Fine-Grained Geolocalisation of Non-Geotagged Tweets

Pavlos Paraskevopoulos
University of Trento
Telecom Italia - SKIL
p.paraskevopoulos@unitn.it

Themis Palpanas
Paris Descartes University
themis@mi.parisdescartes.fr

***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., geolocalisation, of the relevant information, which is necessary for several applications (e.g., on trending venues, traffic jams, etc.). 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.**

Keywords: geotag, geolocation, Twitter, social networks

I. Introduction

Several social networks have emerged during the last decade. Social networks, such as Twitter [1], Facebook [2] and Google+ [3], 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 smart-phones 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 [4], including applications [5], [6], [7], [8], [9], [10], [11], [12] and techniques [13], [14], [15], [16] 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.

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 [8], to identify and report natural disasters, such as earthquakes [5], [10], and to monitor and track mobility and traffic [9]. 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.

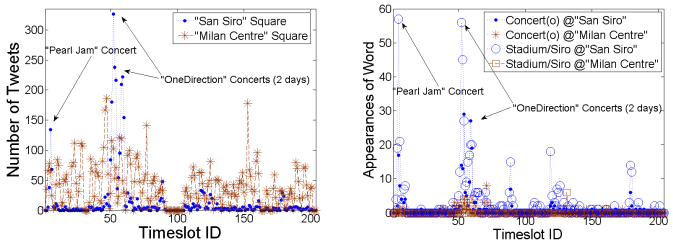
Evidently, the availability of geotagged data, determines not only the possibility to use such applications, but also their quality-performance characteristics: the more geotagged data posts are available, the better the quality of the results will be (more accurately: the higher the probability for 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 [17], [18], [19]. 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 geolocalising tweets that are non-geotagged. Even though previous works have recognized the importance and have studied this problem [20], [21] (for a comprehensive discussion of this problem refer to [19]), 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 zipcodes, cities, and geographical areas larger than cities. In contrast, we study this problem at a much finer granularity, providing location estimates for *individual* tweets at the level of *city neighborhoods*, thus enabling a new range of applications that require detailed geolocalised data.

We illustrate and motivate some of these ideas in Figure 1. Figure 1a depicts the number of tweets posted from the neighborhood in which the "$SanSiroStadium$" is located, and from a neighborhood located in the center of the Milan (Italy), while Figure 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

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



(a) Number of Tweets (b) Appearances of Words

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

performance despite the increased size of the search space of the algorithm. The framework we describe for the fine-grained geolocalisation 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.

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

- We describe and define the problem of fine-grained geolocalisation 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 document is organized as follows. In Section II we present the related work. Section III formalizes the problem, and Section IV describes our solution. We present our experimental evaluation in Section V, and conclude in Section VI.

II. State Of The Art

Several works have studied the problem of geotagged tweet analysis. Balduini et al. [9] 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 [22]. Abdelhaq et al. [23] 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 [5]. 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) is the focus of a recent study [24]. The authors analyzing tweets that have check-in data of Foursquare. Another study proposed a framework that can automatically recognize POIs by correlating geotagged tweets with geotagged data deriving from Flickr [25]. 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 smart-phone, has been used in order to predict the next location of the users [26].

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 spatio-temporal profiles of users, and ultimately lead to identification of mobility patterns [27]. Cheng et al. [28] 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 [20]. 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 [19].

Two studies that target to geotag tweets are presented in [29] and [30]. These two methods create chains of words that represent a location by using Latent Dirichlet Allocation (LDA) [31]. 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 [21]. In this study, the authors construct language models by using Bayesian inversion, achieving good results for the country and state level identification tasks. Finally, [32] presented a method for identifying the geolocation of photos by using the textual annotations of these photos.

Even though some of these studies are closely related to our work (e.g., [20], [21], 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 zipcodes 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.

III. 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 $P_{t_j}^{l_1}, ..., P_{t_j}^{l_i}$, $t_1 \leq t_j \leq t_2$, 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}, $t_1 \leq t_q \leq t_2$, 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 zipcode). 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.

IV. Proposed Approach

In this section, we describe our solution to the problem of fine-grained geolocalisation 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 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 meters). 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 hour 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.

