EvenTweet: Online Localized Event Detection from Twitter

Hamed Abdelhaq, Christian Sengstock, and Michael Gertz
Institute of Computer Science
Heidelberg University, Germany
{hamed.abdelhaq, sengstock, gertz}@informatik.uni-heidelberg.de

ABSTRACT

Microblogging services such as Twitter, Facebook, and Foursquare have become major sources for information about real-world events. Most approaches that aim at extracting event information from such sources typically use the temporal context of messages. However, exploiting the location information of georeferenced messages, too, is important to detect localized events, such as public events or emergency situations. Users posting messages that are close to the location of an event serve as human sensors to describe an event. In this demonstration, we present a novel framework to detect localized events in real-time from a Twitter stream and to track the evolution of such events over time. For this, spatio-temporal characteristics of keywords are continuously extracted to identify meaningful candidates for event descriptions. Then, localized event information is extracted by clustering keywords according to their spatial similarity. To determine the most important events in a (recent) time frame, we introduce a scoring scheme for events. We demonstrate the functionality of our system, called EvenTweet, using a stream of tweets from Europe during the 2012 UEFA European Football Championship.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS, Data mining

Keywords

Event detection, localized events, online approach

1. INTRODUCTION

The proliferation and adoption of Web 2.0 and GPS-enabled technologies has led to a rapid increase in the amount of georeferenced and timestamped social media, such as messages in Twitter. This huge amount of data can be seen as an up-to-date source of event-related information [3]. Users participating in or observing an event are motivated to publish a relatively larger number of *event-relevant messages* and talk about it using their mobile devices, acting as *human sensors*. As a consequence, exploiting social media sources to extract relevant event information allows to convey important knowledge to people much faster than traditional news media [6].

Research has primarily focused on identifying and tracking events from this large amount of data using the temporal context of messages [1, 4, 8]. In addition, there is a number of efforts to exploit the spatial (location) information to detect *localized events*, i.e., events that are important within a small geographic area, e.g., public events or emergency situations. Chen et al. [2] exploit the spatio-temporal information of photos in Flickr to detect events. Their approach, however, operates only on a static data set and thus is not meant to work on a continuous stream of data. Lappas et al. [5] track the spatio-temporal burstiness of individual features (terms) in real-time with application to a context-aware document search framework. However, their approach is not trying to detect events by grouping related terms.

In the context of Twitter, only about 1% of tweets are georeferenced. Thus, Watanabe et al. [7] try to assign geo-coordinates to non-geotagged tweets to increase the chance of finding localized events. Then, they search for place names and count the number of key terms that co-ocurr with each place name. Nevertheless, their method fails to find localized events when no places are mentioned in the tweets.

In this demonstration, we present *EvenTweet*, a system to detect localized events from a stream of tweets in real-time. What distinguishes EvenTweet from other event detection systems, e.g., [2, 7], is that EvenTweet focuses on detecting localized events in real-time by adopting a continuous analysis of the most recent tweets within a time-based sliding window. Detected localized events are described by (1) a number of related keywords, and (2) an estimation of both the start time and the geographic location. Furthermore, EvenTweet not only detects localized events, but also tracks their evolution over time using a fine-grained temporal resolution. For this, we propose a scoring scheme that gives a score for each event, acting as an indicator of its significance over time.

Our system is not estimating geo-coordinates for non-geotagged tweets as done in [7]. However both geo- and non-geo-tagged tweets are used to identify words best describing events. Then, only geo-tagged tweets are used to estimate the spatial distribution of such words. By this, our system is able to identify localized events using a (possibly small) amount of geo-tagged tweets.

Proceedings of the VLDB Endowment, Vol. 6, No. 12

The remainder of the paper is structured as follows. In Section 2, we detail our methods to detect localized events from a stream of georeferenced tweets. Then, we briefly discuss the system architecture in Section 3 before outlining the demonstration scenarios in Section 4.

