Soc. Netw. Anal. Min. (2016) 6:89
DOI 10.1007/s13278-016-0400-7

ORIGINAL ARTICLE

Where has this tweet come from? Fast and fine-grained geolocalization of non-geotagged tweets

Pavlos Paraskevopoulos[1] · **Themis Palpanas**[2]

Received: 12 January 2016 / Revised: 17 September 2016 / Accepted: 20 September 2016 / Published online: 30 September 2016

Abstract The rise in the use of social networks in the recent years has resulted in an abundance of information on different aspects of everyday social activities that is available online, with the most prominent and timely source of such information being Twitter. This has resulted in a proliferation of tools and applications that can help end users and large-scale event organizers to better plan and manage their activities. In this process of analysis of the information originating from social networks, an important aspect is that of the geographic coordinates, i.e., geolocalization, of the relevant information, which is necessary for several applications (e.g., on trending venues, traffic jams). Unfortunately, only a very small percentage of the twitter posts are geotagged, which significantly restricts the applicability and utility of such applications. In this work, we address this problem by proposing a framework for geolocating tweets that are not geotagged. Our solution is general and estimates the location from which a post was generated by exploiting the similarities in the content between this post and a set of geotagged tweets, as well as their time-evolution characteristics. Contrary to previous approaches, our framework aims at providing accurate geolocation estimates at fine grain (i.e., within a city). The experimental evaluation with real data demonstrates the efficiency and effectiveness of our approach.

✉ Pavlos Paraskevopoulos
p.paraskevopoulos@unitn.it

Themis Palpanas
themis@mi.parisdescartes.fr

[1] Telecom Italia - SKIL, University of Trento, Trento, Italy

[2] Paris Descartes University, Paris, France

Keywords Geotag · Geolocation · Twitter · Social networks

1 Introduction

Several social networks have emerged during the last decade. Social networks, such as Twitter, Facebook and Google+, give users the opportunity to express themselves and report details about their everyday social activities. The combination of this behavior with the widespread use of mobile smartphones and tablets led to a very interesting phenomenon, where the activities reported within social networks are happening in real time, with individual users adding reports from several different locations (not just from their homes or workplaces).

The above observation means that we now have access to datasets containing important information for the better and more detailed understanding of social activities. To that effect, several studies (Tsytsarau and Palpanas 2012), including applications (Sakaki et al. 2010; Mathioudakis and Koudas 2010; Tsytsarau et al. 2010; Frias-Martinez et al. 2012; Balduini et al. 2013; Crooks et al. 2013; Balduini et al. 2014; Tsytsarau and Palpanas 2014; Zafarani and Liu 2015) and techniques (Tsytsarau et al. 2011, 2013, 2014; Paraskevopoulos et al. 2013), have been developed that analyze datasets created through the use of social networks, in order to provide benefits to end users, businesses, civil authorities, and scientists alike (Paraskevopoulos et al. 2016).

Note that several of these applications depend on the knowledge of the user location at the time of the posting. For example, this knowledge is necessary for applications that target to characterize an urban landscape, or to optimize urban planning (Frias-Martinez et al. 2012), to

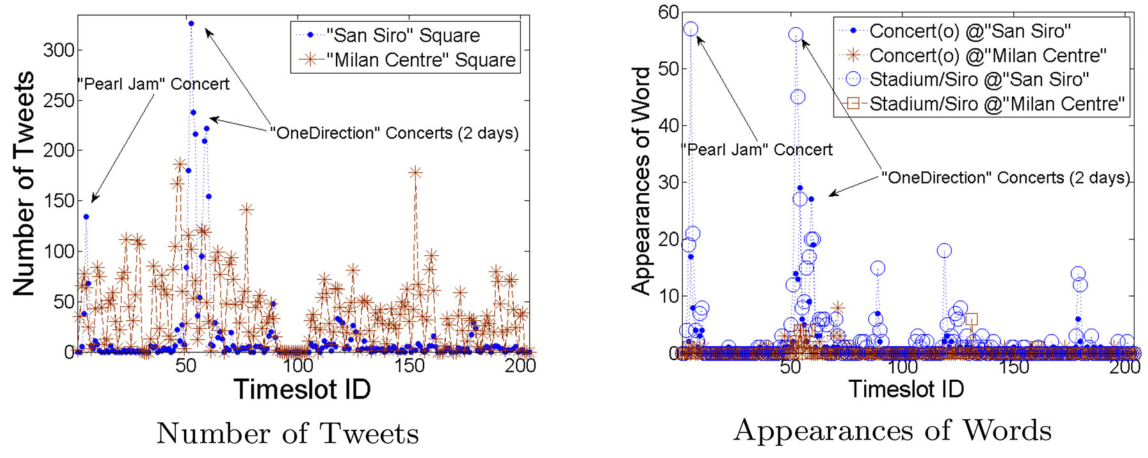


Fig. 1 Data generated from different neighborhoods (i.e., squares with side 1000 m) in Milan (Italy), for time intervals of 4 h, between June 20 and July 23, 2014

identify and report natural disasters, such as earthquakes (Sakaki et al. 2010; Crooks et al. 2013), and to monitor and track mobility and traffic (Balduini et al. 2013). Such applications, which represent an increasingly wide range of domains, are restricted to the use of geotagged data,[1] that is, posts in social networks containing the geographic coordinates of the user at the time of posting.

Evidently, the availability of geotagged data determines not only the possibility to use such applications, but also their quality performance characteristics: The more geotagged data posts are available, the better the quality of the results will be (more accurately: the higher the probability for being able to produce better quality results). Nevertheless, the availability of geotagged data is rather limited. In Twitter, which is the focus of our study, the number of geotagged tweets is a mere 1.5–3 % of the total number of tweets (Leetaru et al. 2013; Murdock 2011; Han et al. 2014). As a result, the amount of useful data for these applications to analyze is small, which in turn limits the utility of the applications.

In this study, we address this problem by describing a method for geolocalizing tweets that are non-geotagged. Even though previous works have recognized the importance and have studied this problem (Chang et al. 2012; Kinsella et al. 2011) (for a comprehensive discussion of this problem, refer to Han et al. (2014)), their goal was to produce a coarse-grained estimate of the location of a set of non-geotagged tweets (e.g., those originating from a single user). The algorithms they propose operate at the level of postal zip codes, cities, and geographic areas larger than cities. In contrast, we study this problem at a much finer granularity, providing location estimates for *individual* tweets first at the level of cities and then at the level of *city neighborhoods*, thus enabling a new range of applications that require detailed geolocalized data.

We illustrate and motivate some of these ideas in Fig. 1. Figure 1a depicts the number of tweets posted from the neighborhood in which the "*San Siro Stadium*" is located and from a neighborhood located in the center of the Milan (Italy), while Fig. 1b shows the number of appearances of the keywords *concert* (in English and Italian) and *stadium*/*siro* in these neighborhoods. As these graphs show, the "San Siro" geolocation exhibits an unusually high activity during the time intervals that coincide with the concerts that took place in this stadium. Furthermore, during these concerts, the words *concert*(*o*) and *stadium* / *siro* originate from the "San Siro" geolocation much more frequently than a random geolocation in the city.

There are two main challenges that emerge when the granularity level becomes fine: first, to maintain high accuracy despite the wider range of possible locations available to the prediction algorithm, and second, to achieve high time performance despite the increased size of the search space of the algorithm. The framework we describe for the fine-grained geolocalization of non-geotagged tweets is based on the careful evaluation of the similarities in the content between a new, non-geotagged tweet and a training set of geotagged tweets. The solutions we propose for this similarity evaluation make use of efficient-to-compute information retrieval and statistical measures, namely Tf-Idf among the tweet contents, and correlation among the time series representing the volume of tweets in different candidate locations. The advantages of these measures are that they can effectively capture the most significant pieces of information needed to solve the problem and that they have low time complexity.

[1] For the rest of this paper, we will use the terms *geotagged* and *geolocalized* interchangeably.

The contributions we make in this paper can be summarized as follows.

- We describe and define the problem of fine-grained geolocalization of non-geotagged tweets, which aims to operate on individual tweets, at the level of city neighborhoods. We argue that the efficient solution of this problem will enable a multitude of applications that require detailed location information.
- We propose a framework for the solution of the above problem, which is based on the content similarities of tweets, as well as their time-evolution characteristics. The solution we describe is general and essentially parameter free.
- Finally, we perform a detailed experimental evaluation of our approach, using real data from Twitter. The results demonstrate the efficiency and effectiveness of the proposed approach when compared to various alternatives.

The rest of the paper[2] is organized as follows. In Sect. 2, we present the related work. Section 3 formalizes the problem, and Sect. 4 describes our solution. We present our experimental evaluation in Sect. 5 and conclude in Sect. 6.

2 State of the art

Several works have studied the problem of geotagged tweet analysis. Balduini et al. (2013) studied the movement of people by analyzing geotagged tweets. The authors analyzed tweets originating from London and more precisely close to the Olympic stadium during the Olympic games. The results show that they could identify and track the movement of the crowd, especially during the opening ceremony. Some studies focus on the extraction of local events by analyzing the text in the tweets (Earle et al. 2012). A recent study describes an approach of how to use social media data (including Twitter), in order to better understand and manage city-scale events, part of which involves the extraction of location information for tweets (Balduini et al. 2014). However, this is only done for tweets that are already geotagged, or tweets that mention the venue and/or event of interest, for a predefined set of venues and events. On the other hand, our framework is able to estimate the location of tweets regardless of the use of hashtags, or any other entity reference in the content, leading to a general solution that is effective even in cases where of tweets referring to an unforeseen event (e.g., an accident), or tweets that do not explicitly mention the venue.