A. 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: $df_{keyword} = \log(\frac{n}{k})$, where n is the number of documents, k is the number of documents that keyword appears in, and $tfidf_{i,keyword} = \frac{count}{l} * df_{keyword}$, 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 geolocalise, we follow the same process as before.

B. 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 candidate geolocations i, mag_{i_t}, for a given time interval t. We denote with $kwVector[j]$ the weight of the j-th 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

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** $location$

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).

Algorithm 2 Similarity Calculation and Probability Extraction

1: **procedure** VECTORSIM(vectors for Q_t and candidate geolocations)
2: $mag_{Q_t} \leftarrow \sqrt{\sum_{\forall j \in kwVector_{Q_t}} kwVector_{Q_t}[j]^2}$
3: **for all** $i \in Geolocs$ **do**
4: $mag_{i_t} \leftarrow \sqrt{\sum_{\forall j \in kwVector_{i_t}} kwVector_{i_t}[j]^2}$
5: $Sim_{i_t,Q_t} \leftarrow \frac{\sum_j {}^{kwVector_{i_t}[j]*kwVector_{Q_t}[j]}}{mag_{i_t}*mag_{Q_t}}, \forall j \in kwVector_{Q_t} \cap kwVector_{i_t}$
6: **for all** $i \in Geolocs$ **do**
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 $Prob_{i_t,Q_t}$

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 meters) 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 ($Prob_{j_t,Q_t}$).

C. Similarity Based on Correlation of Activity Time Series

The similarity measure that we 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 activity 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.

Algorithm 3 Activity Correlation

1: **procedure** CORRELATIONSIM(global Gts_i and local Lts_j activity time series)
2: **for all** $j \in i$ **do**
3: $corr_{i_t,j_t} \leftarrow \frac{\Sigma(Gts_{i_t} - \bar{Gts_i})(Lts_{j_t} - \bar{Lts_j})}{\sqrt{\Sigma(Gts_{i_t} - \bar{Gts_i})^2 \Sigma(Lts_{j_t} - \bar{Lts_j})^2}}$
4: **for all** $j \in i$ **do**
5: $Prob_{i_t,j_t} = \frac{corr_{i_t,j_t}}{\sum corr_{i_t}}$, (fine-grain geolocation) $j \subset i$ (coarse-grain geolocation)
6: **return** $Prob_{i_t,j_t}$

This process is shown in Algorithm 3. Since we are only interested in similar behavior between local and global activity, we only keep the positive correlations, which we normalize over the sum of all correlation values to produce a probability distribution. More specifically, we construct the global activity time series, Gts_i, for a coarse-grain geolocation (e.g., a city i), as well as the local activity time series, Lts_j, for all the fine-grain geolocations within the coarse-grain one (e.g., the neighborhoods j inside the city i). The correlation we compute between these time series is the Pearson's correlation. Finally, the algorithm returns the probability of each candidate geolocation.

We can then combine this method with the TG algorithm, by multiplying the two similarity measures (based on concordance and correlation), to obtain the *TG-C (Tweet Geotagging with activity Correlation)* algorithm. When we do the same with the TG-TI algorithm, we get the *TG-TI-C (Tweet Geotagging with Tf-Idf and activity Correlation)* algorithm.

V. EXPERIMENTAL EVALUATION

Experimental Setup. We run the experiments at a machine that has OS Ubuntu 14.0.4 LTS, "4GB RAM" and processor "Intel Core i3 CPU M370 @2.40GHz x 4". For the implementation of our methods and the reimplementation of the PM-T, KL and GL we used Python 2.7.

Dataset. For the evaluation of our approach, we use a dataset containing English and Italian geotagged posts from Twitter,

generated in Italy between June 20 and July 23, 2014. In particular, we have data from 6 of the largest Italian cities, namely, Rome, Milan, Naples, Bologna, Venice and Turin. The granularity of the neighborhood level we use is a square with side of 1000 meters. The time intervals we use have a duration of 4 hours (which can effectively capture an important event, as well as the start and the aftermath of this event), but we also keep detailed aggregated information for every 15min interval. The total number of tweets is 543.295 (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).

