Monitoring Spatial Coverage of Trending Topics in Twitter

Kostas Patroumpas[§,†]
[§]Institute for the Management of Information Systems
Athena Research Center, Hellas
kpatro@dblab.ece.ntua.gr

Manolis Loukadakis[†]
[†]School of Electrical and Computer Engineering
National Technical University of Athens, Hellas
manos.loyk@gmail.com

ABSTRACT

Most messages posted in Twitter usually discuss an ongoing event, triggering a series of tweets that together may constitute a trending topic (e.g., *#election2012, #jesuischarlie, #oscars2016*). Sometimes, such a topic may be trending only locally, assuming that related posts have a geographical reference, either directly geotagging them with exact coordinates or indirectly by mentioning a well-known landmark (e.g., *#bataclan*). In this paper, we study how trending topics evolve both in space and time, by monitoring the Twitter stream and detecting online the varying spatial coverage of related geotagged posts across time. Observing the evolving spatial coverage of such posts may reveal the intensity of a phenomenon and its impact on local communities, and can further assist in improving user awareness on facts and situations with strong local footprint. We propose a technique that can maintain trending topics and readily recognize their locality by subdividing the area of interest into elementary cells. Thus, instead of costly spatial clustering of incoming messages by topic, we can approximately, but almost instantly, identify such areas of coverage as groups of contiguous cells, as well as their mutability with time. We conducted a comprehensive empirical study to evaluate the performance of the proposed methodology, as well as the quality of detected areas of coverage. Results confirm that our technique can efficiently cope with scalable volumes of messages, offering incremental response in real-time regarding coverage updates for trending topics.

CCS Concepts

•Information systems → Data streaming; Spatial-temporal systems; Location based services;

Keywords

geostreaming; similarity; spatial coverage; trending topics; Twitter.

1. INTRODUCTION

Over the past decade, rapid proliferation of microblogging and social networking has emerged as a global phenomenon with unprecedented political, economic, and societal impact. For instance, in Facebook[1] users can exchange messages, post photos or videos, get notifications about the status of their friends, etc. In Foursquare[2], users may share their check-ins, i.e., their visits to places like bars, cinemas, or sporting venues. Each day, Flickr[3] receives millions of new photos and videos for hosting. And in Twitter[4], hundreds of millions of people are posting each day more than 500 million messages of max 140 characters, called *'tweets'*. This user-generated content offers a variety of information (textual, pictorial, temporal, spatial, hyperlinks, user interconnections, etc.) and gets updated continuously in a streaming fashion. Such a rich data source has obviously become a treasure trove for researchers and practitioners alike, as it can be used to detect emerging events [23], analyze ongoing trends [5, 11, 14], identify user communities [13], etc.

One important aspect of such user-generated content is that it may also include *geographical* references. Such georeferences can be provided either directly by specifying the current coordinates of the user through her mobile device, or indirectly via a well-known place that is stated in the message (e.g., *#bataclan*) and can be uniquely identified in the real world. If known, geographic location can offer tremendous benefits when analyzing and mining this data [11, 16]. It can be used for localized detection of events [2, 14], local news extraction [19], getting recommendations from local users [13], locally-targeted marketing and advertising of products, and much more. Hence, identifying geographical locality among such massive, streaming messages is valuable.

In this work, we turn our focus on detecting ongoing discussions in social media that have a strong *local footprint*. For example, such discussions are usually marked in Twitter with one or more *hashtags* to enable searching, e.g., *#ParisAttacks, #PrayForParis*. But many of the users that participate in a discussion may often be located in a certain area, which can even change in shape and size while this discussion is evolving. For instance, tweets concerning a wildfire may initially come from observers near its origin. Later, as the fire may be spreading, more users around may join in and post updates. Interest may drop once the fire is brought under control, and when eventually it is put out, this topic dissolves. Similar situations usually reflect issues with a *limited spatial coverage* that mostly matter to local communities, and not events of a wider or global appeal (such as elections or climate change).

In Twitter, *trending topics* of such discussions (created by exchanging or retweeting messages) are essentially *clusters* of these tweets on the basis of their similarity in hashtags. In our case, we also require that those messages be temporally and spatially close, hence we search for messages with *textual similarity* (in their hash-

SSDBM '16, July 18 - 20, 2016, Budapest, Hungary
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4215-5/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2949689.2949716

[1]http://www.facebook.com
[2]http://foursquare.com
[3]http://www.flickr.com
[4]http://twitter.com

tags), *temporal relevance*, and *spatial proximity*. This is not an easy task, considering the sheer volume, the rapid flow, and the great variety in content among all incoming messages. Unfortunately, more than half of the tweets are posted without any hashtags, whereas only 2% of the total messages are geotagged. Hence, we can only utilize a very small portion of tweets for detecting locally trending topics. Despite this deficiency, the outcome of such online analysis can be quite representative of events or emergency situations (e.g., a concert, a football match, or a demonstration), since fairly enough users may not have privacy concerns and could be keen to geotag their tweets so as to manifest their presence in such events.

Therefore, we opt for an approximative mining approach and suggest heuristics in order to obtain indicative, but fast estimates regarding locally trending topics. We employ three levels of discretization, respectively in the temporal, spatial, and textual domain. First, the use of a *sliding window* enables consideration of tweets in a given time horizon, hence the resulting topics should only reflect discussions over the recent past. Second, a fixed *spatial tessellation* (typically a uniform grid of cells) imposes a coarser representation compared to coordinates and can only yield rough estimates. Yet, it avoids costly exact clustering computations [10], and also provides an indirect measure of the locality of topics, since messages in the same cell are considered spatially close. By modifying the granularity of the grid, we can configure the desired level of resolution in the discovered spatial coverage of trending topics. Last, but not least, we extract *only hashtags* from the body of posted messages; apart from fully exploiting the very reason of their existence in Twitter to uniquely characterize the detected topics, these strings are much more amenable to processing due to their short length. This is why we have chosen to address this problem specifically in Twitter, although our methodology may be used to identify such locally concentrated discussions in other microblogs as well (e.g., over Foursquare check-ins or geotagged Flickr photos using keywords instead of hashtags).

In particular, our contribution can be summarized as follows:

- We formulate the problem of approximately detecting locally trending topics from massive, geo- and hash-tagged tweets.
- We propose a stream processing strategy that can effectively identify the spatial coverage of trending topics in real time.
- We conduct an extensive empirical validation of this framework on a real world dataset from Twitter, offering concrete evidence of its efficiency, timeliness, and quality of results.

The remainder of this paper is structured as follows. In Section 2, we introduce a system model and important notions employed in this application framework. In Section 3, we propose a methodology for detecting trending topics and their evolving local footprint across time. Section 4 reports results from a comprehensive empirical study involving a real-world dataset. Section 5 surveys related approaches on online spatial monitoring in social networks. Section 6 offers conclusions and points out further research directions.

2. SYSTEM MODEL

In this Section, we discuss the core notions involved in our framework and we rigorously formulate the problem. Table 1 summarizes the notation used throughout the paper.

Table 1: Notations

G	Uniform grid partitioning of universe $\mathcal{U}$
g	Grid cell granularity per axis
ω	Range of sliding window (in timestamps)
β	Slide step of window (in timestamps), $\beta = \frac{1}{\lambda}\omega, \lambda \in \mathbb{Z}$
τ_c	Timestamp value marking the current execution cycle
$\mathcal{S}$	Stream of posted, geo-hash-tagged messages in Twitter
s	Tweet tuple $\langle \tau, uid, loc, H \rangle$
τ	Timestamp value of a tweet posting
uid	User identifier
loc	Geographic location (in *lat/lon* coordinates) of a tweet
H	Set of hashtags mentioned in a given tweet
N_ω	Number of tweets in a given window instantiation
θ	Similarity threshold for topic detection ($0 < \theta \leqslant 1$)
ϕ	Minimum popularity for a topic to be trending in a grid cell
T_k	Topic detected in a grid cell
$\mathcal{P}$	Set of trending topics detected in any cell of grid G
$\mathcal{T}^g$	Set of topics in cell $g \in G$ over window range ω
$\mathcal{T}^g_t$	Set of topics detected in cell g during interval $(t-\beta, t]$
cnt	Intensity of a topic in a given area (as count of tweets)
Q_k	Set of grid cells where topic T_k is trending
A	Coverage area (of contiguous cells) for a trending topic
$\mathcal{A}[T_k]$	Set of coverage areas for topic T_k

2.1 Twitter Stream

Let $\mathcal{S}$ a stream of messages in Twitter, each marked by a timestamp value τ that denotes the time it was posted. From each actual message, i.e., the set of words W in a tweet posted by user uid, we actually retain only the set of hashtags $H \subseteq W$ and ignore the remainder of the text. We only consider messages that contain at least one hashtag, i.e., $H \neq \emptyset$. Due to their limited size (up to 140 characters), tweets cannot contain more than a few hashtags. Most importantly, we consider only tweets that are *geotagged*, i.e., those that include the actual location loc of the user at the time of posting. This location can be specified either by exact geographic coordinates (latitude, longitude) measured by her mobile device; alternatively, it can be derived through a well-known landmark (like a concert hall, a bar, etc.) mentioned in the message, as long as such a place can be unmistakably pinpointed on the map (e.g., via a gazetteer). Therefore, the incoming tweets are abstracted as a *geostream* of tuples $\mathcal{S} = \{\ldots, \langle\, \tau, uid, loc, H \,\rangle, \ldots\}$. Tuples in $\mathcal{S}$ are ordered by their timestamp value and we assume there are no imperfections (e.g., delayed or garbled messages).