2. LOCALIZED EVENT DETECTION

We define an *event* as a phenomenon that stimulates people to post messages for a certain period of time. We restrict our approach to events that occur within a limited time period (recurring events are treated as unique events). Events might happen in a substantial part of geographic space (e.g., "Mother's Day") or within a small region. We call the latter events that have a small spatial extent *localized events* (e.g., festivals, road jams, football matches). Formally, a localized event is described as a tuple $le = (el, et, K)$; el is the event location represented as a small set of connected rectangular cells, et is the start time, and K is a set of words frequently published during the event time and at that location. In the following, we give some definitions and detail our online algorithm to extract localized events.

2.1 Online Detection

Each tweet $tw = (W, uid, l, t)$ consists of a set of words W, a user id uid, a geographic location $l = (lon, lat)$[1], and a timestamp t. We use a timeline that is divided into a sequence of equal-length time frames $(..., f_{c-1}, f_c)$, where f_c denotes the current time frame. Each time frame represents a short time interval during which tweets are posted. We use a time-based sliding window $win^k_{f_c}$ composed of k time frames and f_c as its end point. The detection procedure of EvenTweet is triggered every time a new time frame elapses.

Given a new time frame, the following general methodology, whose steps are discussed in more detail in the following sections, is used to extract localized events $le_1, \ldots, le_k$: (1) Extraction of words showing a bursty frequency in the current time frame (these words are called *keywords*), (2) selection of those keywords that have a local spatial distribution, which is estimated from corresponding observations collected during the sliding window, (3) clustering of selected keywords by their spatial signatures; as new time frames are evaluated, clusters can become less significant and thus will be removed, and (4) scoring of the clusters to show how likely they represent a localized event.

Temporal Keyword Extraction

Given a set of words W_c retrieved from the tweets published during the recent time frame f_c. We want to extract a subset $Y_c \subseteq W_c$ representing words that are likely to describe localized events. We call the set Y_c the event *keywords*.

The discrepancy paradigm [5] is used to extract keywords based on their burstiness. Assume that during time frame f_c, the number of users publishing tweets containing word w, normalized by the total number of users, is denoted by $u(w, c)$. In addition, $hist_w = (u(w, 1), u(w, 2), ..., u(w, m))$ is a fixed historical sequence of usage values for w collected before the current time frame f_c, such that $m < c$. The usage history of word w, $hist_w$, is used when the system needs to describe the normal (non-bursty) behavior of word w over previous time frames. The discrepancy paradigm measures the deviation between the word usage value $u(w, c)$ in the current time frame and an expected word usage baseline. The higher the deviation, the higher the burstiness degree.

We assume that the word usage baseline $b(w)$ is constant and estimated from $hist_w$. To model noise, we expect that the values of $hist_w$ 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 ($hist_w$) for word w.

The burstiness degree of a word w is the z-score defined as $b\_degree(w, c) := \frac{u(w,c)-b(w).\mu}{b(w).\sigma}$. We choose those words as keywords whose burstiness degree is larger than two standard deviations above the mean. Keywords observed for the first time will have $\mu = 0$ and $\sigma = 0$. For all keywords having $\sigma = 0$ (either because we have not observed the keyword yet or because we only observed the same counts for a keyword), we assume a prior noise level of $\sigma = u(w, c)/r$. That is, the initial noise level is $1/r$-th of the observed number of keywords. In our work, we use $r = 3$ such that a keyword is considered bursty the first time it occurs.

Spatial Keyword Identification

In this step, we utilize georeferenced tweets to select keywords that are likely to describe localized events. For this, we calculate a *spatial signature* for each keyword. A spatial signature is the spatial density distribution over the usage ratio of a keyword at a particular location during $win^k_{f_c}$. We define a regular grid G with a bandwidth a. The usage ratio of keyword k_i in a cell $g \in G$ is the number of users using the keyword in g, normalized by the total number of users in g. The density of a keyword k_i in cell g then is the usage ratio normalized over the usage ratios of k_i in all cells. We refer to this discrete spatial distribution of keyword k_i as S_i and call it the *spatial signature* of keyword k_i.