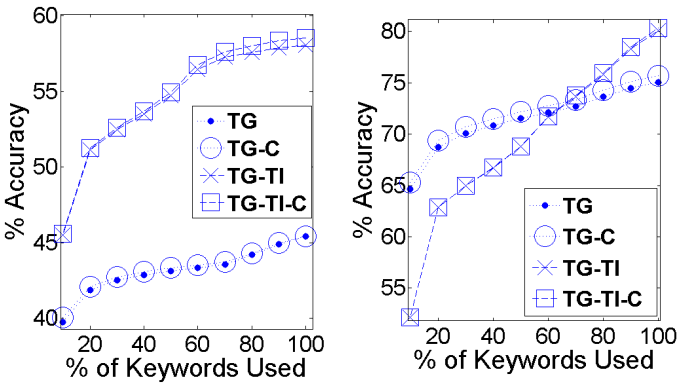
Algorithms. We experimentally evaluate the four algorithms we described in Section IV, namely, TG, TG-TI, TG-C, and TG-TI-C. As baselines, we implemented the PM-T and KL methods [21], which aim to solve a similar problem. We experimented with several values for the μ parameters used in these methods, and verified that $\mu = 10000$ gave the best results in our setting, as well. Furthermore, we implemented the GL method [20], considering each unique tweet as a unique user (which resulted in user profiles with only a few keywords).

Evaluation Measures. We study the time performance, as well as the effectiveness of each approach using the precision and recall measures: $Precicion = \frac{cgTweets}{gTweets}$ and $Recall = \frac{cgTweets}{aTweets}$, where $cTweets$ is the number of the correctly geolocalised tweets, $gTweets$ is the number of tweets we geolocalised, and $aTweets$ is the number of all tweets in the test set. In the cases 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. We also report the balanced F1 measure, $F1 = 2 * \frac{Precicion*Recall}{Precicion+Recall}$. Following previous work [21], we report the results when we consider the top-1 (@Top1), top-3 (@Top3), and top-5 (@Top5; only for neigborhood 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 in 80%training and 20% testing, repeated each experiment 30 times, and reported the mean values in the results.

A. City-Level Results

We start our analysis by running our method on city-level. We extract the English and Italian tweets from the 6 cities, removing the duplicated posts in order to avoid spam. We record the activity every 15 minutes, and we consider time intervals of 4 hour, leading to 181 timeslots (due to technical problems some of the timeslots were empty,and we do not consider those in 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 Section IV-C): we use the correlation between the activity time-series of the 6 cities and the activity time series of Italy. The time series are z-normalized every 24 hours and we compute the Pearson's correlation. The results (@Top1 and @0-Step) are presented in Figure 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



(a) @Top1 (b) @Top3

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

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 those two algorithms, we found that for 134 timeslots TG-TI-C has better accuracy, for 4 timeslots TG-TI and TG-TI-C have the same accuracy, and for the rest 43 timeslots TG-TI has better results. We note that the accuracy of an algorithm based on random choice was 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 Figure 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.

B. Neighbourhood-Level Results

At this subsection we present the evaluation we did for our approach at the neighborhood level. Every time we run the algorithm, we get the similarity both before and after achieving correlation between the total number of tweets from Milan and the number of tweets from every square.

In Figure 3a, we present the mean accuracy that our algorithms have among all timeslots, depending on the percentage of the keywords used while taking into consideration only the first answer. After analyzing the results, we come to the conclusion that the best mean accuracy is 38% and is achieved by using 80% of the keywords and when using the TG-TI-C algorithm. The second best algorithm is TG-TI. In this case, the best accuracy is achieved when using 80% of the keywords and is almost 38%. The best accuracy achieved by TG is 35%, while TG-C reached 34% (both achieved when using 100% of the keywords). In order to make fair the random choice, we were not choosing between all the 400 squares but only

between those which had data at the train datasets. The mean accuracy of the random algorithm was 2%.