We consider that all processing is carried out by a centralized mechanism, which accepts fresh tweets and issues results at each execution cycle (always denoted by current time τ_c). Typically for such a streaming context, we employ a time-based *sliding window* [18] over $\mathcal{S}$ that can keep up with the continuously arriving messages and always abstract a portion containing the most fresh tweets only. As illustrated in Fig. 1, this window employs a *range* spanning over the most recent ω timestamps backwards from current time τ_c, in order to fetch all tweets posted during time interval $(\tau_c - \omega, \tau_c]$. The window moves forward with time according to a fixed *slide step* of β timestamps. Upon each slide, the window accepts fresh tuples received during most recent interval $(\tau_c - \beta, \tau_c]$, while evicting obsolete ones with time indications earlier than $\tau_c - \omega$. For example, specifying $\omega = 24$h and $\beta = 1$h will provide a subset of $\mathcal{S}$ as the current *state* of the window instantiation, which will get refreshed every hour and always provide all tweets received during the past 24 hours. At each iteration, the window state consists of stream tuples within its current bounds; usually $\beta < \omega$, so successive window instantiations may share tuples (i.e., state overlaps) as depicted with the tweets in the shaded area in Fig. 1. Note that the amount N_ω of tuples within each window instantiation is not fixed, but it fluctuates according to the number of tweets posted in the corresponding time interval (e.g., less posts during night hours, or a burst of tweets following a breaking event).

For smooth transition between successive window states, we pre-

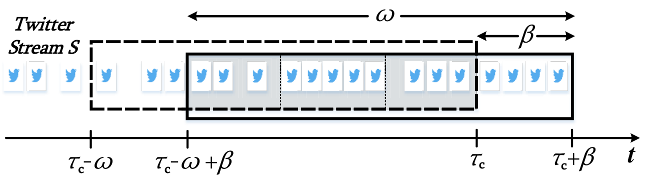


Figure 1: Window of *range* ω and *slide* β on Twitter Stream $\mathcal{S}$

scribe that

$$\frac{\omega}{\beta} = \lambda, \text{where } \lambda, \omega, \beta \in \mathbb{Z}.$$

Hence, window range ω may be considered as the union of λ consecutive, non-overlapping *time frames* (equivalent to *panes* introduced in [15]), each spanning a time period of β timestamps. In the example shown in Fig. 1, the window is composed of $\lambda = 4$ panes. As in [15], this subdivision into panes greatly facilitates window maintenance with time, since at each slide (e.g., at time $\tau_c + \beta$) the oldest pane should be evicted and a new one (containing the most fresh tweets) gets included in the window. Therefore, the contents of all other $\lambda - 1$ panes remain intact (e.g., the three panes in the shaded area in Fig. 1), so that computations performed separately over each such pane need not be repeated.

2.2 Topics in Streaming Tweets

Suppose that $\mathcal{T} = \{T_1, T_2, \ldots, T_k, \ldots T_n\}$ contains all topics detected so far during a given time interval over stream $\mathcal{S}$. Each topic T_k has a *signature*, i.e., it is characterized by the set of hashtags mentioned in any of the tweets that relate to the respective discussion over this period. For example, suppose that three topics have been detected so far from several tweets posted in the past hour. Those topics have signatures $T_1 = \{\#worldcup, \#brazil\}$, $T_2 = \{\#tubestrike\}$, $T_3 = \{\#xfactor, \#song\}$. For any subsequent tweet $s \in \mathcal{S}$ examined from the same batch (this one hour), we must detect whether s belongs to an existing topic $T_k \in \mathcal{T}$; otherwise, if it does not match any topics currently in $\mathcal{T}$, we should create a new topic T_{n+1} using the hashtags from tweet s as its seed.

In order to check such potential matches for tweet $s \in \mathcal{S}$, we make use of function $\mathsf{similarity}(s.H, T_k)$, which takes as arguments the set $s.H$ of hashtags mentioned in tweet s and the signature of an existing topic $T_k \in \mathcal{T}$. This function returns a real value that measures the degree of *textual similarity* between the sets of terms (i.e., hashtags) in each of its arguments, so several such metrics may be used [7]. Some metrics are *character-based* and use edit distances over the compared terms, like Levenshtein, Jaro-Winkler, N-gram, etc.

But instead of looking at them as strings of characters, hashtags in a tweet message can be considered as a set of terms (or tokens). Adhering to this view, several *token-based similarity metrics* can be applied. For example, the *Jaccard similarity* between two such sets of terms (i.e., hashtags) H_1 and H_2 is the number of shared terms over the number of all unique terms:

$$\mathsf{Jaccard}(H_1, H_2) = \frac{|H_1 \cap H_2|}{|H_1 \cup H_2|} = \sigma, \ \ 0 \leqslant \sigma \leqslant 1.$$

The higher this value, the greater the similarity between the two sets; value 1 indicates full coincidence of hashtags.

Cosine similarity is another common vector-based similarity measure. In order to apply it over two sets of terms (i.e., hashtags) H_1 and H_2, a common 'corpus' is first created containing all terms in $H_1 \cup H_2$. Two vectors $V(H_1), V(H_2)$ are constructed, each denoting the frequency of each term of the dictionary in the original sets H_1, H_2 respectively. Then, the Euclidean cosine rule can be used to determine similarity as

$$\mathsf{Cosine}(H_1, H_2) = \frac{V(H_1) \cdot V(H_2)}{\|V(H_1)\| \|V(H_2)\|} = \sigma, \ \ 0 \leqslant \sigma \leqslant 1.$$

Note that in our case of term matching, this function cannot return negative values, since hashtag frequencies cannot be negative. Again, value 1 indicates that the two sets are exactly the same.

If the returned similarity value $\mathsf{similarity}(s.H, T_k) = \sigma$ exceeds a given *threshold* θ, $0 < \theta \leqslant 1$, then we may consider that tweet s matches existing topic T_k. However, the same tweet s may be found similar to several other topics $T_i \in \mathcal{T} \setminus \{T_k\}$, if similarity() returns values above threshold θ when checked against each such topic T_i. In this case, we choose to get s assigned to a single topic, i.e., the one for which the similarity metric is maximized. In the aforementioned example, a fresh tweet including hashtag $\#worldcup$ has a Jaccard index of 0.5 with existing topic $T_1 = \{\#worldcup, \#brazil\}$ and zero similarity with the rest. Hence, with a threshold $\theta = 0.5$ or less, the new tweet will be assigned to topic T_1; otherwise, it will create a new topic with signature $T_4 = \{\#worldcup\}$.

More concretely, a *topic* in Twitter can be defined as follows:

$$T_k = \big\{\langle\{s.H : s \in \mathcal{S}(T_k)\}, |\mathcal{S}(T_k)|\rangle\big\}, \text{ where}$$

$$\mathcal{S}(T_k) = \{s \in \mathcal{S} : (s_k.H \in T_k, \ \mathsf{similarity}(s.H, s_k.H) \geqslant \theta) \wedge (\forall s_i \in \mathcal{S} \backslash \mathcal{S}(T_k), \mathsf{similarity}(s_i.H, s_k.H) < \mathsf{similarity}(s.H, s_k.H))\}$$

represents the subset of tweets that are actually supporting T_k, i.e., their hashtags contributed into its signature $\{s.H : s \in \mathcal{S}(T_k)\}$.

For effectively maintaining topics as the stream evolves, we suggest that the set of detected topics $\mathcal{T}$ be maintained per pane. So, once the window slides, a set $\mathcal{T}$ gets created afresh on the basis of newly published tweets. In the next slide, a new set $\mathcal{T}'$ will be created, without inheriting any of the topics in $\mathcal{T}$. To obtain the topics over the current window range ω, we simply have to merge all such sets created separately for each of the λ panes comprising the entire window. Note that even a topic emanating from a single tweet may be also contained in this unified set.

2.3 Locally Trending Topics

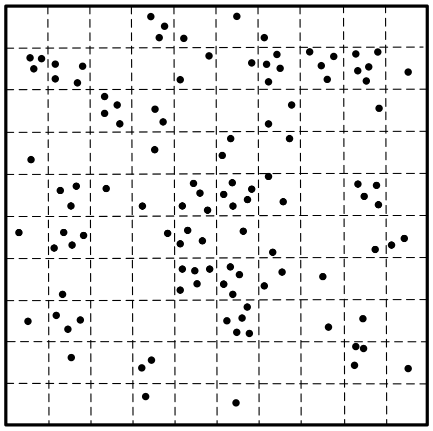
Since our objective is to identify the spatial coverage of topics, we do not characterize topics as *trending* if they only emerge from a significant number of recent related tweets, but if they also happen to be popular *locally*. Let A be an arbitrary spatial area, e.g., a polygon delineating the boundaries of a given neighborhood or county. To quantify the impact of a topic T_k in area A over range ω of the current window, we make use of this function:

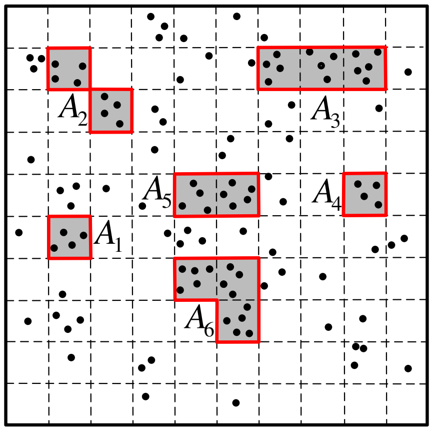
$$\mathsf{popularity}(T_k, A, \omega) = \{s \in \mathcal{S}(T_k) : s.loc \in A \wedge s.\tau \in (\tau_c - \omega, \tau_c]\},$$

which returns the set of related tweets that actually support this topic. As an alternate measure of popularity we could have used the number of distinct users participating into this discussion. However, since we consider that geotagging refers to the tweet itself (i.e., GPS coordinates of the user when posting the message), and we wish to observe the spatial coverage of trending topics, we have chosen to express their popularity in tweet counts. Thus, topic T_k is *locally trending* in a given area A during interval ω once

$$|\mathsf{popularity}(T_k, A, \omega)| \geqslant \phi.$$

Parameter ϕ signifies a *minimum popularity* value that suffices to denote a topic as trending (i.e., supported by enough tweets).





