–486
15. Lappas T, Vieira MR, Gunopulos D, Tsotras VJ (2012) On the spatiotemporal burstiness of terms. PVLDB 5(9):836–847

16. Lee CH, Wu CH, Chien TF (2011) BursT: a dynamic term weighting scheme for mining microblogging messages. In: ISNN ’11, pp 548–557
17. Li C, Sun A, Datta A (2012) Twevent: segment-based event detection from tweets. In: CIKM ’12, pp 155–164
18. Magdy A, Mokbel MF, Elnikety S, Nath S, He Y (2014) Mercury: A memory-constrained spatio-temporal real-time search on microblogs. In: ICDE ’14, pp 172–183
19. Morstatter F, Pfeffer J, Liu H, Carley KM (2013) Is the sample good enough? comparing data from Twitter’s streaming API with Twitter’s firehose. In: ICWSM ’13
20. Petrović S, Osborne M, Lavrenko V (2010) Streaming first story detection with application to Twitter. In: HLT ’10, pp 181–189
21. Rattenbury T, Good N, Naaman M (2007) Towards automatic extraction of event and place semantics from Flickr tags. In: SIGIR ’07, pp 103–110
22. Sakaki T, Okazaki M, Matsuo Y (2010) Earthquake shakes Twitter users: real-time event detection by social sensors. In: WWW ’10, pp 851–860
23. Samet H (1990) Applications of spatial data structures: computer graphics, image processing and GIS. Addison-Wesley
24. Sankaranarayanan J, Samet H, Teitler BE, Lieberman MD, Sperling J (2009) TwitterStand: news in tweets. In: GIS ’09, pp 42–51
25. Skovsgaard A, Sidlauskas D, Jensen C (2014) Scalable top-k spatio-temporal term querying. In: ICDE ’14, pp 148–159
26. Tanimoto S, Pavlidis T (1975) A hierarchical data structure for picture processing. Comput Vision Graph 4(2):104–119
27. Valkanas G, Gunopulos D (2013) How the live web feels about events. In: CIKM ’13, pp 639–648
28. Vlachos M, Meek C, Vagena Z, Gunopulos D (2004) Identifying similarities, periodicities and bursts for online search queries. In: SIGMOD ’04, pp 131–142
29. Watanabe K, Ochi M, Okabe M, Onai R (2011) Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In: CIKM ’11, pp 2541–2544
30. Weng J, Lee BS (2011) Event detection in Twitter. In: ICWSM ’11
31. Yang Y, Pierce T, Carbonell J (1998) A study of retrospective and on-line event detection. In: SIGIR ’98, pp 28–36
32. Zhou X, Chen L (2014) Event detection over Twitter social media streams. VLDB J 23(3):381–400

Hamed Abdelhaq is a senior data analyst at moovel group GmbH, Stuttgart, Germany. He received his Bachelor degree from the University of Jordan in 2005 and his MSc degree in Computer Science from the same university in 2007. During his Master study, he worked as a Web developer at Maktoob Group, Amman, Jordan. Then, he worked for about 3 years as a lecturer in the Computer Science department at An-Najah National University, Palestine. Mr. Abdelhaq was awarded a DAAD scholarship to complete his PhD at the Database Systems Research group at Heidelberg University. In 2016, he received his PhD degree in Computer Science from Heidelberg University. His research interests include spatio-temporal data analysis, data mining, and event extraction from social media.

Michael Gertz is a full professor at Heidelberg University where he heads the database systems group at the faculty of Mathematics and Computer Science. He received his diploma in Computer Science from the University of Dortmund, Germany, and his Ph.D. from the University of Hanover, Germany, in 1996. From 1997 until 2008 he was a faculty at the University of California at Davis. He currently serves on the editorial board of the ACM Transactions on Spatial Algorithms and Systems, and he is an associate editor of the ACM Journal on Data and Information Quality. His research interests include the management and analysis of temporal, spatial, and spatio-temporal data, data mining, text mining and social network analysis.



Ayser Armiti is the head of data science at moovel Group GmbH, Stuttgart, Germany. He got his BSc in computer science from the Arab American University-Jenin. In 2008, he received his MSc in computer science from the University of Jordan. Between 2008 and 2010, Mr. Armiti worked as a data integration engineer at PDF Solutions, San Jose, CA. After that, he joined the database research group at Heidelberg University as a PhD student and research assistant where in 2015 he got the PhD degree under the supervision of Prof. Michael Gertz. Mr. Armiti's research interests include spatio-temporal pattern extraction and graph similarity.