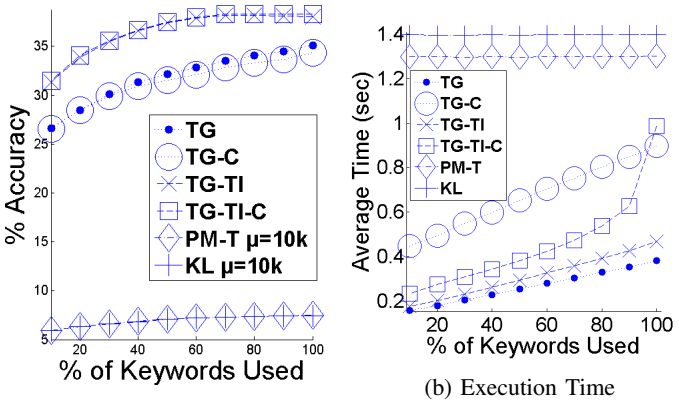
The maximum accuracy we got for one timeslot is 74% and we got it using the TG-TI algorithm. Nevertheless, the best results tend to be when we use TG-TI-C. As happens at city level, the accuracy tends to get increased after the use of the correlation between city and square activity when using Tf-Idf, but on the contrary to the city level, it gets decreased when adding the correlation parameter to the method that uses as weights the raw number of the appearances of each word. This may be caused due to the fact that people post for the same topic even if they are at neighboring squares. For example, during the concert identified at city-level, we do not have great accuracy because people were tweeting even on their way there, causing neighboring squares to have the same topic.

After evaluating our algorithms, we compared them to the PM-T and KL baselines, using the same spatial and temporal granularities as those we used for our algorithms. These results are depicted in Figure 3a. Surprisingly, we found that our results are up to 31% better. This is due to the spatial and temporal setting-up that we use. The authors of [21] originally use much bigger spatial granularity, while the temporal granularity of the two datasets that they use is 4 weeks and 3 months respectively. Moreover, probably due to our granularity, the results between PM-T and KL are almost the same.

Furthermore, we run experiments with the GL method, whose accuracy was in the best case around 4-5%. We believe that this is due to the differences in the problem definition and focus of this method, which is geared towards spatio-temporal granularities that are much larger than the ones we consider in our work.

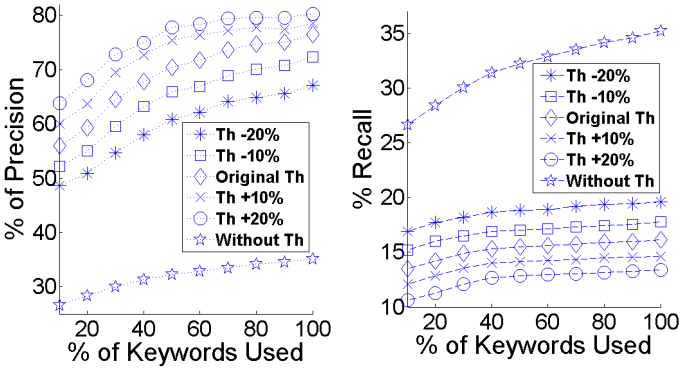
In order to check the trade-offs when using a percentage of the keywords and to compare the execution times needed for our algorithms and the state-of-the-art used, we measured the mean execution time needed per timeslot for training the models and answering the query-tweets. The results are depicted in Figure 3b. As we can see in the graph, the best time is achieved by TG. The reason is that it does not spend time for calculating neither the new keyword weights, nor the correlations. The worst execution time of our 4 methods is achieved by the method TG-C. This is due to the fact that although we prune the keyword-space, we do not remove stopwords or common words that appear in many squares. As a result, the new similarities have to be recalculated for all those candidate squares that have the common or stopwords, while by using Tf-Idf we eliminate many of the candidates.

After having evaluated our methods and compared them with the state-of-the-art when answering to all the queries, we evaluated our methods when using a dynamically defined similarity threshold. The threshold we have chosen to use are automatically calculated by the results of the 4-hour timeslots of the previous days. As a result, we have 6 user-free dynamic thresholds that are calculated by taking the mean of the mean similarities of the previous respective timeslots. By introducing the thresholds in our methods, we answer to less query-tweets, reducing the recall but increasing the precision up to 100%. In Figure 4, we present the precisions and the recalls



(a) Accuracy (b) Execution Time

Fig. 3: Trade-off Between Execution Time and Accuracy for Neighbourhood Level (@Top1 and @0-Step).