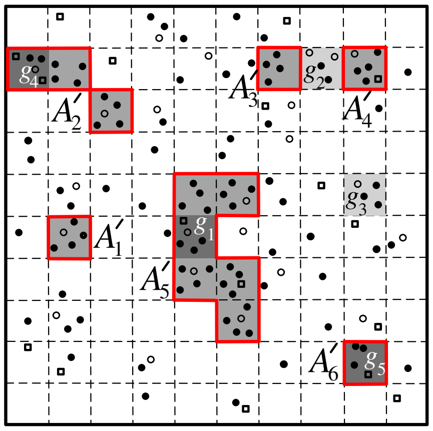


Figure 2: Detecting coverage areas for a specific topic over grid cells

2.4 Spatial Coverage of Trending Topics

The aforementioned definition of a locally trending topic presupposes that area A is known in advance. This may be the case when the entire area of monitoring (Universe $\mathcal{U}$) is subdivided into fixed polygons, e.g., statistical or administrative zones, and we wish to detect trends over such areas. Alternatively, a planar *tessellation* [4] with non-overlapping regions (i.e., *tiles*) may be applied over $\mathcal{U}$, using regular, symmetrically arranged shapes as its building blocks, such as squares, equilateral triangles, hexagons, etc. Such tessellations enable a *data-driven* composition of compact areas, each comprising several contiguous tiles, where a certain phenomenon is observed. Hereafter, we opt for this latter option, as we intend to discover the most extensive regions where a topic is trending, each consisting of a varying number of elementary tiles.

Without loss of generality, in order to determine the *spatial coverage* of topics, we employ a uniform grid partitioning G into a set of $g \times g$ cells over the entire Universe $\mathcal{U}$, as in other similar approaches [2, 11, 17, 21]. Figure 2a illustrates a grid of 10×10 square cells over locations of tweets supporting a given topic T_k in window range ω. At each execution cycle τ_c, we can examine whether topic T_k is trending over grid cells, and provide a set of such cells as an indication of its current local popularity:

$$Q_k = \{g_i \in G : |\mathsf{popularity}(T_k, g_i, \omega)| \geqslant \phi\}.$$

Note that Q_k may be empty, if this topic T_k is not popular enough in any cell. Besides, Q_k may contain discontiguous groups of cells, as illustrated in Fig. 2b. So, we may discover maximal *compact areas of coverage* for topic T_k, each consisting either of multiple contiguous cells in Q_k or singleton ones. More concretely, let $A \subseteq Q_k$ be one such coverage area for topic T_k. Then,

$$A = \bigcup_i \{g_i \in Q_k : \exists\, g_j \in Q_k \setminus \{g_i\}, \mathsf{adjacent}(g_i, g_j) = \mathsf{true}\}$$

represents a compact surface as the spatial union of several cells, where each cell is adjacent with at least another one. Otherwise,

$$A = \{g_i \in Q_k : \forall\, g_j \in Q_k \setminus \{g_i\}, \mathsf{adjacent}(g_i, g_j) = \mathsf{false}\}$$

is formed by a singleton, detached cell such as A_1 in Fig. 2b. Note that $\mathsf{adjacent}()$ is a topological function examining whether any two given cells share an edge or a vertex in grid G. For example, A_2 in Fig. 2b is regarded as one coverage area, although it consists of two cells that they only touch in a common vertex.

Concerning the local trend of a popular topic T_k over time, we may assess whether it is actually *expanding* (i.e., stretching over a greater area) or *contracting* (i.e., shrinking in terms of spatial extent). Again, let Q_k stand for the subset of cells where T_k was trending at execution cycle τ_c (Fig. 2b). In the setting shown in Fig. 2c for the next execution cycle at $\tau'_c = \tau_c + \beta$, some fresh tweets (depicted as squares) appear in certain cells, while some previous tweets expire from the window (depicted as empty bullets). Consequently, the same topic T_k covers a new subset Q'_k of grid cells incurring changes (rearrangement, appearance, disappearance) in the respective coverage areas. For instance, coverage area A'_5 is formed by merging previously separated areas A_5 and A_6 thanks to a newly added cell adjacent to both (cell g_1 depicted in dark grey color in Fig. 2c). At the same time, other coverage areas may split because some of their constituent cells have not enough support for T_k anymore ($\phi = 4$ in this setting). For example, area A_3 gives place into two separate ones A'_3 and A'_4 because topic T_k is no longer trending in cell g_2 at cycle τ'_c. Of course, the shape of other coverage areas may remain intact (e.g., A'_1), whereas new ones may appear (e.g., A'_6) or disappear (e.g., A_4, since T_k ceased to be trending in its only cell g_3).

We stress that the aforementioned notions only refer to the spatial coverage of a given topic T_k, as long as it has a minimum ϕ of popularity in each of the involved cells. Of course, the same topic may be very popular in some coverage area A_i and less popular in another one A_j with regard to the number of related tweets that have been posted in each one. Hence, we can also quantify the *intensity* (i.e., overall popularity) of a trending topic T_k in a coverage area A as the total count of tweets that give enough support to T_k in any of the cells constituting A during the current window range ω. Essentially, this estimated intensity is

$$cnt = |\mathsf{popularity}(T_k, A, \omega)|$$

provided that $\forall\, g \in A, |\mathsf{popularity}(T_k, g, \omega)| \geqslant \phi$.

2.5 Problem Statement

Given a stream $\mathcal{S}$ of geo-hash-tagged tweets $\langle \tau, uid, loc, H \rangle$, we wish to identify the spatial coverage of each trending topic detected over a time-based sliding window of range ω and slide β. The entire region of monitoring (Universe $\mathcal{U}$) is uniformly partitioned into equi-sized square cells organized in a grid G. Assuming that $\mathcal{T}$ is the set of topics already detected from tweets in current window range ω, each incoming tweet $s \in \mathcal{S}$ should be assigned to a single

Algorithm 1 Monitor spatial coverage of topics in Twitter Stream

```
1: Procedure TwitterFootprintMonitor
2: Input: Twitter Stream $\mathcal{S}$ of geotagged tuples $\langle \tau, uid, loc, H\rangle$;
3: Parameter: Grid partitioning $G$ of universe $\mathcal{U}$ in $g \times g$ cells;
4: Parameter: Window specification $\langle$range $\omega$, slide $\beta\rangle$;
5: Parameter: Similarity threshold $\theta$ for hashtags and topics;
6: Parameter: Minimum popularity $\phi$ of a topic per cell;
7: Output: Trending topics $\mathcal{P}$; their spatial coverage $\mathcal{A}$ per cycle;
8: $\tau_c \leftarrow \tau_0$;   //Initiation time of monitoring
9: for each execution cycle spanning $(\tau_c - \beta, \tau_c]$ do
10:    $N_\omega \leftarrow |\{s \in \mathcal{S} : s.\tau \in (\tau_c - \omega, \tau_c - \beta]\}|$;
11:    for each tweet $s$ with $s.\tau \in (\tau_c - \beta, \tau_c]$ do
12:        TopicClustering($s$, $\tau_c$, $\theta$, $\beta$, $G$); //Update topics by tweet s
13:        $N_\omega \leftarrow N_\omega + 1$;   //Update tweet count in window range
14:    end for
15:    $\mathcal{P} \leftarrow$ PopularityFiltering($\tau_c$, $\phi$, $\omega$, $N_\omega$, $G$); //Trending topics
16:    $\mathcal{A} \leftarrow$ CoverageDiscovery($\mathcal{P}$); //Coverage per trending topic
17:    $\tau_c \leftarrow \tau_c + \beta$;   //When to start the next execution cycle
18: end for
19: End Procedure
```

topic $T_k \in \mathcal{T}$, the one with a signature most similar to its hashtags $s.H$, provided that $\mathsf{similarity}(s.H, T_k) \geqslant \theta$, where $0 < \theta \leqslant 1$ is a predefined similarity threshold. Otherwise, tweet s does not support an existing topic, and initiates a new topic that is appended to $\mathcal{T}$ maintained for the current window.

Upon each window slide, we search for a set $\mathcal{P}$ of *locally trending topics*, i.e.:

$$\mathcal{P} = \bigcup_k \{\langle T_k, Q_k\rangle\},$$

$$Q_k = \{\langle g, cnt\rangle : g \in G, cnt = |\mathsf{popularity}(T_k, g, \omega)| \geqslant \phi\},$$

where topic T_k is popular enough in every cell of Q_k, given a minimum popularity value ϕ for a topic to qualify as trending within a single cell $g \in G$.

In addition, we have to discover all *maximal coverage areas* for every trending topic T_k along with its *intensity* in each such area:

$$\mathcal{A} = \Big\{\Big\langle T_k, \bigcup_j \{\langle A_j, cnt_j\rangle\}\Big\rangle : \langle T_k, Q_k\rangle \in \mathcal{P} \wedge (A_j \subseteq Q_k) \wedge$$
$$\wedge\ (cnt_j = \sum_{g \in A_j} Q_k[g].cnt)\Big\},$$

and since every A_j may consist either of a singleton cell or multiple contiguous ones, its overall intensity cnt_j must be derived after summing up local popularity of T_k in the respective cell(s).

3. DISCOVERING FOOTPRINT OF TOPICS

In this Section, we present our framework for monitoring the footprint of locally-trending topics in Twitter, i.e., their coverage areas, their expansion or contraction, as well as local intensity. At each *execution cycle*, it accepts fresh tweets that were posted during the most recent slide β of the window until current time τ_c, hence it processes tweets received in interval $(\tau_c - \beta, \tau_c]$. As a result, it reports all trending topics and their footprint detected amongst the N_ω tweets examined across the actual window range ω. This method employs a uniform grid of cells, which are used as the building block for constructing and rearranging coverage areas.

Algorithm 1 outlines the monitoring mechanism that is applied against the Twitter Stream. In particular, the proposed strategy consists of three successive stages in each execution cycle:

i) *Topic Detection*: Incoming geotagged tweets that fall in the same grid cell are checked for textual similarity on their hashtags; effectively, each message is assigned to a topic currently active in this cell.

ii) *Popularity Filtering*: Each topic detected in each grid cell is compared against a certain popularity value. Topics with enough popularity in a cell are considered as locally trending; the rest are filtered out and excluded from any further consideration in this execution cycle.

iii) *Coverage Discovery*: For each trending topic, cells where it is currently trending are examined so as to identify compact areas of coverage consisting of contiguous cells. It is also assessed whether there is an increase or decrease in the number of affected cells w.r.t. the previous execution cycle, respectively indicating spatial expansion or contraction in this topic's influence.