The entropy of the spatial signature S_i, denoted $H(S_i)$, is used to select local keywords. A high entropy means that the keyword is widely spread over space, a small entropy means that the keyword occurs only at a few locations. Because we are only interested in keywords that have a small entropy, we discard keywords with an entropy larger than a threshold ρ. The resulting set of keywords then is $Y_c = Y_c - \{k_j : H(S_j) > \rho\}$ such that $\rho \in [0, log(|G|)]$.

Keyword Clustering

The main two tasks in this step are (1) to find subsets of $Y_c \subseteq W_c$ that represent new localized events, and (2) to update the information about localized events that have already been detected earlier.

Keywords related to the same localized event tend to show some spatial proximity. This means that they have similar spatial signatures. Therefore, we use a single-pass clustering algorithm, similar to Birch [9], to group event keywords based on the cosine similarity of their spatial signatures. The centroid of each cluster is the average of the spatial signatures of its keywords. Using this clustering procedure, the spatial signature of each new keyword is compared to the centroids of existing clusters. The keyword is placed into the most similar cluster if the distance is within a threshold τ, which can be assigned a value from $[0, 1]$. Choosing a higher threshold means that only very similar keywords in terms of spatial signature are assigned to the same cluster. Otherwise, a new cluster is created for this keyword. During a localized event, the cluster receives the same bursty keywords several times until the event diminishes.

[1]In case of a georeferenced tweet.

Cluster Scoring

Due to the noisy nature and the increasing vocabulary size of tweets, the extracted keyword set is enormous and has many spurious keywords, which results in creating clusters related to no events. Thus, a scoring scheme is applied to identify and highlight clusters that refer to real-world, localized events. Such clusters are referred to as *event clusters*. After each new time frame, the state of a cluster is changed, and thus, its score is updated accordingly. Before discussing the cluster scoring scheme, we introduce some general properties describing the state of a cluster cl:

1. Start Time $cl.st$: It represents the time frame at which the cluster was created. We define the cluster life time $cl.lt$ as the number of time frames from the current time frame f_c back to $cl.st$, i.e., $cl.lt := f_c - cl.st + 1$.

2. Keyword Descriptors $cl.Y$: This is a set of tuples, $cl.Y = \{(k_i, o_i, e_i, b\_degree(k_i, e_i))\}_{i=1...h}$ where h is the number of keywords in cluster cl. Each tuple consists of the keyword, the number of times it was bursty and assigned to the cluster, the last time frame it was bursty at, and its last burstiness degree, respectively.

3. Spatial Coverage $cl.SC$: A vector representing a smoothened version of the spatial signatures of the keywords in cl. It describes the spatial extent of the corresponding event over space. The vector is the averaged spatial distribution of the spatial signatures $S_1, \ldots, S_h$ and is updated every time a keyword enters the cluster.

We assume that a cluster is likely to describe a localized event if its keywords (1) have high burstiness degrees, (2) are members of the cluster for a relatively long time, and (3) are recently assigned to it. To determine this, we first calculate individual scores for the keywords in a cluster cl. Given a keyword descriptor $(k_i, o_i, e_i, b\_degree(k_i, e_i)) \in cl.Y$, i.e., information about keyword k_i in cluster cl. The individual keyword score is then defined as:

$$k\_score(i, cl) := o_i \cdot b\_degree(k_i, e_i) \cdot prominence(k_i, cl) \quad (1)$$

with *prominence* being defined as:

$$prominence(k_i, cl) := \left(\frac{cl.o_i}{cl.lt}\right) \cdot \left(1 - \frac{f_c - cl.e_i}{cl.lt}\right) \quad (2)$$

The prominence takes values from [0,1]. The first factor indicates how many times k_i was clustered in cl relative to the cluster life time $cl.lt$. The second factor indicates the closeness of the last assignment of k_i in cluster cl to the current time frame f_c. If k_i is bursty at the current time frame f_c and assigned to cluster cl during the entire $cl.lt$, then $prominence(k_i, cl) = 1$.

Based on the individual keyword scores, the score of cluster cl is defined as:

$$score(cl) := \sum_{i=1}^{|cl.Y|} k\_score(i, cl) \quad (3)$$

The top-k scoring clusters are considered possible event clusters. If a cluster cl is recognized as an event cluster, the components of the localized event $le = (el, et, K)$ are given by the set of event keywords $K = cl.Y$ (ordered in decreasing order by their individual keyword scores), the event location $el = cl.SC$, and the event start time et as the time frame at which the cluster got the highest score.