Abdelhaq et al. (2013) use both geotagged and non-geotagged tweets for identifying keywords that best describe events. Then they keep only the geotagged tweets in order to extract the local events. Twitter posts have also been studied in order to identify the location of earthquakes (Sakaki et al. 2010) or fine-grained details on user activities (such as drinking alcohol) (Hossain et al. 2016). We note that in all the above studies, the tweets that are analyzed are already geotagged. In contrast, our focus is on non-geotagged tweets.

The identification of points of interest (POIs) with temporal awareness is the focus of a recent study (Li and Sun 2014). The authors are analyzing tweets posted by Singaporean users, while using Foursquare check-ins referred in the tweets. Another study proposed a framework that can automatically recognize POIs by correlating geotagged tweets with geotagged data deriving from Flickr (Van Canneyt et al. 2012). The goal is to identify places, such as restaurants and hotels, that are not already part of databases such as LinkedGeoData, GeoNames, Google Places, or Foursquare. The combination of data from Foursquare and the logs of executed applications on a smartphone has been used in order to predict the next location of the users (Malmi et al. 2013).

The problem of using tweets in order to identify the location of a user or the place that an event took place has been studied in the past. The "who, where, what, when" attributes extracted from a user's profile can be used to create spatiotemporal profiles of users and ultimately lead to identification of mobility patterns (Yuan et al. 2013). Cheng et al. (2010) create location profiles based on idiomatic keywords and unique phrases mentioned in the tweets of users who have declared those locations as their origins.

The similarity between user profiles and location profiles has also been used in Chang et al. (2012). In this approach, they create user profiles for the active users and extract the keywords that are characteristic of specific locations (i.e., they usually appear in some location and not in the rest of locations). For the extraction of these keywords, they initially assign weights using the Geometric-Localness (GL) method and then prune them using a predefined keyword-weight threshold. This leads to a set of representative keywords for each location, which allows the algorithm to compute the probability that a given user comes from that location. A recent study evaluates the GL method and compares it to other methods that solve the same problem. The experimental evaluation shows that the GL method achieves the best results (Han et al. 2014).

Two studies that target to geotag tweets are presented in Eisenstein et al. (2010) and Paradesi (2011). These two

[2] This paper extends and improves on our earlier results (Paraskevopoulos and Palpanas 2015).

methods create chains of words that represent a location by using Latent Dirichlet Allocation (LDA) (Blei et al. 2003). The latter study takes in addition into consideration the location a user has recorded as their home location. A study that predicts both a user's location and the place a tweet was generated from is presented in Kinsella et al. (2011). In this study, the authors construct language models by using Bayesian inversion, achieving good results for the country- and state-level identification tasks. Finally, Serdyukov et al. (2009) presented a method for identifying the geolocation of photographs by using the textual annotations of these photographs.

Even though some of these studies are closely related to our work (e.g., Chang et al. 2012; Kinsella et al. 2011, which we further discuss in the experimental section evaluation section), we observe that they operate at a very different time and space scale. The profiles they create involve the tweets generated over a long period of time (up to several months), and the location that has to be estimated is the location of origin of the user, rather than the location from where a particular tweet was posted. Moreover, the space granularity used in these studies ranges from postal zip codes to areas larger than a city. On the contrary, in our work we predict the location of individual tweets, at the level of city neighborhoods.

Two studies that target to geotag unique tweets are presented in Ikawa et al. (2012) and Schulz et al. (2013). The first method trains a model using past messages associated with locations, by extracting keywords that are connected to this location. In the latter study, the authors develop a multi-indicator approach that combines information from the user's profile and the tweets' message for estimating both the location of a unique tweet and a user's residence location. The main difference to our approach is that these methods rely on users that post many tweets in a time interval t or on data from the user's profile. In contrast, we target to geotag tweets even from users that have never posted before or do not provide any profile data (such as their home location).

A recent survey presents methods relevant to location inference (Ajao et al. 2015).

3 Problem formulation

The problem we want to solve in this work is the estimation of the geographic location of individual, non-geotagged posts in social networks.

Problem 1 Given a set of geotagged posts $P_{t_j}^{l_1}, \ldots, P_{t_j}^{l_i}$, $t_1 \le t_j \le t_2$, where l_i is the location the post was generated from and t_j is the time interval during which the post was generated at, and a non-geotagged post Q_{t_q}, $t_1 \le t_q \le t_2$, we wish to identify the location l from which Q was generated.

The timestamps t_1 and t_2 represent the start and end times, respectively, of the time interval we are interested in.

In the context of this work, we concentrate on fine-grained location predictions: We wish to estimate the location of a post at the level of a city neighborhood (which is usually much smaller than a postal zip code). Furthermore, we focus on twitter posts, whose particular characteristics are the very small size (i.e., up to 140 characters long), and the heavy use of abbreviations and jargon language.

4 Proposed approach

In this section, we describe our solution to the problem of fine-grained geolocalization of non-geotagged tweets.

We provide a high-level description of our approach in Algorithm 1. Our method is based on the creation of vectors describing the Twitter activity in terms of important keywords for each geolocation we have data from, and for

Algorithm 1 Tweet Geotagging Algorithm

```
INPUT: A training set of timestamped and geotagged tweets, a timestamped query-tweet ($Q_t$) that is not geotagged.
OUTPUT: The most eligible candidate location.
1: for all $i \in$ {candidate geolocations: Geolocs} do    ▷ process training dataset, for all locations
2:     for all $t \in$ {time intervals} do    ▷ and for all time intervals
3:         $Doc_{i_t} \leftarrow$ all tweets in location $i$ at time interval $t$
4:         $kwVector_{i_t} \leftarrow$ create vector of $Doc_{i_t}$ keywords and their weights
5: $kwVector_{Q_t} \leftarrow$ create vector of $Q_t$ keywords and their weights    ▷ process non-geotagged tweet $Q_t$
6: $location \leftarrow argmax_{i \in Geolocs}$\{similarity between $kwVector_{i_t}$ and $kwVector_{Q_t}$\}    ▷ identify location of tweet $Q_t$
7: return $location$
```

the period of time we are interested in. The geolocations correspond to fine-grained spatial regions (in our study, they are squares with side length of 1000 m). The time intervals correspond to brief time segments, during which posts on the same or related topics may be observed (in our study, they are 4-h intervals). The vectors represent the weights of each keyword and are stored in *kwVector* for each geolocation and time interval. There are several ways to compute these weights: We consider the number of appearances of a keyword in a given geolocation and the significance of a keyword, measured using Tf-Idf, for a given geolocation and the entire dataset.

In order to identify the geolocation for a non-geotagged tweet, *Q*, we compute the similarity between the vector of *Q* and the vector of each candidate geolocation. When calculating this similarity, we can additionally take into account the correlation between the local and the global activity time series, i.e., the evolution over time of the number of tweets in a given geolocation and all the geolocations, respectively. Finally, the algorithm returns the geolocation with the highest similarity value.

In the following sections, we elaborate on the methods discussed above.

4.1 Grouping the posts and extracting important keywords

We start by processing the training set of geotagged posts. We group these posts according to the geolocation that they were generated from, and the time interval they belong to. After this grouping step, we calculate the concordance of the keywords in each group: the dictionary containing the number of appearances of each keyword in a geolocation. At the end, we have for each geolocation and time interval a vector of the important keywords, along with the corresponding weights. We call the algorithm that uses this method for generating the keyword vectors *TG (Tweet Geotagging)*.

We observe that concordance is a simple measure that only accounts for the frequencies of keywords, but fails to take into account their relative significance. Therefore, we also employ the Tf-Idf model: $idf_{keyword} = \log(\frac{n}{k})$, where n is the number of documents, k is the number of documents that keyword appears in, and $tfidf_{i,keyword} = \frac{count}{l} * idf_{keyword}$, where l is the total number of keywords in document i. Using Tf-Idf, we can calculate the significance of each keyword in our training dataset (according to the former equation above) and set the weight for a keyword in some geolocation, depending on the number of its appearances at this geolocation (according to the latter equation). This method leads to high weights for the keywords that appear at a small number of geolocations. As a final step, we sort the keywords according to their weight and prune the keywords with low weights, and therefore only keep the significant keywords for each geolocation, which correspond to the keywords that best characterize the activity of the given geolocation at a particular time interval. We call the algorithm that uses this method for generating the keyword vectors *TG-TI (Tweet Geotagging Tf-Idf)*.

In order to create the keyword vector for the non-geotagged tweet, *Q*, we wish to geolocalize, we follow the same process as before.

4.2 Similarity calculation and best match extraction

Our next target is to calculate the similarity between the keyword vector of *Q* and the keyword vector of each one of the candidate geolocations.