Before going into the details of each phase, we present several data structures employed in our methodology for effective, real-time processing against streaming, geotagged tweets:

- G: A uniform partitioning of Universe $\mathcal{U}$ into a regular grid of $g \times g$ cells (Fig. 2a). Grid granularity plays an important role not only in performance, but also in the quality of results, as will be discussed in our empirical study.

- *TopicsInfo*: At every cell $g \in G$ and for each window pane $(\tau - \beta, \tau]$, this list holds the set $\mathcal{T}^g_\tau$ of all topics (regardless of popularity) that have been detected strictly during this interval. Assuming that the window consists of $\lambda = \frac{\omega}{\beta}$ panes, there must be λ distinct such lists in every cell. Indeed, a topic may appear in a pane, but not necessarily in the rest.

- $\mathcal{T}^g$: A dictionary per grid cell g that holds all topics (in lexicographic order by their signature) along with their respective popularity during the current window range $(\tau_c - \omega, \tau_c]$. Such a dictionary is derived by merging the *TopicsInfo* lists in each of the λ window panes maintained in a cell.

- Q_k: List of cells where topic T_k is trending.

- $\mathcal{P}$: A dictionary that retains all locally trending topics (in lexicographic order by their signature), which have been detected in the entire Universe $\mathcal{U}$ over the actual window range $(\tau_c - \omega, \tau_c]$. Using a trending topic T_k as a key, this dictionary returns its respective list Q_k of cells.

- $\mathcal{A}$: A dictionary that for each locally trending topic T_k holds a list of its coverage areas (either singleton or contiguous cells), as well as the overall intensity of T_k in each such area.

3.1 Topic Detection

Algorithm 2 presents the various steps employed for refreshing topics by an incoming tweet. Initially, for each tweet s from the recent batch received during interval $(\tau_c - \beta, \tau_c]$, this process determines which cell g is affected, simply by hashing the location $s.loc$ of that message against grid G. Of course, only one such cell g may correspond to a given location. Afterwards, the sets of hashtags $s.H$ mentioned in this tweet will be checked for similarity with topics retained in this cell only. Admittedly, the size of cells actually determines how fresh messages will be split into subsets (one per cell) regardless of textual similarity on their hashtags. Thus, although some messages in close proximity may refer to the same subject, they could be shared by force among multiple cells because of the discretization imposed by the grid. In essence, grid

Algorithm 2 Refresh topic by an incoming tweet

```
1: Procedure TopicDetection (tweet $s$, timestamp $\tau$, threshold $\theta$, window slide $\beta$, grid $G$)
2: $g \leftarrow$ HashLocation$(s.loc)$;   //Grid cell where tweet s appears
3: $\mathcal{T}_\tau^g \leftarrow G[g].TopicsInfo((\tau-\beta,\tau])$; //Topics at latest pane
4: if $\mathcal{T}_\tau^g = \emptyset$ then
5:     $T_{new} \leftarrow \{s.H\}$;        //From hashtags of tweet s, create...
6:     $\mathcal{T}_\tau^g \leftarrow \langle \tau, T_{new}, 1\rangle$;     //... a new topic with popularity 1
7: else
8:     $n \leftarrow |\mathcal{T}_\tau^g|$;        //Number of topics in latest pane at cell g
9:     for $i \leftarrow 1$ to $n$ do
10:        $\sigma[i] \leftarrow$ similarity$(s.H, \mathcal{T}_\tau^g[i])$;   //Checking with $i^{th}$ topic
11:    end for
12:    $i_{max} \leftarrow \text{argmax}_{i\in\{1\ldots n\}}\sigma[i]$; //Points to most similar topic
13:    if $\sigma[i_{max}] \geqslant \theta$ then {        //Update most similar topic}
14:        $\mathcal{T}_\tau^g[i_{max}].topic \leftarrow \mathcal{T}_\tau^g[i_{max}].topic \cup \{s.H\}$;
15:        $\mathcal{T}_\tau^g[i_{max}].cnt \leftarrow \mathcal{T}_\tau^g[i_{max}].cnt + 1$;
16:    else {          //Create a new topic only from this tweet}
17:        $T_{new} \leftarrow \{s.H\}$;
18:        $\mathcal{T}_\tau^g \leftarrow \mathcal{T}_\tau^g \cup \langle \tau, T_{new}, 1\rangle$;
19:    end if
20: end if
21: $G[g].TopicsInfo((\tau-\beta,\tau]) \leftarrow \mathcal{T}_\tau^g$;  //Retain topics in cell
22: End Procedure
```

G acts as a filtering step that provides the set of topics (not only the trending ones, but all topics) detected in each of its cells.

Thanks to the temporal discretization of the sliding window into non-overlapping panes, each fresh tweet s is checked for similarity only against topics in $\mathcal{T}_\tau^g$. These are the topics in cell g already created from messages in the same batch, i.e., over the latest pane $(\tau_c-\beta, \tau_c]$, and are kept in the respective *TopicsInfo* list under cell g. If no topic exists in the list, then a new topic is created from tweet s only, having hashtags $s.H$ as its signature (Lines 4-6).

However, if $n > 0$ topics have been discovered thus far in the latest pane, then tweet s must be tested for potential matching against each one of them (Lines 8-11). In the actual implementation, we make use of token-based similarity metrics Jaccard and Cosine that examine similarity between terms. Therefore, it may happen that hashtags $H_1 = \{\#eurovisionsongcontest\}$ and $H_2 = \{\#eurovision\}$ could create two distinct topics, although we may interpret that they should form a single one. However, as discussed in Section 2, any similarity metric on strings may be employed for this test, as long as it can yield a good estimate of the best possible match, i.e., a topic from $\mathcal{T}_\tau^g$ most similar in its signature with hashtags $s.H$ of the tweet. If this similarity exceeds threshold θ, then the chosen topic is updated by integrating hashtags $s.H$ in its signature, and its popularity in this cell is incremented (Lines 13-15). In case that tweet s does not have sufficient similarity ($< \theta$) with any of the existing topics in $\mathcal{T}_\tau^g$, then s alone creates a new topic (Lines 16-19).

Overall, each fresh tweet affects one topic only (existing or just created); once all tweets in the recent batch are exhausted, set $\mathcal{T}_t^g$ will contain all topics detected over the last window slide (i.e., pane). The example in Fig. 3 shows topics that have been detected in the latest pane at this execution cycle $t = \tau_c$, as well as in the previous $\lambda - 1$ panes constituting the current window. Note that a given topic may arise in a single pane only (e.g., $\{\#africa\}$ in Fig. 3) or in multiple panes. The detection process simply registers all topics (and their popularity) per pane into the *TopicsInfo* list of the respective cell where they all have recently appeared.

Essentially, our method performs *online clustering* into topics of tweets having textual, spatial, and temporal relevance. Once processed in a single pass, raw messages can be discarded and not retained in the grid; only the resulting summary (i.e., the detected

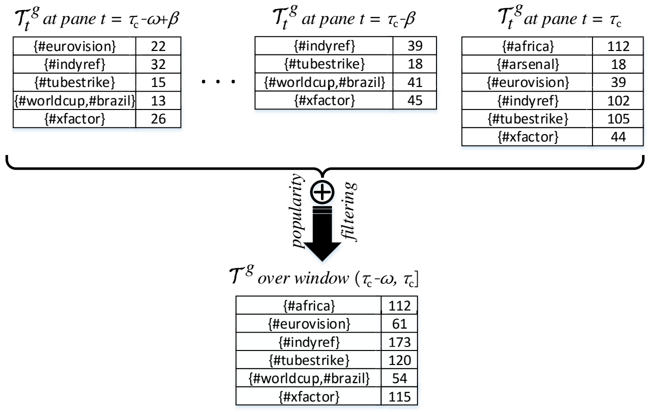


Figure 3: Merging topics and popularity filtering at cycle τ_c

topics) matters in subsequent stages. Our technique is loosely inspired from the clustering paradigm suggested in [3], but it maintains clusters in a totally different fashion. That algorithm was originally intended to detect the k most active clusters from streaming text messages (documents without any spatial features). It applied time-decaying weights on documents participating in clusters in order to account for recency and thus promote the most actively updated clusters. Instead, we employ a sliding window with a fixed range across time; thus, the detected topics may be compared both in terms of content (their signatures) and magnitude (popularity). In addition, by decomposing each window into panes, not only can we perform faster and never repeat computations, but also retain topics at finer temporal resolution (spanning β timestamps).

3.2 Popularity Filtering

This stage identifies *locally trending topics* in each cell of the grid by filtering those topics detected in any pane of the current window. As mentioned in Section 2, we specify a parameter ϕ as the *minimum popularity* that a topic must have in order to qualify as trending in a given cell. This can be defined in several flavors:

- An *absolute* value, i.e., a minimum number of tweets (e.g., $\phi = 50$) that should support a given topic T_k.
- A *density* value expressed as tweet count per area units, e.g., at least $\phi = 20$ tweets per km^2.
- A *percentage* value ($0 < \phi \leqslant 1$), e.g., prescribing that at least $\phi = 2\%$ of all recent messages must support topic T_k.

Without loss of generality, in the sequel we make use of a percentage value over the total amount N_ω of tweets in the current window. Hence, a given topic T_k is currently trending in cell g if its popularity therein is $\geqslant \phi \cdot N_\omega$. This adjustment can better cope with seasonal, daily, or sudden fluctuations in the rate of posted messages. Indeed, the same T_k may be supported by an equal number of tweets over two different window instantiations $(\tau_1 - \omega, \tau_1]$ and $(\tau_2 - \omega, \tau_2]$, e.g., in two different days, but it may qualify as trending only in the latter, because less tweets were posted during that period, and thus the relative importance of T_k is greater.