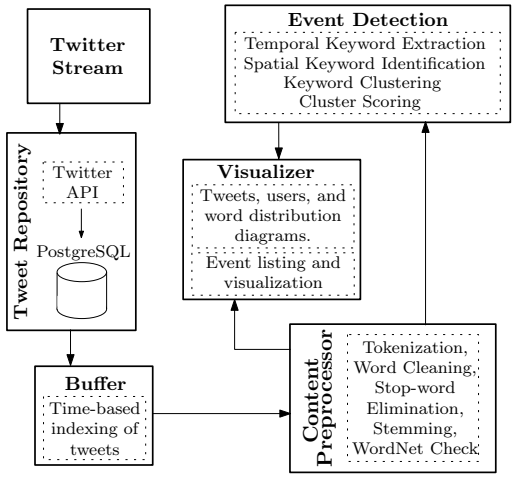

Figure 1: System overview of EvenTweet

3. SYSTEM OVERVIEW

Figure 1 shows the system overview of EvenTweet. It is implemented as a plugin for the JOSM[1] framework and consists of the following components:

- Tweets repository: Tweets are collected continuously using the Twitter API and stored in a database, which is accessed repeatedly to retrieve the most recent tweets collected during f_c. In addition, the system supports off-line event detection by allowing users to retrieve tweets for some specified time intervals.

- Buffer: EvenTweet keeps tweets published during $win_{f_c}^k$ in main memory. These tweets are indexed by time frames allowing for fast random access. Furthermore, the word usage history for a fixed period of time before f_c is kept in main memory to be accessed whenever a baseline for a certain word is required.

- Content Preprocessor: EvenTweet adopts a number of text preprocessing tasks to cope with the noisy nature of Twitter content. This includes stop-word removal, stemming, and WordNet dictionary lookups.

- Localized-Event Detector: The event detection procedure is triggered every new time frame. The detection procedure consists of the steps discussed in Section 2 using data from the buffer.

- Visualizer: This component provides users with a number of functions to explore tweets and word usage distributions. In particular, it allows for inspecting detected events and displaying them on a map.

4. DEMONSTRATION

In our demonstration we show the ability of EvenTweet to detect localized events, in particular, to detect the start time and the location of football matches taking place during the 2012 UEFA European Football Championship. We used the Twitter API to gather a stream of tweets, originating from Ukraine (bounding box: 23.99, 45.20, 40.51, 53.26), during the time interval 2012/6/22 00:00 - 2012/7/8 23:59, which resulted in a total of 67,368 tweets.