We follow the steps presented in Algorithm 2. The magnitude, *mag*, is the Euclidean Norm, computed over all the keywords that appear in the vector. We calculate the magnitude of the *Q* vector, mag_{Q_t}, and of each one of the

Algorithm 2 Similarity Calculation and Probability Extraction

```
1: procedure VectorSim(vectors for $Q_t$ and candidate geolocations)
2:     $mag_{Q_t} \leftarrow \sqrt{\sum_{\forall j \in kwVector_{Q_t}} kwVector_{Q_t}[j]^2}$     ▷ Extract the magnitude of the Q-tweet kwVector ($Q_t$)
3:     for all $i \in Geolocs$ do     ▷ For every Candidate Location (CL)
4:         $mag_{i_t} \leftarrow \sqrt{\sum_{\forall j \in kwVector_{i_t}} kwVector_{i_t}[j]^2}$     ▷ extract the magnitude of its kwVector
5:         $Sim_{i_t,Q_t} \leftarrow \frac{\sum_j kwVector_{i_t}[j] * kwVector_{Q_t}[j]}{mag_{i_t} * mag_{Q_t}}, \forall j \in kwVector_{Q_t} \cap kwVector_{i_t}$     ▷ Calculate the similarity between $Q_t$ and CL
6:     for all $i \in Geolocs$ do     ▷ Get the probability distribution
7:         $Prob_{i_t,Q_t} \leftarrow \frac{Sim_{i_t,Q_t}}{\sum Sim_{Q_t}}$
8:     SortDescending $Prob_{i_t,Q_t}$
9: return $i$ with highest $Prob_{i_t,Q_t}$
```

candidate geolocations i, mag_{i_t}, for a given time interval t. We denote with $kwVector[j]$ the weight of the jth term of the vector. The similarity is computed using the formula shown in line 5 (over all the keywords that appear in both the vector Q and the vector of the geolocation i). The algorithm stores in a sorted list the similarity values for each candidate geolocation. It then normalizes these values over the sum of all similarities, giving us the probability that each candidate geolocation produced Q. Transforming these values into a probability distribution gives us more flexibility: For example, as we discuss next, we can readily combine this similarity measure with similarities computed using other methods. Furthermore, we can use the probability values in order to produce geolocation predictions only in the cases where we are confident (i.e., these probabilities are high). At the end, the algorithm returns the geolocation(s) with the highest probability(ies).

In our approach, this similarity calculation happens in two phases (using for both the same general method presented above). First, we determine the city with the highest probability for having generated Q, and then the neighborhood (i.e., square with side 1000 m) within that city, with the highest probability. This solution has the added benefit that it can be effective even in the presence of small training datasets (i.e., few geotagged posts), which would not normally be adequate to directly train models with a very large number of candidate geolocations, as in our problem.

Therefore, when we change granularity, from the city to the neighborhood level, we add one more step to our method: $Prob_{i_t,Q_t} = Prob_{city_{j_t},Q_t} * Prob_{j_t,Q_t}$, where $city_{j_t}$ ranges over all the candidate cities, and j_t ranges over all the candidate neighborhoods within a city, $city_{j_t}$. The probability that a candidate neighborhood in a specific city ($Prob_{i_t,Q_t}$) is the correct geolocation for Q_t is computed by multiplying the probability that a candidate city is the correct one ($Prob_{city_{j_t},Q_t}$) by the probability that a given neighborhood within that city is the correct location ($Prob_{j_t,Q_t}$).

4.3 Similarity based on correlation of activity time series

The similarity measure discussed earlier is based entirely on the contents of the relevant posts, but ignores other useful characteristics of the data. In what follows, we describe a method that exploits the time-evolution behavior in order to derive an additional similarity measure.

This method is based on the activity time series, which record the number of posts generated by a given geolocation over time. We call these series *local activity* time series. We also compute the *global activity* time series, where we record the sum of the number of posts for all geolocations over time. The similarity is then expressed as the correlation value between the local activities of a candidate geolocation with the global activity. The intuition is that posts about an important event will significantly change the local activity and influence in the same way the global activity.

A straightforward idea is to compute the Pearson's correlation between the local and the global activities. Since we are only interested in similar behavior between local and global activity, we can only keep the positive correlations and then normalize them over the sum of all correlation values to produce a probability distribution. More specifically, we can construct the global activity time series, Gts_i, for a coarse-grained geolocation (e.g., a city i), as well as the local activity time series, Lts_j, for all the fine-grained geolocations within the coarse-grained one (e.g., the neighborhoods j inside the city i). Finally, we can compute the correlation between these time series using the Pearson's correlation.

The above idea proved to be somewhat useful, but with limited benefits [refer to algorithms *TG-C* and *TG-TI-C*, described in Paraskevopoulos and Palpanas (2015)]. The reason is that this method employs the correlation measure irrespective of the trend exhibited by the local and global activities. For example, these activities can be positively correlated, but have a negative trend (i.e., activity is diminishing). Evidently, in such cases, the correlation does not help and should not be taken into account.

We now describe a new technique that addresses this problem. More specifically, we consider a location as a candidate location only if both the local and the global activities increase. As we demonstrate later, this modification on the usage of the correlation measure leads to a significantly better result.

This new correlation-based technique is shown in Algorithm 3.

Initially, we construct the global activity time series, Gts_i, for a coarse-grained geolocation (e.g., a city g), as well as the local activity time series, Lts_{CL}, for all the fine-grained geolocations within the coarse-grained one (e.g., the neighborhoods j inside the city i). Since we are only interested in similar behavior between local and global activities only in the case where we have an increasing trend, we use the linear regression line in order to test this trend. In particular, we use the λ parameter of the equation representing the linear regression line, $y = \lambda * x + b$ (refer to line 4). If λ is positive, we assume that the time series has a positive slope (lines 5–6 and 12–13). In this process, we use smaller sliding sub-windows of size n / 2 (lines 3 and 10), sliding them across the original window. As a result, we have a sub-window that slides n / 2 times on the

Algorithm 3 Activity Correlation

```
1: procedure CorrelationSim(global $Gts$ and local $Lts_{CL}$ activity time series, threshold $th_{LR}$, Candidate Locations $CL$, Window time intervals $[t_1, t_2]$)
2:     $counter_g \leftarrow 0$
3:     for all $subWindow_i \in Window$ do        ▷ For how many subWindows Global time-series have positive $\lambda$
4:         $\lambda_g \leftarrow \frac{\Sigma((x-\bar{x})(y-\bar{y}))}{\Sigma(x-\bar{x})^2\Sigma(y-\bar{y})^2}$
5:         if $\lambda_g \geq 0$ then
6:             $counter_g \leftarrow counter_g + 1$
7:     if $counter_g > th_{LR}$ then    ▷ If Global time-series have at least $th_{LR}$ subWindows with positive $\lambda$
8:         for all $loc \in CL$ do    ▷ check Local time-series of all Candidate Locations ($loc$)
9:             $counter_{loc} \leftarrow 0$
10:            for all $subWindow_i \in Window$ do
11:                $\lambda_{loc_t} \leftarrow \frac{\Sigma((x-\bar{x})(y-\bar{y}))}{\Sigma(x-\bar{x})^2\Sigma(y-\bar{y})^2}$
12:                if $\lambda_{loc_t} \geq 0$ then
13:                    $counter_{loc} \leftarrow counter_{loc} + 1$
14:            $corr_{g,loc} \leftarrow \frac{\Sigma_{t=t_1}^{t_2}(Gts_{g_t}-\bar{Gts}_g)(Lts_{loc_t}-\bar{Lts}_{loc})}{\sqrt{\Sigma_{t=t_1}^{t_2}(Gts_{g_t}-\bar{Gts}_g)^2\Sigma_{t=t_1}^{t_2}(Lts_{loc_t}-\bar{Lts}_{loc})^2}} + 1$    ▷ Calculate the correlation between
15:                                            ▷ the time-series of the $loc$ and the global time-series
16:            if $counter_{loc} > th_{LR}$ then        ▷ Check if $loc$ exceeds the $th_{LR}$
17:                Assign to loc its correlation value, and True (for exceeding the $th_{LR}$ threshold)
18:            else
19:                Assign to loc its correlation value, and False (for not exceeding the $th_{LR}$ threshold)
20:    else
21:        Assign to all loc in $CL$ the values 0 (for the correlation) and False
22: return $CL$
```

original n-timeslot window, counting the number of the slides that result in positive linear regressions for both time series describing the local and the global activities.

After having calculated all the λ for each candidate locations, we calculate the Pearson correlation between the time series describing the local and the global activities and add 1 to this value, in order to shift the range of values between [0,2] (line 14). This has the desirable effect that we avoid negative similarities (that would result from negative correlations). Note that candidate locations that correspond to positive correlation receive a bonus (they get multiplied by a number in the range (1,2]), while those that correspond to a negative correlation get penalized [they get multiplied by a number in [0,1)]. Finally, we set a threshold th_{LR}, and we check whether the number of the sliding windows for each location that have positive λ is greater than th_{LR} (line 7 and 16).

If the number of the sliding sub-windows that have positive λ exceeds th_{LR}, then this location is considered as a candidate location, and we assign to the location its correlation and the value *True* for exceeding the threshold (line 17); otherwise, we assign to the location its correlation and the value *False* (line 19). Finally, the algorithm returns the final set of candidate locations, CL, which includes for each location its correlation value and the attribute that indicates whether the location exceeds the th_{LR} threshold (line 22).

We can then combine this method with the TG algorithm, by multiplying the two similarity measures (concordance similarity and correlation), to obtain the *TG-CLR (Tweet Geotagging with activity Correlation with Linear Regression)*. When we do the same with the TG-TI algorithm, we get the *TG-TI-CLR (Tweet Geotagging with Tf-Idf and activity Correlation with Linear regression)* algorithm. If the candidate location that has the greatest similarity with the non-geotagged tweet Q does not exceed the th_{LR} threshold (i.e., it has been assigned the value *False*), then we do not match Q to any location.

4.4 Sliding windows

We observe that previous methods use all past data in order to build their models. Methods such as Kinsella et al. (2011) and Chang et al. (2012) start building their models taking into consideration all available data. However, this may lead to situations where some local events may be mishandled. For example, consider the case, where a concert takes place in a city, followed by a second concert the following day. Then, a model that is based on all the data (and in the absence of specific and detailed keywords) is likely to assign the tweets relevant to the second concert to the location of the first concert, for which more data are available.