The pseudocode for this process is listed in Algorithm 3. First, the currently active topics in each grid cell g must be identified (Lines 5-6). Thanks to topic maintenance per pane, this involves eviction of the set of topics $\mathcal{T}_t^g$ detected at cycle $t = \tau_c - \omega$ (i.e., the pane that just expired from the window), without affecting the

Algorithm 3 Identify locally-trending topics at an execution cycle

```
1: Function PopularityFiltering (timestamp $\tau$, min popularity $\phi$,
   window range $\omega$, tweet count $N_\omega$ in current window, grid $G$)
2: Output: $\mathcal{P} = \cup_k \{T_k, \{g \in G : T_k$ is popular in cell $g\}\}$;
3: $\mathcal{P} \leftarrow \emptyset$; //Locally-trending topics to be discovered at this cycle
4: for each cell $g \in G$ do
5:     DropExpiredTopics($g, \tau-\omega$);//Topics in evicted window pane
6:     $\mathcal{T}^g \leftarrow$ MergeTopics($g, \omega$); //All topics in window at cell g
7:     for each topic $T_k \in \mathcal{T}^g$ do
8:         $p \leftarrow \mathcal{T}^g[k].cnt$;    //Popularity: support for $T_k$ in cell g
9:         if $p \geqslant \phi \cdot N_\omega$ then
10:            $\mathcal{P} \leftarrow \mathcal{P} \cup \{\langle T_k, \langle g, p\rangle\rangle\}$;   //$T_k$ is trending in this cell
11:        end if
12:    end for
13: end for
14: return $\mathcal{P}$;   //For each topic, list the cells where it is trending
15: End Function
```

sets maintained in any other pane. From these sets over the subsequent λ panes, the algorithm merges topics with *identical signatures* and sums up their popularity over the entire window. This task is greatly expedited because topics in any pane dictionary $\mathcal{T}^g_\tau$ are ordered lexicographically, whereas their signatures are also defined by hashtags in lexicographic order. Hence, dictionary items are organized by using the first letter of their signature as a key. Given a topic T_k we can directly access items starting with the same letter as the signature of T_k and then identify appearance of a given topic T_k in linear time. Once merging is completed, a unified dictionary $\mathcal{T}^g$ is available and lists every topic in the window along with its overall popularity in cell g. Since $\mathcal{T}^g$ also contains the overall popularity of each entry T_k in cell g, we can trivially determine if it is trending there with a simple comparison to the adjusted minimum popularity $\phi \cdot N_\omega$. Every qualifying T_k is then appended to the dictionary $\mathcal{P}$ of locally trending topics. $\mathcal{P}$ eventually collects all such topics across the entire Universe $\mathcal{U}$, and for each topic T_k, it lists the set of cells Q_k where T_k is trending and its respective popularity in each cell (Lines 7-12).

This process is illustrated in Fig. 3 with example topics over several panes. Note that topics not sufficiently supported by tweets are excluded from the merged dictionary $\mathcal{T}^g$ over the entire window. For instance, topic $\{\#arsenal\}$ is not listed in the trending topics, because it has very few appearances and falls behind in popularity.

3.3 Coverage Discovery

Having identified the locally trending topics $\mathcal{P}$ (at this execution cycle t_c) facilitates a lot the task of identifying maximal areas of coverage for each topic. Indeed, as shown in the pseudocode in Algorithm 4, we probe each cell g_i where a given topic T_k is trending and we check if g_i is adjacent with another coverage area A_j (maybe currently consisting of a singleton cell) already created in the same cycle t_c. In case of adjacency (Lines 10-13), this area A_j should be replaced by its spatial union with cell g_i. However, the same cell may be also found contiguous with multiple coverages. For the setting in Fig. 2c, cell g_1 is adjacent to both areas A_5 and A_6, which were disjoint at the previous execution cycle in Fig. 2b. Such a cell acts as the 'connecting glue' that can unify previously disjoint coverage areas into greater ones without gaps. To this end, intermediate list dA is used to keep any such contiguous cells that must be consolidated into a single coverage area A_j. Its intensity is equivalent to the total popularity of topic T_k in these cells (Lines 16-19). Of course, list dA is empty if current cell g_i is not adjacent to any of its fellow cells in Q_k; then, a coverage area is formed from this cell only (Lines 20-23).

Moreover, finding alterations in spatial coverage of topic T_k between successive execution cycles τ_c, τ'_c can be done by checking changes in its local popularity, i.e., comparing the respective subsets Q_k, Q'_k of cells (those in the thick red boundaries in Fig. 2):

Algorithm 4 Find coverage areas in the grid for all trending topics

```
1: Function CoverageDiscovery (locally trending topics $\mathcal{P}$)
2: Output: Coverages $\mathcal{A} = \{\langle T_k, \cup_j\{\langle A_j, cnt_j\rangle\}\rangle\}, \forall\, T_k \in \mathcal{P}$
3: for each topic $T_k \in \mathcal{P}$ do
4:     $\mathcal{A}[T_k] \leftarrow \emptyset$;  //Initialize list of coverage areas for topic $T_k$
5:     $Q_k \leftarrow \mathcal{P}[T_k]$;        //Set of cells where $T_k$ is now trending
6:     for each $\langle g_i, cnt_i\rangle \in Q_k$ do
7:         $dA \leftarrow \emptyset$;           //Set of contiguous cells to consolidate
8:         if $\mathcal{A}[T_k] \neq \emptyset$ then
9:             for each $A_j \in \mathcal{A}[T_k]$ do
10:                if adjacent($g_i, A_j$) then
11:                    $dA \leftarrow dA \cup A_j$;
12:                    $\mathcal{A}[T_k] \leftarrow \mathcal{A}[T_k] \setminus A_j$;   //$A_j$ should be replaced
13:                end if
14:            end for
15:        end if
16:        if $dA \neq \emptyset$ then
17:            $A_j \leftarrow dA \cup \{g_i\}$; //Compact area of contiguous cells
18:            $cnt_j = \sum_{g \in A_j} Q_k[g].cnt$;         //Overall intensity
19:            $\mathcal{A}[T_k] \leftarrow \mathcal{A}[T_k] \cup \langle A_j, cnt_j\rangle$;
20:        else
21:            $A_j \leftarrow \{\langle g_i, cnt_i\rangle\}$; //Coverage area of a single cell
22:            $\mathcal{A}[T_k] \leftarrow A_j$;        //...where $T_k$ is now trending
23:        end if
24:    end for
25: end for
26: return $\mathcal{A}$;
27: End Function
```

- $D^+_k = \{g_i \in G : g_i \notin Q_k \;\wedge g_i \in Q'_k\}$ contains every cell where the given topic T_k just became trending at τ'_c.
- $D^-_k = \{g_i \in G : g_i \in Q_k \;\wedge g_i \notin Q'_k\}$ contains cells where previously popular topic T_k has just lost its popularity at τ'_c.

Note that these two indicators are computed for each trending topic T_k over the entire Universe $\mathcal{U}$, and not separately for each individual coverage area A_j. In the example setting of Fig. 2c, we can easily observe that $D^+_k = \{g_1, g_4, g_5\}$ shown in dark color, whereas $D^-_k = \{g_2, g_3\}$ marks the two cells in light grey color.

From these sets of cells, we can easily identify:

– *Expansion*: If $|D^+_k| - |D^-_k| > 0$, we consider that topic T_k has increasing spatial coverage from time τ_c to τ'_c.

– *Contraction*: Otherwise, if $|D^+_k| - |D^-_k| < 0$, this topic T_k has shrunk in terms of its spatial coverage from time τ_c to τ'_c.

Thus, in the example of Fig. 2, we can infer that topic T_k has expanded by $|D^+_k| - |D^-_k| = 1$ cell. We stress that this only serves as a rough indication of the spatial mutability for this topic. Even if the coverage area remains unchanged in shape, the intensity of the topic may actually fluctuate between successive cycles.

4. EMPIRICAL STUDY

In this Section, we empirically validate the proposed methodology against real data from Twitter and we assess its capabilities in terms of performance, timeliness, and quality of results.

4.1 Experimental Setup

In order to empirically validate our method, we obtained a dataset of 12,625,312 real messages posted in Twitter. This is actually a small subset of the data collected through the Twitter API from around the world, and was also used in [8, 9]. However, neither were all these posts geotagged nor did they all contain hashtags. In addition, there were very few posts before January 2014 or after October 2014. The vast majority of those tweets were found in the Greater London urban area. Hence, we finally retained only tweets tagged with a location inside a box of 50×50 km^2 (Universe $\mathcal{U}$) enclosing London, which they also included at least one hashtag, for a time period spanning from 30-12-2013 to 14-10-2014. The resulting dataset $\mathcal{S}$ contains only 807,479 timestamped, geotagged tweets with hashtags (the remainder of each message is ignored). Naturally, fluctuations in the rate of such postings were expected and can be observed in Fig. 4, which indicates how many both *geo-* and *hash*-tagged tweets were collected throughout this period. But our major concern was that only a meagre 2,794 such tweets were available per day. Of course, such volume cannot be really considered as a data stream; hence, we artificially inflated this dataset. In particular, each original tweet was repeated 9, 19, 49, and 99 times, leading respectively to datasets with an increase factor of $10\times$, $20\times$, $50\times$, and $100\times$ the size of $\mathcal{S}$, so as to test performance of our method against scalable volumes of tweets.

Our framework was implemented in GNU C++ and it runs entirely in main memory. We simulated a streaming behavior by accepting batches of fresh tweets within a specified window according to their original timestamp order. All experiments were conducted on an Intel(R) Core(TM) i7 CPU at 2.4 GHz with 8GB RAM running Ubuntu Linux. Table 2 lists the parameters and their range of values tested in the experiments; default values used in most simulations are shown in bold. In the experiments, we measured average processing time per execution cycle separately for each phase (Topic Detection, Popularity Filtering, and Coverage Discovery). The total execution time per cycle is the sum of these partial costs. In addition, we counted the amount of *locally trending topics* in every cycle and we display their average per cycle (denoted simply as *#topics* in the graphs). For brevity, performance graphs combine a bar chart on processing times (values on the left y-axis) and a line plot on trending topics (values on the right y-axis).