[1]http://josm.openstreetmap.de/

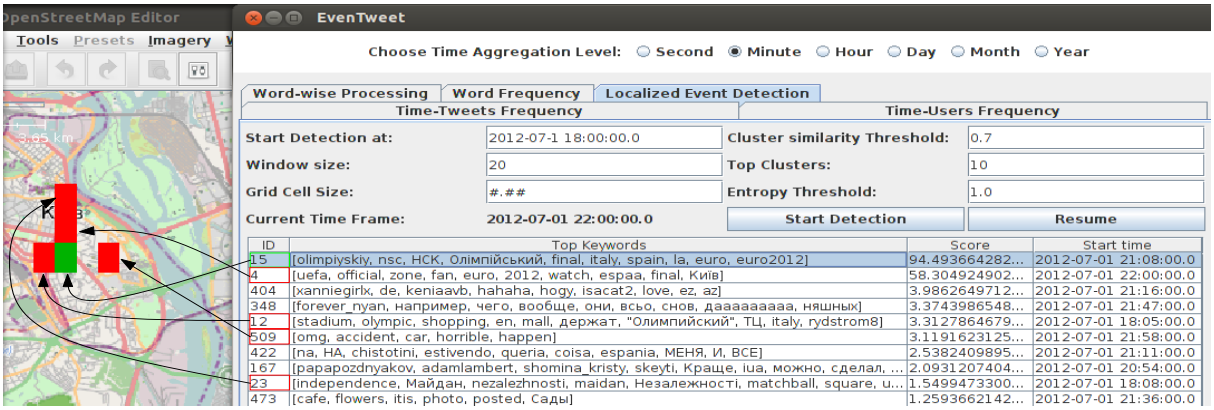

Figure 2: EvenTweet's interface to visualize detected localized events. A user can enter values for the different parameters and select a timestamp at which the real-time detection starts. Localized events, along with other associated information, are listed in descending order by their scores and updated once a new time frame elapses. Events selected by the user will be shown as a green cell on the map.

The baseline usage of words is estimated from historical data collected from 2012/6/22 00:00 to 2012/7/1 00:00. To find events having the scale of a stadium, we set the bandwidth a to 1.1 km for the spatial grid. To detect the start time of events with a resolution of a minute, we chose a minute interval for the time frames. After simulating the real time detection, the system was paused at timestamp 7/1 22:00. Figure 2 shows the detected events with the system started on 7/1 at 18:00. The event E15 (ID=15) has the top score and refers to the final match in the 2012 UEFA European Football Championship between Italy and Spain. The estimated start time is (21:08), which is close to the actual start time (20:45). Moreover, the associated keywords describe the detected match, e.g., olimpiyskiy (Olympic Stadium), italy, spain, final, nsc (National Sport Complex). E15 is centered at the stadium where the match took place.

Other detected events (E4 and E12) refer to the same match but at different locations and start times. The events close to each other are described by the same cluster using a lower cluster similarly threshold τ, as will be shown in our demo. In addition, we will show that the entropy threshold ρ is an important parameter, as it discards keywords describing global events. Choosing a higher threshold results in less aggressive filtering and will lead to a dominance of global temporal events other than localized events. As can be seen in Figure 2 the score reflects the importance of an event as described in Section 2. By changing the resolutions, the cluster similarity threshold, and/or the entropy, the score will either tend to favor more localized or more global events.

Moreover, we injected artificial tweets about a traffic accident into our dataset. The associated geo-coordinates are chosen to be close to an assumed accident location (long: 30.53, lat: 50.42). The first tweet was attached the timestamp (7/1 21:57), and the others the timestamp (7/1 21:58). To set their content, we asked 4 students to write what they will publish in case they see such an accident. Interestingly, EvenTweet detected the artificial accident event (E509), with a start time of (7/1 21:58), i.e., one minute after posting the first relevant tweet. The fact that EvenTweet considers both the burstiness of a keyword and how frequent it recurs over time allows it to detect events with a small number of relevant tweets. In our demo, we will show the detection of localized events using a real-time Twitter stream.

5. CONCLUSIONS AND ONGOING WORK

We outlined the key features of EvenTweet, a system to detect localized events from a stream of tweets. In the demonstration, we show the influence of the spatial and temporal resolution parameters as well as spatial similarity and entropy thresholds. We are currently improving the scoring and labeling of events, and we are also looking into estimating the end time of localized events.

6. REFERENCES

[1] H. Becker, M. Naaman, and L. Gravano. Beyond Trending Topics: Real-World Event Identification on Twitter. In *ICWSM*, 438–441, 2011.

[2] L. Chen and A. Roy. Event detection from Flickr data through wavelet-based spatial analysis. In *CIKM'09*, 523–532, 2009.

[3] M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 211–221, 2007.

[4] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *SIGIR*, 207–214, 2007.

[5] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras. On the spatiotemporal burstiness of terms. In *PVLDB*, 836–847, 2012.

[6] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes Twitter users: real-time event detection by social sensors. In *WWW*, 851–860, 2010.

[7] K. Watanabe, M. Ochi, M. Okabe, and R. Onai. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM '11*, 2541–2544, 2011.

[8] J. Weng and B.-S. Lee. Event detection in twitter. In *5th In. AAAI Conf. on Weblogs and Social Media*, 2011.

[9] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. In *Data Mining and Knowledge Discovery* 1(2):141–182, 1997.