In order to avoid similar problems, we can use a tumbling window model (Paraskevopoulos and Palpanas 2015). Although this helps to address the problem mentioned above, tumbling windows may still mis-assign tweets that are generated at the beginning, or at the end of the window, and are connected to an event that is outside the window period.

A better idea is to use sliding windows, which we exploit in this work. In this case, a particular timeslot can be part of $n - 1$ windows, where n is the length of the window. If a timeslot is at the beginning of an event (the latest in the window), the new timeslots to be inserted later are going to be more relevant. As a result, the timeslot is going to be in $n - 1$ windows, the majority of which will be relevant.

Using the sliding window idea, we can now take advantage of the already extracted models of each location and incrementally update them for every slide, reducing dramatically the time needed for the contraction of the keyword vectors. In order to achieve this, we do not recalculate the concordance of each word for each location across the window. Instead, we extract the concordance across the window only for the first model created, and for every slide, we update the concordances of each word by subtracting the concordance of the words in the data removed and adding those in the data added to our dataset. We can see the steps of the incremental update of the vectors in Algorithm 4.

Furthermore, due to the incremental update that we achieve at our concordance *kwVectors*, we prove that our method can be applied in streaming manner. Unfortunately, the incremental update is not straight applicable on the Tf-Idf *kwVectors*, but still Tf-Idf methods get advantage on the incremental update of the concordances.

5 Experimental evaluation

5.1 Experimental setup

We performed the experiments on a server running on Ubuntu 14.04.2 LTS, with 64 GB RAM, and an Intel(R) Xeon(R) CPU E5506 @ 2.13 GHz processor. For the implementation of our methods and the reimplementation of the QL and KL, we used Python 2.7.

5.2 Datasets

For the evaluation of our approach, we use 3 datasets containing geotagged[3] posts from Twitter, generated in Italy, Germany, and the Netherlands. In particular, we have data from 6 of the largest Italian cities, namely Rome, Milan, Naples, Bologna, Venice, and Turin, and from the capital of Germany, Berlin, and the capital of Netherlands, Amsterdam. The tweets from Italy were generated between June 20 and July 23, 2014, while the tweets from Germany and the Netherlands were generated between August 10 and September 11, 2014. The granularity of the neighborhood level we use for every city is a square with side of 1000 m. The number of tweets is 543.295 for Italy (219.681 originated from Rome, 137.622 from Milan, 60.065 from Naples, 49.434 from Bologna, 46.982 from Turin, and 29.511 from Venice), 77.179 for Berlin and 136.189 for Amsterdam. The time windows we use have a duration of 4 h (which can effectively capture an important event, as well as the start and the aftermath of this event), while also keeping the detailed aggregated information for every 15-min time interval. As mentioned in Sect. 4.4, we use the sliding window model. We experimented sliding the window by 1 and 2 time intervals, getting almost the same results; thus, we chose to slide our window by 2 time intervals per slide (30 min), which led to faster execution times. Finally, the default grid we use in this study is 20 by 20 squares.

Algorithm 4 Incremental Update of kwVector

```
1: procedure UPDATE OF KWVECTOR(all $kwVectors_t$,geotagged tweets from location $i$
   for time intervals $t-1$ and $t+1$)
2:     for all $kwVector_{i_t} \in \{kwVectors_t\}$ do
3:         for all $word \in \{kwVector_{i_t}\}$ do
4:             $conc_{i_{t-1}} \leftarrow$ concordance in $i$ at $t-1$
5:             $conc_{i_{t+1}} \leftarrow$ concordance in $i$ at $t+1$
6:             $conc_i \leftarrow conc_{i_t} - conc_{i_{t-1}} + conc_{i_{t+1}}$
7: return $kwVectors_t$
```

[3] Earlier studies have shown that techniques and models built for geotagged data indeed generalize to non-geotagged data, since geotagged and non-geotagged tweets have similar data characteristics (Han et al. 2014).

5.3 Algorithms

We experimentally evaluate the six algorithms we described in Sect. 4, namely TG, TG-TI, TG-C, TG-TI-C, TG-CLR, and TG-TI-CLR (the last two only for the neighborhood level). As baselines, we implemented the QL and KL methods (Kinsella et al. 2011), which aim to solve a similar problem. In order to choose the value for the μ parameter, we followed the same methodology as in the original paper (Kinsella et al. 2011): We experimented with several values for the μ parameter, in the range [100, 10,000]), and verified that $\mu = 10,000$ gave the best results in our setting, as well.

5.4 Evaluation measures

We study the time performance, as well as the effectiveness of each approach using the precision and recall measures: $Precision = \frac{cgTweets}{gTweets}$ and $Recall = \frac{cgTweets}{aTweets}$, where *cgTweets* is the number of the correctly geolocalized tweets, *gTweets* is the number of tweets we geolocalized, and *aTweets* is the number of all tweets in the test set. In the case of the city level, where we predict the geolocation for all the tweets in the test set, the above precision and recall measures coincide, and we use the term *accuracy* instead. For the neighborhood level though, we do not predict the geolocation of tweets for which all our candidate locations have a similarity of 0. Thus, we report results for both precision and recall. We also report the balanced F1 measure, $F1 = 2 * \frac{Precision*Recall}{Precision+Recall}$. Following previous work (Kinsella et al. 2011), we report the results when we consider the top-1 (@Top1), top-3 (@Top3), and top-5 (@Top5; only for neighborhood level) predicted geolocations, as well as the results when considering as correct the prediction of the exact geolocation (@0-Step), or of any geolocation at distance 1 (@1-Step; exact and its eight immediate neighbors), or 2 (@2-Step; exact and its 24 closest neighbors) from the exact. In all our experiments, we randomly divided the dataset into 80 % training and 20 % testing, repeated each experiment 30 times, and reported the mean values in the results.

5.5 City-level results

We start our analysis by running our method on city level. We extract the geotagged tweets from the 6 cities, removing the duplicated posts in order to avoid spam. We record the activity every 15 min, and we consider time intervals of 4 h, leading to 181 timeslots (due to technical problems, some of the timeslots were empty, and we excluded those from our analysis).

In this case, we extracted the similarities between the test tweets and the 6 cities, and we also evaluated our approach using the correlation of the activity time series (refer to Sect. 4.3): We use the Pearson's correlation between the activity time series of the 6 cities and the activity time series of Italy. The results (@Top1 and @0-Step) are presented in Fig. 2a. As we can see in this plot, the accuracy for the city level is increased compared to the accuracy before the correlation. More precisely, we get the maximum number of matches in all four cases when we keep 100 % of the keywords. The accuracy of TG and TG-C is almost identical, at 45 %. For TG-TI we get 58 % accuracy, while when using TG-TI-C we get 59 % (though, our *t* test analysis revealed that this difference is not statistically significant). After further analyzing the results of these two algorithms, we found that for 134 windows TG-TI-C has better accuracy, for 4 windows TG-TI and TG-TI-C have the same accuracy, and for the rest 43 windows TG-TI performs better. We note that the accuracy of the random algorithm is 17 %.

After evaluating the algorithms using the most similar candidate geolocation (@Top1), we also evaluated them using the 3 most similar candidates (@Top3). As we can see in Fig. 2b, when using only a small percentage of the keywords we get better results with the TG and TG-C algorithms. In contrast, when we use more than 70 % of the keywords, the Tf-Idf-based algorithms, TG-TI and TG-TI-C, result in better accuracy. The accuracy is increasing when the percentage of the keywords used increases.

5.6 Neighborhood-level results

In this subsection, we present the results for the neighborhood-level evaluation, for which we used data from four different European cities: Milan, Rome, Berlin and Amsterdam. As we have already mentioned, we created a grid of 400 squares (20 by 20) for each city. For the city of Rome, we additionally ran some experiments using a grid of 900 squares (30 by 30).

5.7 Setting the parameters

We first identify the best threshold to use for the LR parameter. As we mentioned at the beginning of this section, we use a window of 16 timeslots and sub-windows of size $n_{sub-window} = n_{window}/2 = 8$ (refer to Sect. 4.3). Furthermore, the maximum LR equals to the number of slides, which is 8, as well. We experimented by setting the LR threshold equal to $\{1, 2, 4, 6\}$ and depicted the results in Fig. 3 (precision and recall for algorithms not using Tf-Idf), Fig. 4 (precision and recall for algorithms using Tf-Idf), and Fig. 5 (F1 measure for all algorithms). For

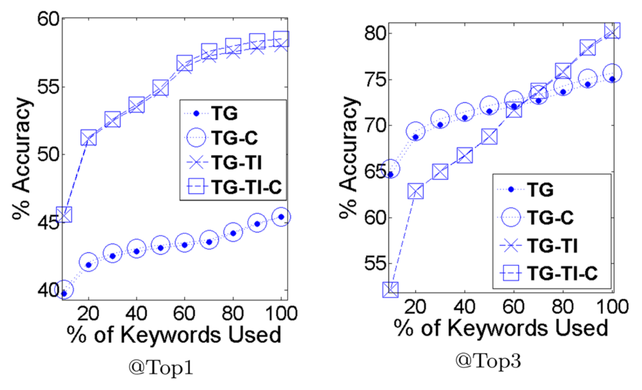


Fig. 2 Accuracy for city level when using TG, TG-C, TG-TI, and TG-TI-C (@0-Step)

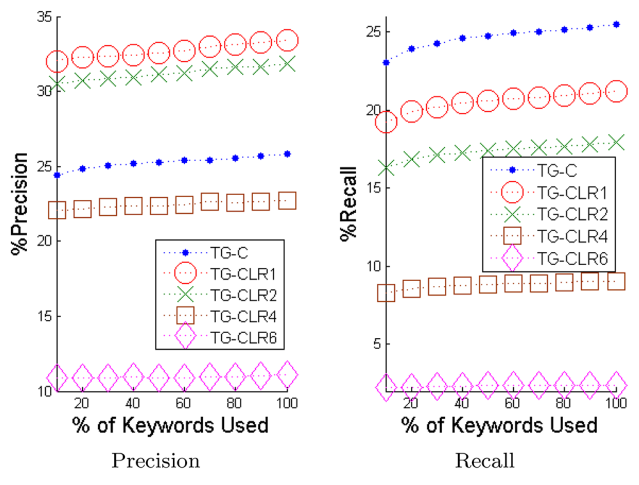


Fig. 3 TG-CLR for different LR parameters (@Top1 and @0-Step)

brevity, we only report the results for the city of Milan; results for the other cities are similar.