4.2 Performance Results

4.2.1 Scalability

Since the number $|\mathcal{S}|$ of available geotagged tweets was quite limited, we multiplied each message several times in order to obtain scalable volumes of data. As we 'replay' each of these streams by their original timestamp values, this is practically equivalent to increasing the *posting rate* of tweets accordingly. The plots in Fig. 5 display the average execution cost for each resulting dataset for two indicative values for minimum popularity ϕ, whereas the window range is fixed at $\omega = 24$h and it slides every $\beta = 1$h. In both graphs, Topic Detection is the phase that gets affected by such increases in data volume. Indeed, this cost grows linearly with the amount of posted tweets (Table 3), since every newly arriving tweet

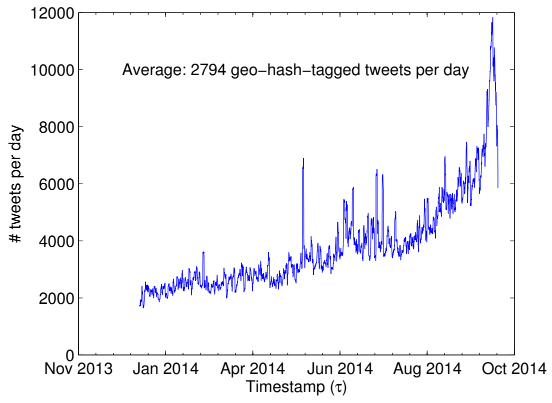


Figure 4: Fluctuations in the rate of posted tweets across time

must be checked against topics already detected at the same cycle. In Table 3, also note that the average number of tweets supporting a given topic (not necessarily trending) also increases linearly.

But observe that the overhead from Popularity Filtering is almost constant. No matter how many times we artificially repeat each message, all copies will be assigned to a single topic during Topic Detection; hence the same number of distinct topics will be next checked for popularity, regardless of the increase factor in the original data. However, the cost on Popularity Filtering differs for varying values of ϕ. With a large value of $\phi = 0.5\%$ (Fig. 5b), less topics qualify as trending (almost 1.5 topics per cycle), since there are not enough related tweets in each window in order to reach this high threshold. To give an idea of its magnitude for the $100 \times |\mathcal{S}|$ dataset, a value $\phi = 0.5\%$ prescribes that on average at least $380108 \cdot 0.5\% \approx 1900$ tweets in the window ($\omega = 24$h) should discuss a given topic in a single cell, so that this topic qualifies as trending. Obviously, very few topics can meet this tough target.

In contrast, close to 9 trending topics are detected per cycle with a lower $\phi = 0.2\%$ (Fig. 5a), a popularity value that for the $100\times|\mathcal{S}|$ dataset translates into an average threshold of 760 related tweets per cell, hence more trending topics emerge. Merging topics from various window panes and creating a common dictionary with the cells they appear costs more if their number grows, hence the cost of this stage depends on the amount of trending topics.

Regarding the last stage for Coverage Discovery, its cost is negligible since it actually involves searching for maximal compact areas over a limited number of cells per topic. This is observed in all experiments, and clearly demonstrates how much it benefits from the effect of summarization (in time, space, and topics) achieved in the preceding stages over the massive stream of tweets. As a stress test on the proposed method, all other experiments were conducted against the largest dataset ($100 \times |\mathcal{S}|$) .

4.2.2 Effect of Grid Granularity

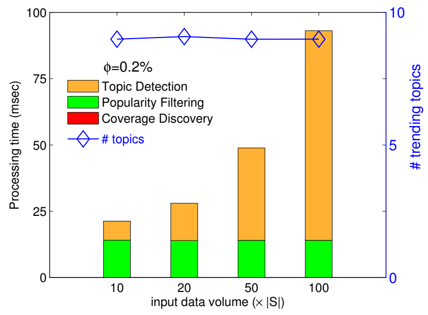
In this experiment, we vary the granularity g per axis, i.e., the number $g \times g$ of cells in the grid partitioning. Figure 7 plots graphs regarding performance against two different values of minimum

Table 2: Experiment parameters

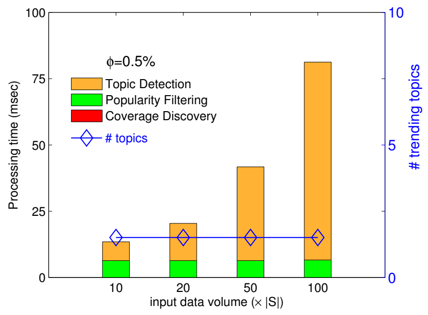
Volume $\|\mathcal{S}\|$ of original dataset	807,479 tweets
Increase factor over original $\|\mathcal{S}\|$	$\times10$, $\times20$, $\times50$, $\times$**100**
Grid granularity g per axis	10, 20, 25, 30, **50**, 75, 100
Range ω of sliding window	2h, 4h, 12h, **24h**
Slide step β of window	**1h**, 2h, 4h, 8h
Similarity metric	**Jaccard**, Cosine
Threshold θ for topic similarity	0.3, **0.5**, 0.6, 0.8
Min popularity ϕ of topic in a cell (as % in # tweets within range ω)	0.02, 0.03, 0.05, 0.1, **0.2**, 0.3, 0.5

Table 3: Average measures on Topic Detection for $\theta = 0.5$

Increase factor over $\|\mathcal{S}\|$	$\times$**10**	$\times$**20**	$\times$**50**	$\times$**100**
# tweets in window (ω =24h)	38011	76022	190054	380108
# tweets on topic per cell	14.1	28.4	70.8	140.1



(a) $\phi = 0.2\%$



(b) $\phi = 0.5\%$

Figure 5: Scalability

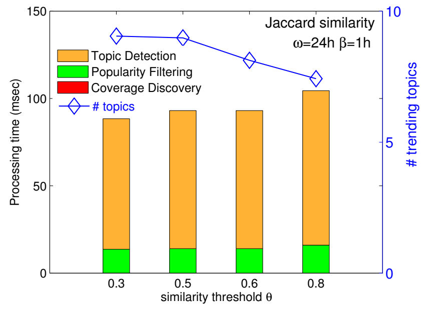
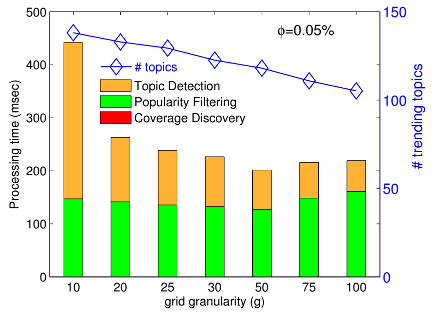
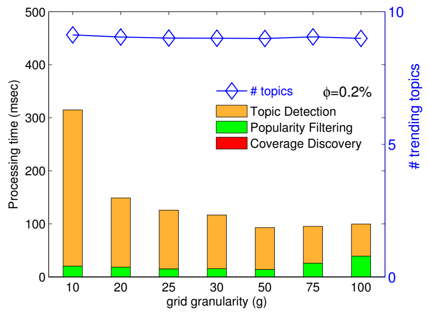


Figure 6: Sensitivity to Jaccard similarity metric



(a) $\phi = 0.05\%$



(b) $\phi = 0.2\%$

Figure 7: Varying grid granularity

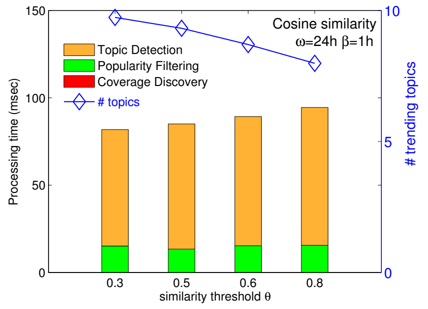


Figure 8: Sensitivity to Cosine similarity metric

popularity per cell, i.e., a small $\phi = 0.05\%$ and a moderate one $\phi = 0.2\%$. In both cases, we note that the finer the granularity, the lower the cost for Topic Detection. With smaller cells, fewer tweets fall within each one, and generally yield less topics (especially those created from just a single tweet). Hence, fewer similarity checks with existing topics are required per incoming tweet.

In contrast, negligible variations are observed in the cost of Popularity Filtering for up to 50×50 cells. For finer subdivisions of the grid ($g = 75$ or 100), this processing takes a little longer, because final dictionary $\mathcal{P}$ of locally trending topics has to collect information from many more cells where each entry is currently popular.

Overall, the cost for detecting topics from tweets steadily drops, whereas finding trending topics slightly grows when the grid is finer. Again, the overhead from Coverage Discovery is negligible. Interestingly, note that the total execution cost gets minimized at $g = 50$ in both graphs, so this value seems suitable for this particular application setting in London. Thus, we henceforth conduct the rest of the experiments with a grid of 50×50 cells.

Concerning the discovered trending topics, we note that their amount for $\phi = 0.05\%$ is diminishing when more cells are employed. This must be considered as an inherent side-effect of the fixed partitioning imposed by the grid. Suppose that a topic T_k is found popular within a large cell (e.g., for $g = 10$, cell size is 5×5 km^2 in this setting for London), as more related tweets can be found therein. With $g = 20$, that cell is split into four pieces, each of $2.5 \times 2.5 = 6.25$ km^2, and popularity for T_k must be checked separately for each one. Chances are that topic T_k may not qualify as trending in none, if it has not sufficient tweets (above ϕ) in any of these smaller cells. This phenomenon is not observed when the minimum popularity barrier is increased to $\phi = 0.2\%$. This is because only very trending topics can cross such a high barrier; such topics must have very strong presence locally, i.e., several tweets on a common subject that create something like a ‘hotspot’.

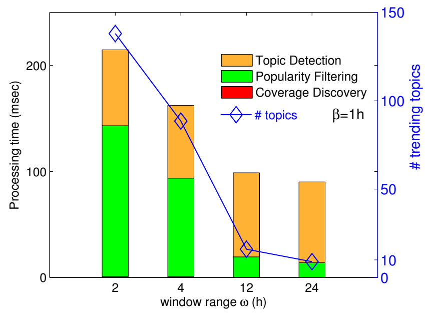
4.2.3 Sensitivity to Similarity Threshold

In order to detect topics amongst tweets, we performed tests using the Jaccard and Cosine similarity metrics against their hashtags. A varying threshold θ was applied in order to check the sensitivity of these metrics and its effect on detected topics.

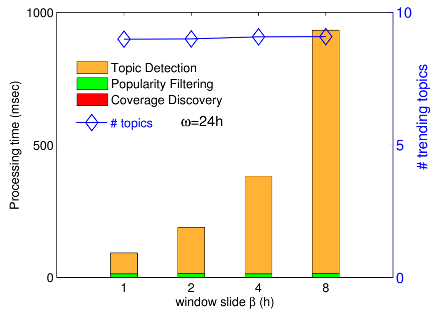
In Table 4, we observe that with a higher threshold θ, the average number of topics detected from tweets per cycle increases (measurements shown only for the Jaccard metric; results for Cosine are similar). This is no wonder, since a higher threshold makes it more difficult for two sets of hashtags to qualify as matching. Hence, each set becomes a distinct topic with its own signature and thus more, but less popular topics will emerge; consequently, fewer can be later characterized as trending. This is exactly the meaning of the line plot in Fig. 6 concerning trending topics.

A subtle issue during topic creation is that their signature may be evolving. The signature of a topic T_k starts with the hashtags of a seed tweet, but extra hashtags may be added once a fresh tweet is found similar to T_k. This may decrease similarity of tweets that had been previously assigned to T_k, perhaps even making them no longer similar enough ($\geqslant \theta$) to the new signature of T_k. If such a tweet were to be probed now, then it might not have been assigned to T_k. But from Table 4, we observe that a signature seldom includes more than two hashtags, and only for a very low similarity threshold $\theta = 0.3$. Hence, the risk of putting loosely similar tweets into the same topic is practically negligible with $\theta \geqslant 0.5$.

With respect to processing cost with a Jaccard metric (Fig. 6), Topic Detection takes more time with higher θ, as it has to maintain a slightly larger number of topics. Cosine similarity metric seems



(a) Varying range ω.



(b) Varying slide β.

Figure 9: Effect of window size

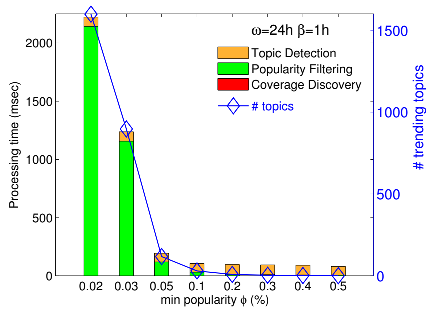


Figure 10: Sensitivity to minimum popularity ϕ

Table 4: Effect of θ on topic detection (Jaccard **similarity**)

Similarity threshold θ	**0.3**	**0.5**	**0.6**	**0.8**
Average # topics	2643	2720	2741	2758
Average # trending topics	9.04	8.98	8.12	7.42
Average # hashtags per topic	2.1	1.83	1.68	1.67

marginally better in both processing cost and amount of trending topics (Fig. 8). Note that the choice of a metric has negligible impact on subsequent phases, mainly because there are no dramatic fluctuations in the amount of topics detected from raw tweets.

4.2.4 *Effect of Window Specification*

Figure 9 illustrates the effect of window specification on performance. By fixing the slide step to $\beta = 1$h and specifying wider ranges ω, we observe (Fig. 9a) that the amount of discovered trending topics gets reduced. This is no surprise, since any such topic must be supported at least by $\phi \cdot N_\omega$ tweets, i.e., this barrier is directly proportional to the total number N_ω of tweets in the current window. The wider the window, the more tweets will be contained in its range; hence the popularity barrier will become higher and much less topics may qualify. This also explains why processing time for Popularity Filtering drops with increasing ω.

With respect to varying sliding steps β while keeping a fixed range $\omega = 24$h, the window moves forward less frequently, but at each slide has to accept a greater batch of incoming tweets. By doubling β as in Fig. 9b, double as many tweets (on average) must be checked for similarity with existing topics. This justifies why the cost of Topic Detection escalates. On the contrary, processing time for Popularity Filtering remains stable, and the number of discovered trending topics does not change either. Indeed, these calculations are performed over the entire window range ω, which is fixed and eventually comprises all detected topics regardless of sliding step.

4.2.5 *Sensitivity to Minimum Popularity*

We have conducted tests specifying the minimum popularity ϕ as an absolute number of tweets that must support a trending topic, as a density of tweets per area units, or as a percentage of the recently posted tweets within the window range ω. Due to lack of space, in Fig. 10 we only report results for a varying ϕ specified as percentage; the other variations offer similar conclusions, although they return differing numbers of trending topics.

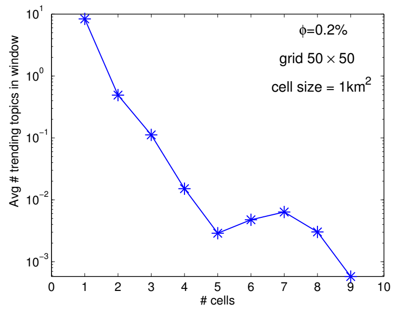
We stress that ϕ makes a lot of difference on Popularity Filtering, since it controls designation of trending topics. With a very low $\phi = 0.02\%$, far too many topics are regarded as trending, even though each one may be supported by just a handful of tweets in a cell. In that case, the overhead of Popularity Filtering dominates the overall execution cost at each cycle, since it has to merge larger lists of topics from the panes that constitute the entire window. On the opposite side, a very high $\phi = 0.5\%$ may cut off a lot of topics that could have significant local influence but still fall short of such a high barrier; hence, their merging and filtering takes practically no time. So, it seems that moderate values of ϕ around 0.2% seem more suitable, both in terms of cost, as well as for effectively locating meaningful trending topics in Popularity Filtering. Note that ϕ is not involved in Topic Detection, so that cost remains unaffected.

4.3 Quality of Spatial Coverages

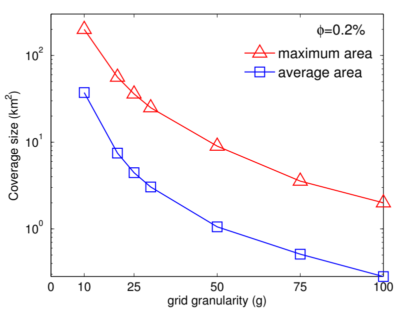
In this set of experiments, we examine the spatial coverage of trending topics discovered by our method. The plot in Fig. 11a (in logarithmic scale) depicts how many trending topics on average stretch over a varying number of cells, i.e., the set Q_k of cells for a given topic T_k used in Coverage Discovery. It turns out that most trending topics are present in a single cell only, whereas less than one topic per window is located in two cells (maybe not contiguous), and rarely does a topic stretch in multiple cells in this 50×50 grid. Considering that each such cell has an area of 1 km^2, this test indicates that locally trending discussions can be captured at a reasonable spatial resolution.

Of course, the size of coverage areas strongly depends on the underlying grid subdivision. Figure 11b reveals that the *average* area (in km^2) of a coverage decreases dramatically with ever finer grid partitions, thus focusing more precisely on the locality of discussions. Also note that the *maximum* coverage of contiguous cells discovered throughout the examined dataset ($100 \times |\mathcal{S}|$) appears very extensive (200 km^2) for a 10×10 grid, but it covers 9 km^2 in a 50×50 grid, and only 2 km^2 in the finest 100×100 subdivision.

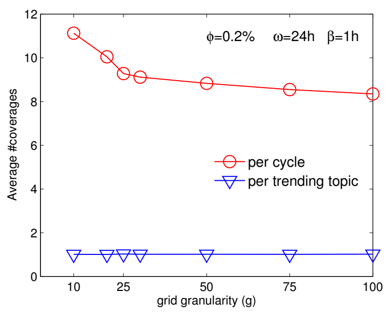
However, the number of discovered coverage areas shows little variations for different cell sizes (Fig. 11c). In each execution cycle, more than 11 coverages are found in a 10×10 grid, which drops to almost 8 coverages for the finest 100×100 grid. This is due to the fixed minimum popularity $\phi = 0.2\%$, which is used to probe each cell in isolation. In particular, a large cell may include enough tweets that can render a topic trending and thus lead into a coverage area; this may not occur so frequently as cells become smaller and smaller. It is interesting that on average about one (compact) coverage area is discovered per trending topic. Indeed, the vast majority (94%) of trending topics were only spotted in a solitary cell; just a small portion (6%) of topics had a larger extent and formed several discontiguous coverage areas. Due to this rigid grid partitioning, some topics may not be identified if their respective messages are shared among several neighboring cells, so such topics could be



(a) Variations in spatial coverage

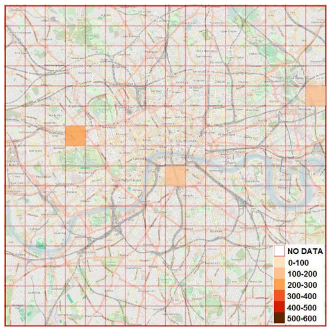


(b) Size of discovered coverage areas

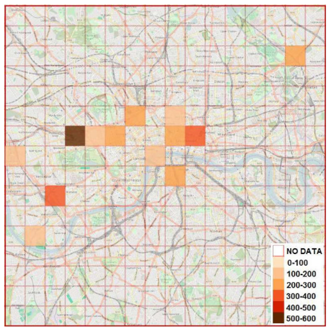


(c) Number of discovered coverage areas

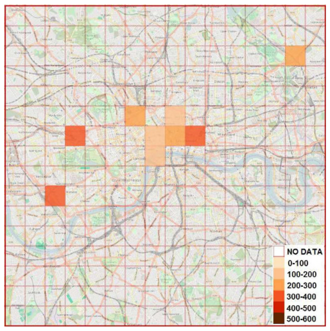
Figure 11: Quality of the spatial coverage for trending topics



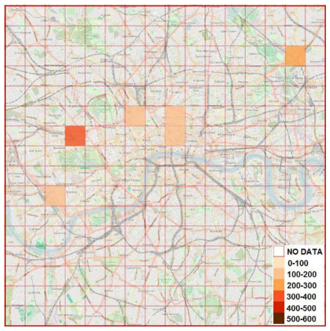
(a) $\tau_c = 1398751200$



(b) $\tau_c = 1398909600$



(c) $\tau_c = 1398931200$



(d) $\tau_c = 1398945600$

Figure 12: Coverage areas of varying intensity on topic *#tubestrike* discovered in the center of London at several execution cycles

falsely recognized with decreased popularity.