In this experiment, we had 3264 15-min timeslots, resulting in 1624 window slides. For each method, we extracted the mean precision, recall and F1 scores among all windows, while varying the percentage of the keywords used. We observe that the best mean precision is 48 %, which is achieved by TG-TI-CLR1 when using 100 % of the keywords (Fig. 4a), while the maximum recall for this method is 32 %, when using 40 % of the keywords (Fig. 4b). Note that the same method without the use of the trends, that is, TG-TI-C, has maximum precision and recall 40 and 39 %, respectively. Regarding the TG-CLR1 algorithm, we get maximum precision 33 % and maximum recall 21 %, both when using 100 % of the keywords. Due to these, and after finding out that the F1 score of the *CLR*1 methods is not too different compared to those not using linear regression, we concluded that for the rest of the experimental part, we are going to use only the *CLR*1 methods.

5.8 Evaluating the correlation-based methods

In the following experiments, we compare the CLR methods to those that do not use correlation. In Fig. 6, we present the mean precision and recall that our algorithms have for the city of Milan among all windows, when varying the percentage of the keywords used. As before, we only consider the first answer given by each algorithm (i.e., @Top1). The best precision is 48 % and is achieved by TG-TI-CLR1 using 100 % of the keywords. The maximum recall is 38 % achieved by TG-TI when using 30 % of the keywords. According to the F1 measure, TG-TI achieves its best using 30 % of the keywords, with F1 equal to 39 %. TG-TI-CLR1 achieves best F1 score 3 % when using 50 % of the keywords. The second best precision is 39 %, achieved by TG-TI when using 30 %. The best precision achieved by TG is 27 %, while its best recall is 26 %. TG-CLR1 reached up to 33 % precision and 21 % recall (both achieved when using 100 % of the keywords). The mean accuracy of the random algorithm, which was choosing one square at random only among that had data at the train datasets, was less than 2 %.

We note that the best precision is always observed when we use the TG-TI-CLR1 algorithm. This means that the correlation between the city and square activities is beneficial, when using the linear regression parameter that prunes activities with negative trends. As a result, we do not estimate the location of tweets that would probably be wrongly predicted, leading to a small penalty in recall, but increased precision.

We now report the results of the same experiment for the cities of Rome (in Fig. 7), Berlin (in Fig. 8), and Amsterdam (in Fig. 9). The best precision we observed for the city of Rome was 48 % and was achieved by TG-TI-CLR1, using 100 % of keywords, while the best recall was achieved when using TG-TI method, using 40 % of keywords. The same methods also resulted in the highest precision and recall for Berlin. In particular, TG-TI-CLR1 achieved a precision of 58 %, for a recall of 40 %. The best recall for Berlin was 47 %, achieved by TG-TI, which also led to the second best precision, 51 %. Regarding the city of Amsterdam, we achieve the highest precision of 44 % with TG-TI-CLR1, while the best recall of 38 % is achieved by TG-TI.

The results show that the behavior of the algorithms is similar across cities, while their relative performance remains the same. An interesting observation is the fact that the precision and recall for Berlin are much higher than the rest of the cities. This is due to the distribution of the keywords among the squares, which resulted in more representative keyword sets for each square.

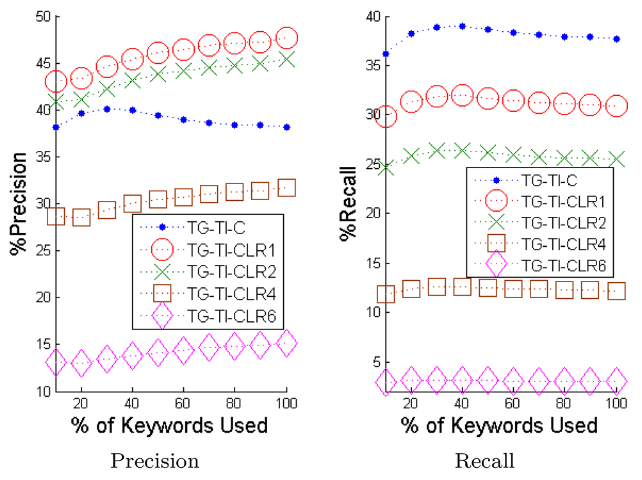


Fig. 4 TG-TI-CLR for different LR parameters (@Top1 and @0-Step)

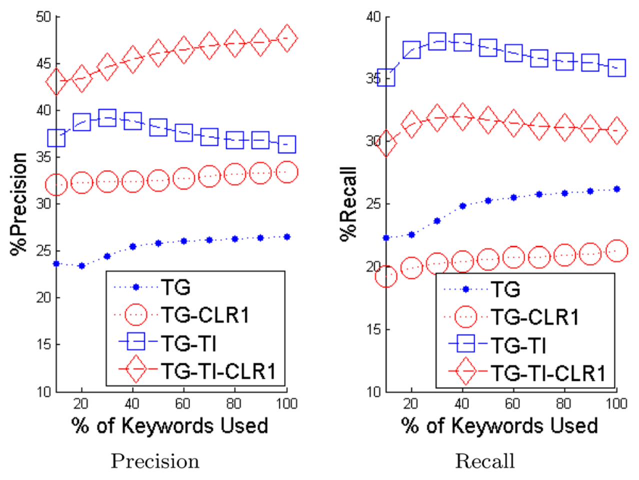


Fig. 6 Trade-off between precision and recall for neighborhood level (Milan, @Top1 and @0-Step)

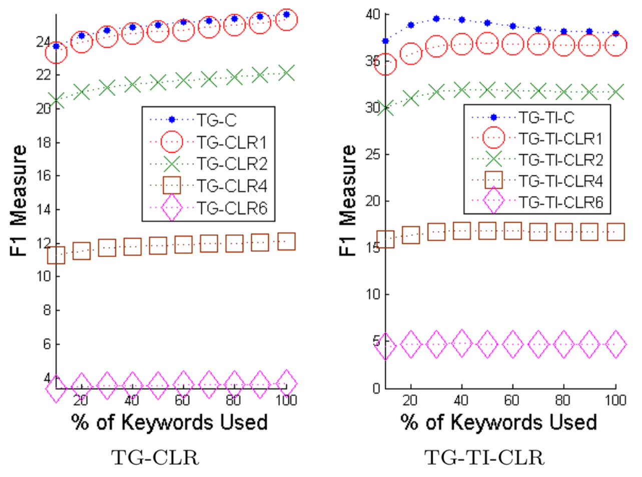


Fig. 5 F1 for TG-CLR and TG-TI-CLR for different LR parameters (@Top1 and @0-Step)

5.9 Comparing to baselines

In this set of experiments, we compare our approach to the QL and KL baseline algorithms. We use the same spatial and temporal granularities for all algorithms. Similarly to our methods, we only consider tweets for which there exists at least one candidate location with similarity greater than 0. The results of this comparison are illustrated in Fig. 10a.

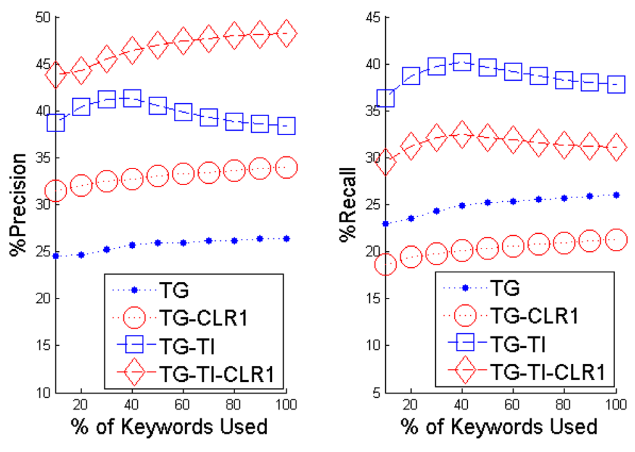
We observe that TG-TI-CLR1 achieves up to 18 % better recall than the QL algorithm[4] and up to 22 % better F1 score. This difference in performance can be explained by the different focus of the QL algorithm, which was developed to operate at much bigger spatial (in the order of zip codes, or cities) and temporal granularities (in the order of weeks or months) (Kinsella et al. 2011). We also note that (for the same reasons) the results between QL and KL are almost the same. Therefore, in our plots, we only report the F1 score for QL.

In terms of time performance, we measured the mean execution time needed per 4-h window for the entire process: training the models and extracting the similarities between the query tweets and the candidate locations. Figure 10b depicts the execution time needed for each algorithm.