To get a clearer insight on the discovered coverage areas, Figure 12 illustrates snapshots taken at several execution cycles on a specific topic *#tubestrike* that was trending in London before and on 1 May 2014. Clearly, this topic became more popular with time and stretched in several zones. Those are coverage areas, consisting either of multiple contiguous cells or detached ones. Note that the number, the size, as well as the shape of these coverages evolves with time, indicating that this discussion was expanding and then contracting. Its intensity was also varying, and darker shaded cells indicate greater number of tweets on *#tubestrike*. So this topic became increasingly trending locally, reached a peak popularity in the cell enclosing Paddington station, and then gradually faded.

5. RELATED WORK

Our framework relates to several recent works on spatial, temporal and textual analysis of streaming data. First, current approaches on *spatio-textual data* like [6, 9] build hybrid indices in order to answer top-k, range, or spatio-textual similarity queries, but ignore the temporal dimension and it is not obvious how to adapt them to work in a streaming fashion.

Spatial clustering could have been employed to group locations of tweets based on their proximity in the Euclidean plane. Density-based methods, like the widely applied DBSCAN [10] can identify clusters of arbitrary shape and size, but involve a lot of distance computations and cannot be affordable in online fashion. Hierarchical clustering techniques like BIRCH [24] are single-pass, but they cannot maintain clusters over a sliding window, i.e., with expiring tweets. The method proposed in [12] offers efficient continuous maintenance of clusters, but can be only applied over a fixed population of moving objects, thus not for fluctuating tweets. Place clustering in geosocial networks [20] extends traditional density-based approach by also considering interconnections of users that visit those places. Instead, in our method we ignore user links and focus on the posted messages in order to cluster them into topics.

Tracking of *bursts* of related messages in social networks or similar documents in microblogs has attracted a lot of research attention recently, especially regarding their temporal, spatial or spatiotemporal properties. In order to identify geographically focused bursts, the sophisticated algorithms proposed in [17] also make use of a grid. They label the state of grid cells as bursty or not w.r.t. documents involving specific keywords by minimizing a cost function on state transitions. But their focus is strictly on queries regarding spatial distributions of bursts on specific keywords and over fixed time intervals, as opposed to our data-driven method for online detection of locally trending topics. The method in [14] captures spatiotemporal burstiness by identifying regional patterns online, i.e., rectangular areas where a given term (i.e., keyword) appears at an unusually high frequency during a maximal time interval. However, this work also considers that the keywords of interest are specified at query time, whereas variations in spatial distribution of documents could return rectangles with large dead space.

For identifying *local events* in social media, the mechanism in [23] automatically assigns a location to non-geotagged documents and analyzes terms co-occurring in many documents that may define such an event. GeoScope [5] can detect correlations between topics and locations over a sliding window. As a result, it offers a list of topic-location pairs that are trending; this differs a lot (both as a notion as well as in processing) from the coverage areas that we discover. The methodology in [21] employs indexing techniques for summarizing frequent terms across a hierarchical spatiotemporal grid in order to provide the top-k most frequent ones in a user-specified spatiotemporal range. A data cube with a spatiotemporal hierarchy is built in [11] in order to cluster hashtags at various resolutions and thus identify local and burst events.

Regarding the framework suggested in [1, 2] specifically over the Twitter Stream, it first ranks all keywords within a sliding window by their burstiness taking into consideration all incoming tweets. Then, using geotagged tweets only, it estimates the spatial distribution of each bursty keyword, i.e., grid cells where it has high density expressed with the number of users that posted related messages. Finally, these distributions are adjusted over greater areas, by taking into account the inherent sparsity and noise in geotagged tweets. Although this setting is relevant to ours, it differs in several crucial aspects. First, we avoid treating and scoring individual keywords, but we attempt to assign them into similar, broader topics. We measure popularity of a topic w.r.t. the amount of tweets and not that of the users. Most importantly, we identify areas where each topic was trending and not just the presumed epicenter of the discussion (i.e., the centroid of the detected event [2]).

6. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a framework for monitoring how trending topics in Twitter evolve both in space and time. From geotagged tweets, our technique can recognize groups with similar hashtags, which effectively constitute evolving topics of discussions. Since many of these topics are active only locally, we can then spot their respective areas of coverage where they are particularly popular. We can also characterize them as expanding or contracting in terms of spatial extent, and quantify their intensity across time. A series of experiments on real-world Twitter data have shown that this framework can offer approximate, yet timely results (in less than a second) with very small overhead.

In the future, we plan to extent this technique with a more flexible spatial partitioning in a hierarchical fashion like [22] and on-the-fly adjustment of parameters for adequately adapting to fluctuations in the incoming stream of tweets. Parallelization is also challenging, where processing nodes could be used to monitor disjoint groups of cells and exchange coverages when necessary.

Acknowledgements

This work was partially supported by the EU Project *City.Risks* (H2020-FCT-2014-653747). We also wish to thank Christodoulos Efstathiades, Alexandros Belesiotis, and Dimitris Skoutas for providing the dataset used in the experiments.

7. REFERENCES

[1] H. Abdelhaq, M. Gertz, and C. Sengstock. Spatio-temporal Characteristics of Bursty Words in Twitter Streams. In *ACM SIGSPATIAL*, pp. 194-203, November 2013.

[2] H. Abdelhaq, C. Sengstock, and M. Gertz. EvenTweet: Online Localized Event Detection from Twitter. *PVLDB*, 6(12): 1326-1329, August 2013.

[3] C.C. Aggarwal. Mining Text and Social Streams: a Review. *ACM SIGKDD Explorations*, 15(2): 9-19, December 2013.

[4] B. Boots. Spatial Tessellations. In: P. Longley, M. Goodchild, D. Maguire, and D. Rhind (eds): *Geographical Information Systems: Principles, Techniques, Applications and Management*. Wiley, New York, pp. 527-542, 1998.

[5] C. Budak, T. Georgiou, D. Agrawal, and A. El Abbadi. Geoscope: Online Detection of Geo-Gorrelated Information Trends in Social Networks. *PVLDB*, 7(4): 229-240, 2013.

[6] L. Chen, G. Cong, C.S. Jensen, and D. Wu. Spatial Keyword Query Processing: An Experimental Evaluation. *PVLDB*, 6(3): 217-228, 2013.

[7] W. Cohen, P.D. Ravikumar, and S.E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *IIWeb*, pp. 73-78, 2003.

[8] H. Efstathiades, D. Antoniades, G. Pallis, and M. Dikaiakos. Identification of Key Locations based on Online Social Network Activity. In *ASONAM*, pp. 218-225, August 2015.

[9] C. Efstathiades, A. Belesiotis, D. Skoutas, and D. Pfoser. Similarity Search on Spatio-Textual Point Sets. In *EDBT*, pp. 329-340, March 2016.

[10] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, pp. 226-231, August 1996.

[11] W. Feng, C. Zhang, W. Zhang, J. Han, J. Wang, C. Aggarwal, and J. Huang. StreamCube: Hierarchical Spatio-temporal Hashtag Clustering for Event Exploration over the Twitter Stream. In *ICDE*, pp. 1561-1572, April 2015.

[12] C.S. Jensen, D. Lin, and B. Chin Ooi. Continuous Clustering of Moving Objects. *IEEE TKDE*, 19(9): 1161-1174, 2007.

[13] J. Jiang, H. Lu, B. Yang, and B. Cui. Finding Top-k Local Users in Geo-tagged Social Media Data. In *ICDE*, pp. 267-278, April 2015.

[14] T. Lappas, M.R. Vieira, D. Gunopulos, and V.J. Tsotras. On the Spatiotemporal Burstiness of Terms. *PVLDB*, 5(9): 836-847, May 2012.

[15] J. Li, D. Maier, K. Tufte, V. Papadimos, and P.A. Tucker. No Pane, No Gain: Efficient Evaluation of Sliding-Window Aggregates over Data Streams. *ACM SIGMOD Record*, 34(1): 39-44, March 2005.

[16] A. Magdy, L. Alarabi, S. Al-Harthi, M. Musleh, T.M. Ghanem, S. Ghani, and M.F. Mokbel. Taghreed: a System for Querying, Analyzing, and Visualizing Geotagged Microblogs. In *ACM SIGSPATIAL*, pp. 163-172, 2014.

[17] M. Mathioudakis, N. Bansal, and N. Koudas. Identifying, Attributing and Describing Spatial Bursts. *PVLDB*, 3(1): 1091-1102, 2010.

[18] K. Patroumpas and T. Sellis. Maintaining Consistent Results of Continuous Queries under Diverse Window Specifications. *Information Systems*, 36(1): 42-61, 2011.

[19] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *IW3C2*, pp. 851-860, April 2010.

[20] J. Shi, N. Mamoulis, D. Wu, and D.W. Cheung. Density-based Place Clustering in Geo-Social Networks. In *ACM SIGMOD*, pp. 99-110, 2014.

[21] A. Skovsgaard, D. Sidlauskas, and C.S. Jensen. Scalable Top-k Spatio-Temporal Term Querying. In *ICDE*, pp. 148-159, April 2014.

[22] W. Wang, J. Yang, and R. Muntz. STING: A Statistical Information Grid Approach to Spatial Data Mining. In *VLDB*, pp. 186-195, 1997.

[23] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: A Realtime Local-event Detection System based on Geolocation Information Propagated to Microblogs. In *ACM CIKM*, pp. 2541-2544, 2011.

[24] T. Zhang, R. Ramakrishan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *ACM SIGMOD*, pp. 103-114, June 1996.