As we can see in the graph, TG is the fastest algorithm. This is natural, since this algorithm does not spend time calculating the Tf-Idf, the correlations, or the linear regressions. The QL algorithm has a consistently high execution time of around 90 s, independent of the number of keywords considered. TG-TI-CLR1 performs in the middle. The interesting point is that although this algorithm has to calculate the Tf-Idf, the correlations and the linear regressions, the total time needed for each square, when using 10–70 % of the keywords is smaller than the time needed for TG-CLR1. The reason is the search space pruning. When compared to TG-CLR1, the TG-TI-CLR1 algorithm prunes stopwords and, thus, eliminates the candidate locations that do not share any keyword with the tweet under examination. We also observe that when TG-TI-CLR1 achieves its best F1 score, i.e., when using 50 % of the keywords, it is significantly faster than the QL algorithm.

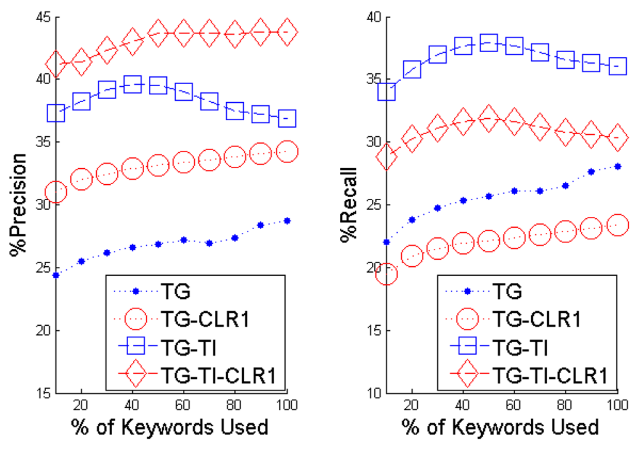
Finally, we note that the KL algorithm performs very similar to QL, but requiring at all cases a bit higher time when compared to QL (around 0.8 secs more).

[4] We note that the QL results reported here are much better than those reported in our earlier study (Paraskevopoulos and Palpanas 2015). This is due to the different experimental setup (i.e., sliding windows) that we now use for all algorithms, which resulted in an increased number of windows with a high number of tweets, leading to higher execution times and better models.



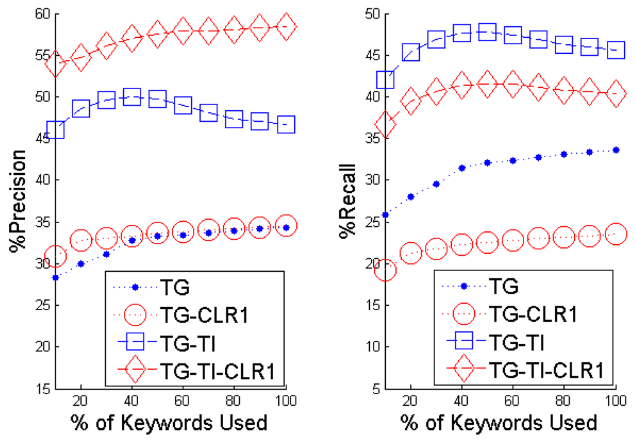
Precision Recall

Fig. 7 Precision and recall for the city of Rome (@Top1 and @0-Step)



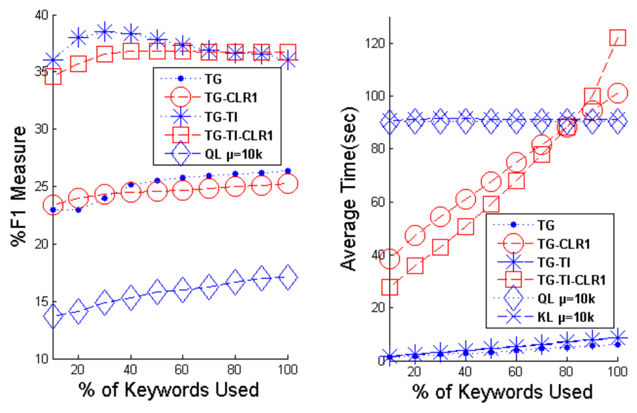
Precision Recall

Fig. 9 Precision and recall for the city of Amsterdam (@Top1 and @0-Step)



Precision Recall

Fig. 8 Precision and recall for the city of Berlin (@Top1 and @0-Step)



F1 Score Comparison Execution Time

Fig. 10 Trade-off between F1 score and execution time for the city of Milan (@Top1 and @0-Step)

5.10 Focusing on precision

We now examine the behavior of our algorithms when we want to achieve high precision, which is useful for several applications.

In the first set of experiments, we employ a dynamic similarity threshold that determines whether the algorithm will make a prediction for the geolocation. The thresholds we use are automatically set, based on the results of the same timeslots of the previous days: They are computed as the mean of the similarities of the correctly identified geolocations, averaged over the corresponding timeslots of the previous days. We have 48 (dynamic) thresholds, one per half-hour slide. Evidently, these thresholds lead to fewer predictions of tweet geolocations, reducing the recall, but increasing the precision.

In Fig. 11, we present the precisions and the recalls after the introduction of the thresholds for the method TG-TI, while in Fig. 12 we present the precision and recall for TG-TI-CLR1. We run experiments by using the exact dynamic threshold, the exact threshold +-10 % and the exact threshold +-20 %. Furthermore, in order to evaluate the results, we use again the balanced F1 score. The F1 score for the two methods presented before is depicted in Fig. 13.

After evaluating our methods using the first most similar answer, we analyzed the results when taking under consideration the first 3 (Top3) and the first 5 (Top5) most similar candidates. In Figs. 14 and 15, we can see depicted mean precisions and recalls when using 10–100 % of the keywords for both cases. The results show that both precision and recall are benefiting, with the F1 scores increasing from around 35 % to around 55 % (Fig. 16).

Finally, we study the performance of our methods in the case where we relax the definition of the correct answer to include answers that are 1 square (1-*Step*) or 2 squares (2-*Steps*) away from the exact answer. That is, we consider the

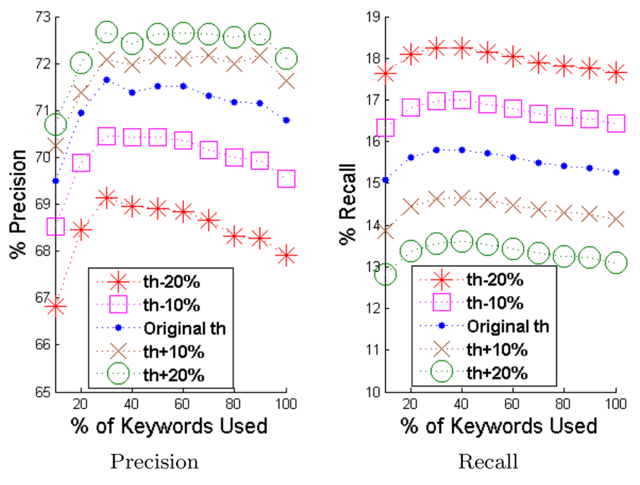


Fig. 11 Precision and recall on neighborhood level for TG-TI when using dynamic thresholds (Th) (@Top1 and @0-Step)

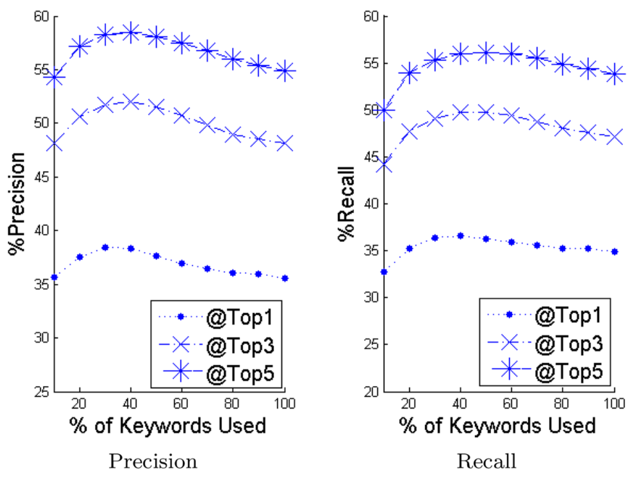


Fig. 14 Precision and recall for TG-TI (@0-Step)

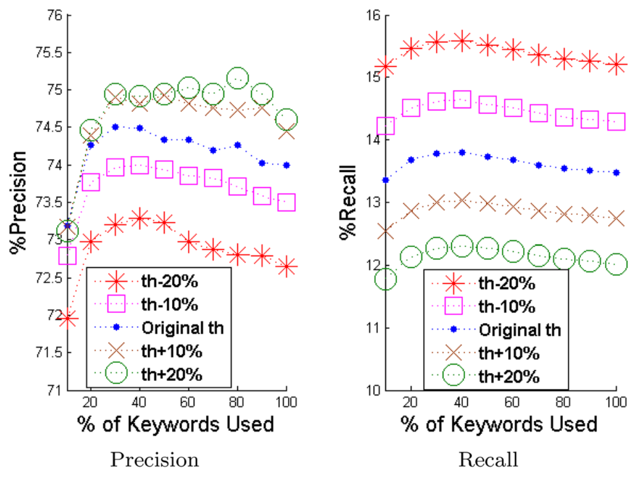


Fig. 12 Precision and recall on neighborhood level for TG-TI-CLR1 when using dynamic thresholds (Th) (@Top1 and @0-Step)

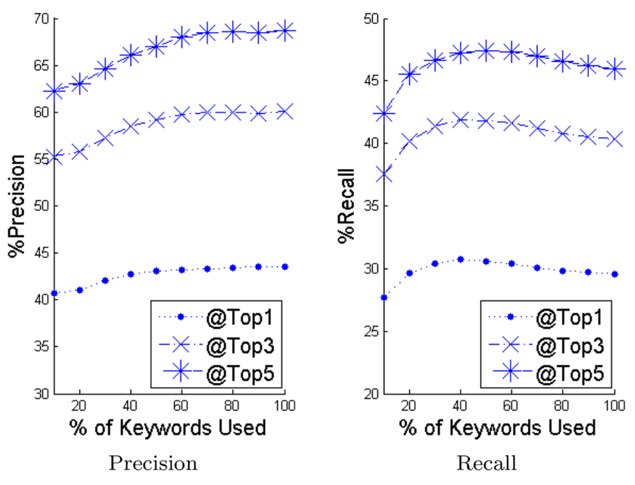


Fig. 15 Precision and recall for TG-TI-CLR1 (@0-Step)

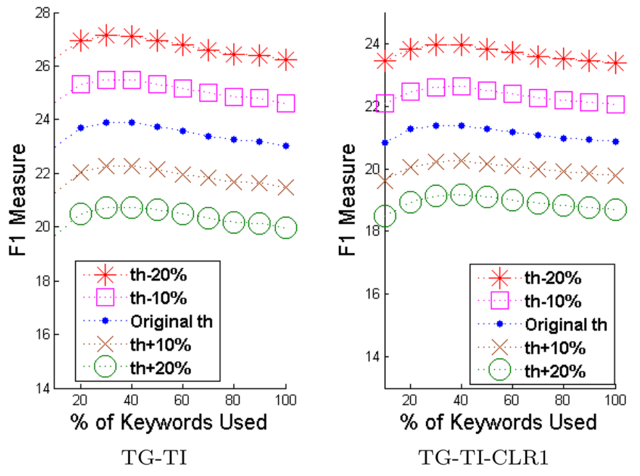


Fig. 13 F1 measure for neighborhood level with threshold (Th) (@Top1 and @0-Step)

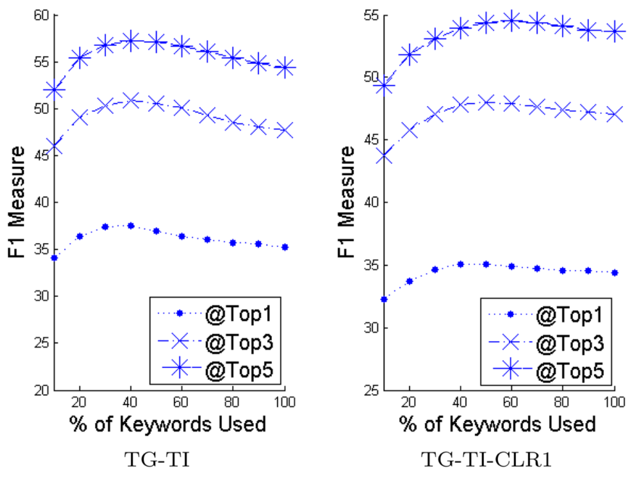


Fig. 16 F1 Score for TG-TI and TG-TI-CLR1 (@0-Step)

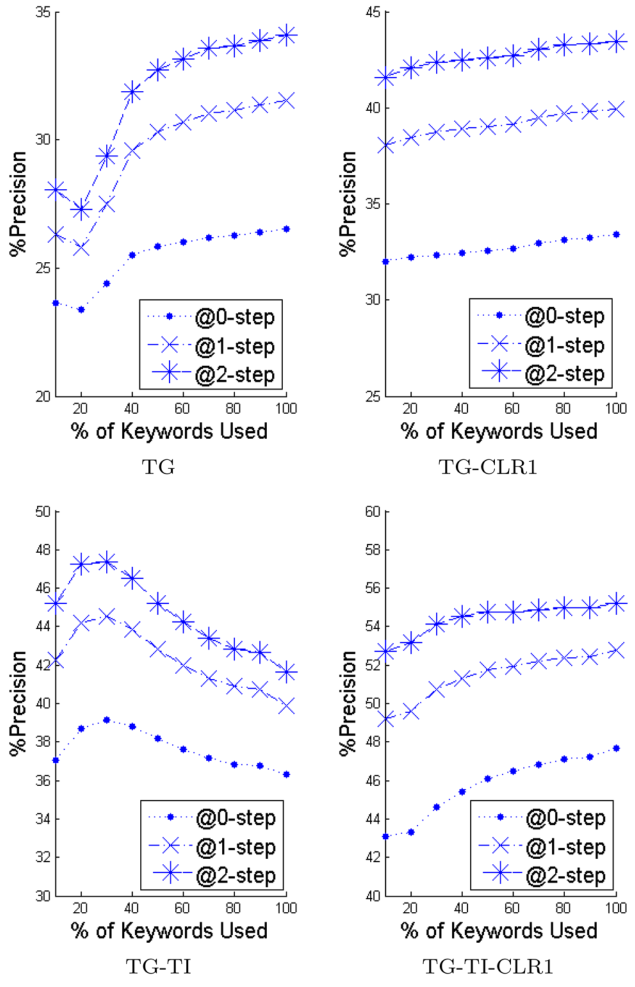


Fig. 17 Precision for neighborhood level (@Top1)

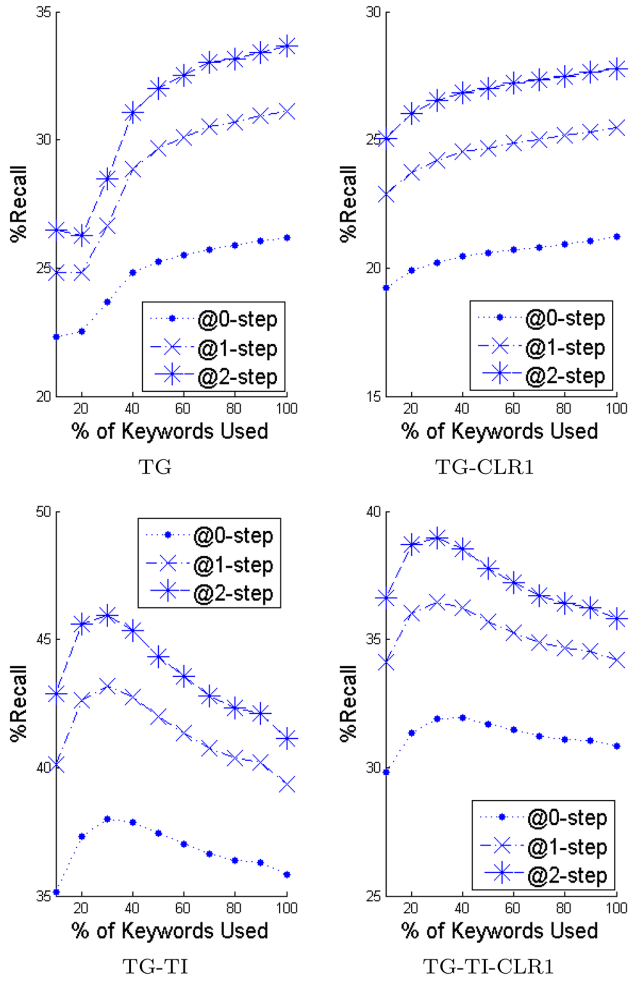


Fig. 18 Recall for neighborhood level (@Top1)

near neighbors of the exact answer to be correct answers, as well. The results of this evaluation are depicted in Figs. 17 and 18 (we report the results for the city of Milan).

When using the 1-*Step* evaluation, we observe an increase of up to 6 % for precision and up to 4 % for recall. The additional benefit for 2 *Steps* is diminishing, exhibiting an increase of up to 4 % for precision and up to 2 % for recall. This effect of diminishing returns is due to the fact that immediately neighboring squares tend to share the same topic, while the topic dilutes and differs more when we move further away. In all cases, TG-TI-CLR1 accounts for the best mean precision. The second best precision in the one is achieved by TG-TI, while the third best is achieved by TG-CLR1.

Finally, we run experiments modifying at the same time all the three parameters presented before, namely the similarity threshold, the @Step and the TopK. In Fig. 19, we illustrate the precision and recall of the TG-TI-CLR1 method. The results show that we can achieve a significant increase in precision, but only a modest increase in recall. We note that precision hovers above the 75 %, therefore making the proposed approach attractive for applications that need access to the geolocations of tweets.

5.11 Size of search space

In order to evaluate our method with larger search spaces, we created a bigger grid for the city of Rome, and we ran experiments on this new dataset. In particular, we created a grid of 900 squares (30 by 30), while keeping the rest of the setup parameters the same. The size of each square is the same as before: 1 km. In Fig. 20, we compare the precision and recall of the Tf-Idf methods for the 20 by 20 and the 30 by 30 grids.

The best precision for the 30 by 30 grid is 45 % and is achieved by TG-TI-CLR1 when using 100 % of keywords, while the best recall is 37 % and achieved by TG-TI when using 40 % of the keywords. As expected due to the higher search space, the precision and recall achieved by each method are lower than those for the smaller grid: They

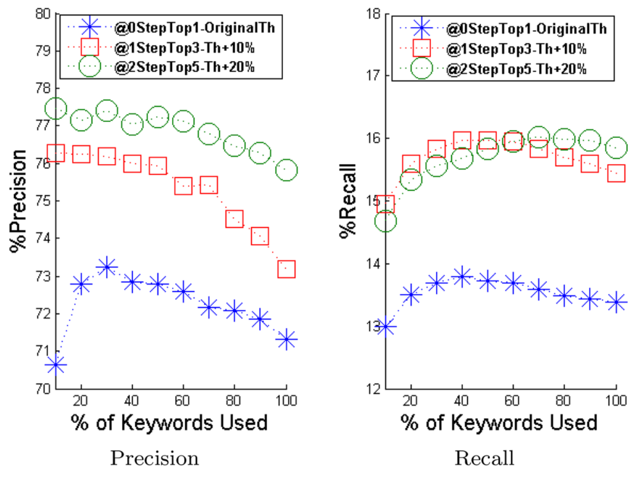


Fig. 19 Precision and recall for TG-TI-CLR1 for varying similarity threshold, TopK, and @Step

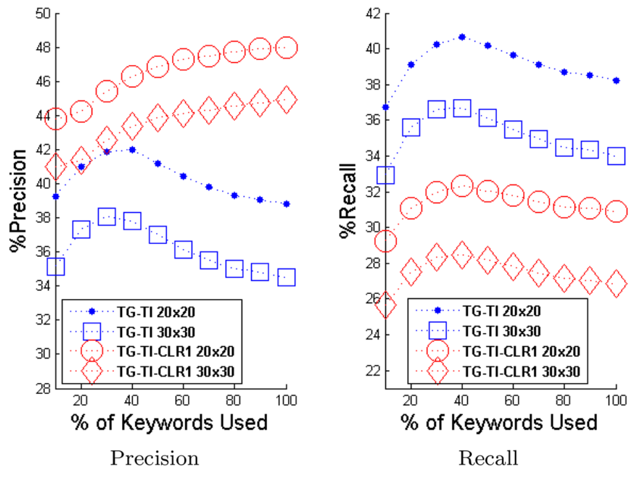


Fig. 20 Precision and recall comparison for the city of Rome (grids 20 × 20 and 30 × 30, @Top1 and @0-Step)

were up to 4 % lower for both algorithms, when the search space increased by 225 %. These results demonstrate that the effect of the increase in the search space on the proposed algorithms is relatively small.

5.12 Discussion

Overall, our results show that using correlation between local and global activities has the potential (when properly employed) to lead to significantly better accuracy.

We also observe that, contrary to previous work, the time needed to train and test our models depends on the percentage of keywords used. This allows us to achieve a trade-off between execution time and accuracy. An interesting point regarding this trade-off is the fact that the increase in the execution time that the *CLR* methods exhibit when using higher percentage of keywords does not pay off with a proportional increase in precision and/or recall.

For the @*Top*1 case, the Tf-Idf-based algorithms are the winners, providing better results than the simpler algorithms based on concordance. Furthermore, when using the Tf-Idf-based algorithms, the best result is achieved when pruning some of the keywords. This is due to the fact that pruning the keywords with the lowest weight, we primarily remove stopwords, which has a positive impact on accuracy. This is also true for TG-TI-CLR1, when considering the F1 score.

Regarding the difference in precision and recall between our approach and the baselines, we believe that it is due to the very different granularity requirements of the problems, especially the temporal granularity. Even though the baselines provide good results for identifying the characteristic topics of a location (when there are enough data), our approach has an advantage for geolocalizing tweets referring to time-focused events, especially those with a relatively short time span (e.g., concerts).

6 Conclusions

The extended use of social networks has resulted in an abundance of information on different aspects of everyday social activities and has led to a proliferation of tools and applications that can help end users and large-scale event organizers to better plan and manage their activities. Several of these applications are based on the knowledge of the geolocation of the relevant information. However, in Twitter, only a small percentage of the posts are geotagged.

In this work, we address the problem of geolocalizing non-geotagged tweets. We have proposed a framework that allows the estimation of the location from which a post was generated, by exploiting the similarities in the content between this post and a set of geotagged tweets. Contrary to previous approaches, our framework provides geolocation estimates at a fine grain, thus supporting a range of applications that require this detailed knowledge. The experimental evaluation with real data demonstrates the efficiency and effectiveness of our approach, which when coupled with the right visualizations (Paraskevopoulos et al. 2016) can become a powerful analysis tool. In our future work, we plan to study the use of more elaborate models for the representation of the keywords in a tweet. The challenge here is to identify the right abstraction, given the short length of tweets.

Acknowledgments This work was supported by a fellowship from Telecom Italia.

References

Abdelhaq H, Sengstock C, Gertz M (2013) Eventweet: online localized event detection from twitter. In: Proceedings of the VLDB Endowment , vol 6, no 12

Ajao O, Hong J, Liu W (2015) A survey of location inference techniques on twitter. J Inf Sci 41(6):855–864

Balduini M, Bocconi, S, Bozzon A, Della Valle E, Huang Y, Oosterman J, Palpanas T, Tsytsarau M (2014) A case study of active, continuous and predictive social media analytics for smart city. In: ISWC workshop on semantics for smarter cities (S4SC)

Balduini M, Della Valle E, DellAglio D, Tsytsarau M, Palpanas T, Confalonieri C (2013) Social listening of city scale events using the streaming linked data framework. In: ISWC

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Chang Hw, Lee D, Eltaher M, Lee J (2012) @ phillies tweeting from philly? Predicting twitter user locations with spatial word usage. In: ASONAM

Cheng Z, Caverlee J, Lee K (2010) You are where you tweet: a content-based approach to geo-locating twitter users. In: CIKM

Crooks A, Croitoru A, Stefanidis A, Radzikowski J (2013) # Earthquake: Twitter as a distributed sensor system. Trans GIS 17(1):124–147

Earle PS, Bowden DC, Guy M (2012) Twitter earthquake detection: earthquake monitoring in a social world. Ann Geophys 54(6):708–715

Eisenstein J, O'Connor B, Smith NA, Xing EP (2010) A latent variable model for geographic lexical variation. In: EMNLP

Facebook. https://www.facebook.com/

Frias-Martinez V, Soto V, Hohwald H, Frias-Martinez E (2012) Characterizing urban landscapes using geolocated tweets. In: SocialCom-PASSAT

Google+. https://plus.google.com

Han B, Cook P, Baldwin T (2014) Text-based twitter user geolocation prediction. J Artif Intell Res 49:451–500

Hossain N, Hu T, Feizi R, Zheng D, White AM, Luo J, Kautz H (2016) Precise localization of homes and activities: detecting drinking-while-tweeting patterns in communities. In: Tenth international AAAI conference on web and social media, Cologne, Germany, May 17-20, 2016, pp 587–590

Ikawa Y, Enoki M, Tatsubori M (2012) Location inference using microblog messages. In: Proceedings of the 21st international conference companion on World Wide Web. ACM, pp 687–690

Kinsella S, Murdock V, O'Hare N (2011) I'm eating a sandwich in glasgow: modeling locations with tweets. In: SMUC

Leetaru K, Wang S, Cao G, Padmanabhan A, Shook E (2013) Mapping the global twitter heartbeat: the geography of twitter. First Monday 18(5). doi:10.5210/fm.v18i5.4366

Li C, Sun A (2014) Fine-grained location extraction from tweets with temporal awareness. In: SIGIR

Malmi E, Do TMT, Gatica-Perez D (2013) From foursquare to my square: learning check-in behavior from multiple sources. In: ICWSM

Mathioudakis M, Koudas N (2010) Twittermonitor: trend detection over the twitter stream. In: SIGMOD

Murdock V (2011) Your mileage may vary: on the limits of social media. SIGSPATIAL Spec 3:62–66

Paradesi SM (2011) Geotagging tweets using their content. In: FLAIRS conference

Paraskevopoulos P, Dinh TC, Dashdorj Z, Palpanas T, Serafini L (2013) Identification and characterization of human behavior patterns from mobile phone data. In: NetMob

Paraskevopoulos P, Palpanas T (2015) Fine-grained geolocalisation of non-geotagged tweets. In: Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015. ACM, pp 105–112

Paraskevopoulos P, Pellegrini G, Palpanas T (2016) When a tweet finds its place: fine-grained tweet geolocalisation. In: International workshop on data science for social good (SoGood), in conjunction with the European conference on machine learning and principles and practice of knowledge discovery (ECML PKDD)

Sakaki T, Okazaki M, Matsuo Y (2010) Earthquake shakes twitter users: real-time event detection by social sensors. In: WWW

Schulz A, Hadjakos A, Paulheim H, Nachtwey J, Mühlhäuser M (2013) A multi-indicator approach for geolocalization of tweets. In: ICWSM

Serdyukov P, Murdock V, Van Zwol R (2009) Placing flickr photos on a map. In: SIGIR

Tsytsarau M, Amer-Yahia S, Palpanas T (2013) Efficient sentiment correlation for large-scale demographics. In: SIGMOD

Tsytsarau M, Palpanas T (2014) Nia: system for news impact analytics. In: KDD workshop on interactive data exploration and analytics (IDEA)

Tsytsarau M, Palpanas T (2012) Survey on mining subjective data on the web. Data Min Knowl Discov 24:478–514

Tsytsarau M, Palpanas T, Castellanos M (2014) Dynamics of news events and social media reaction. In: SIGKDD

Tsytsarau M, Palpanas T, Denecke K (2010) Scalable discovery of contradictions on the web. In: WWW

Tsytsarau M, Palpanas T, Denecke K (2011) Scalable detection of sentiment-based contradictions. In: DiversiWeb, WWW

Twitter. https://twitter.com

Van Canneyt S, Van Laere O, Schockaert S, Dhoedt B (2012) Using social media to find places of interest: a case study. In: SIGSPATIAL (GEOCROWD)

Yuan Q, Cong G, Ma Z, Sun A, Thalmann NM (2013) Who, where, when and what: discover spatio-temporal topics for twitter users. In: SIGKDD

Zafarani R, Liu H (2015) Evaluation without ground truth in social media research. Commun ACM 58(6):54–60