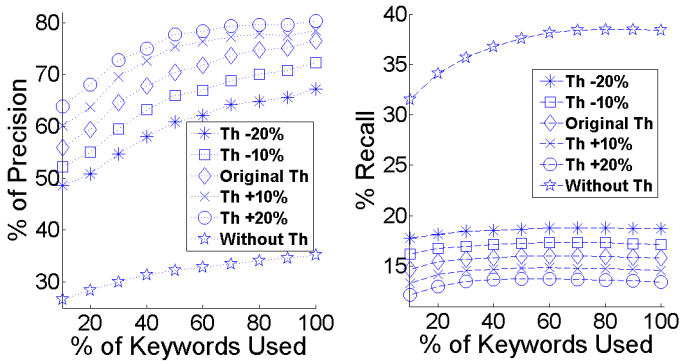
(a) Precision (b) Recall

Fig. 4: Precision and recall on Neighbourhood Level for TG when using dynamic thresholds (Th) (@Top1 and @0-Step).

after the introduction of the thresholds for the method TG-C, while in Figure 5 we present the precision and recall for TG-TI-C. We run experiments by using no threshold, the exact dynamic threshold, and the exact threshold +-10% and +-20%. Furthermore, in order to evaluate the results, we use the balanced F1-measure. The results of the F1 measure for the two methods presented before is depicted in Figure 6.

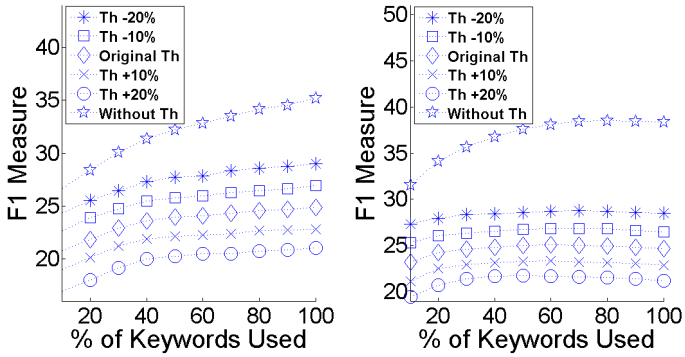
After evaluating our methods using the first most similar answer, we analyzed the results when taking under consideration the first 3 and the first 5 most similar candidates. In Figures 7a and 7b, we can see depicted the mean accuracies when using 10-100% of the keywords for both cases. On the contrary, at the city-level analysis, the TG-TI and TG-TI-C algorithms are always better when compared to TG.

Finally, we study the performance of our methods in the case where we consider the $1 - Step$ and $2 - Steps$ squares neighboring to the exact answer, as correct answers as well. The results of this evaluation are depicted in Figure 8. When using the $1 - Step$ evaluation we have up to 7% difference



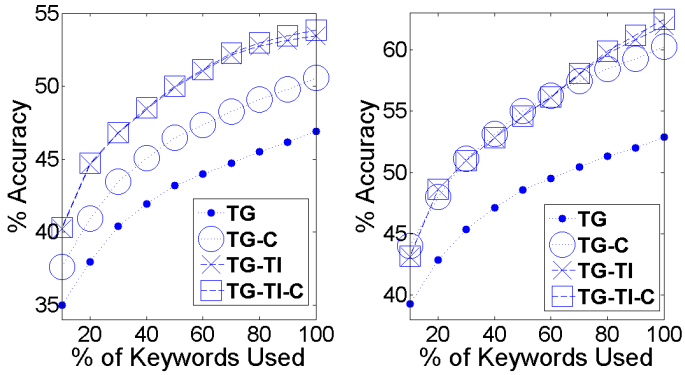
(a) Precision (b) Recall

Fig. 5: Precision and recall on Neighbourhood Level for TG-TI-C when using dynamic thresholds (Th) (@Top1 and @0-Step).



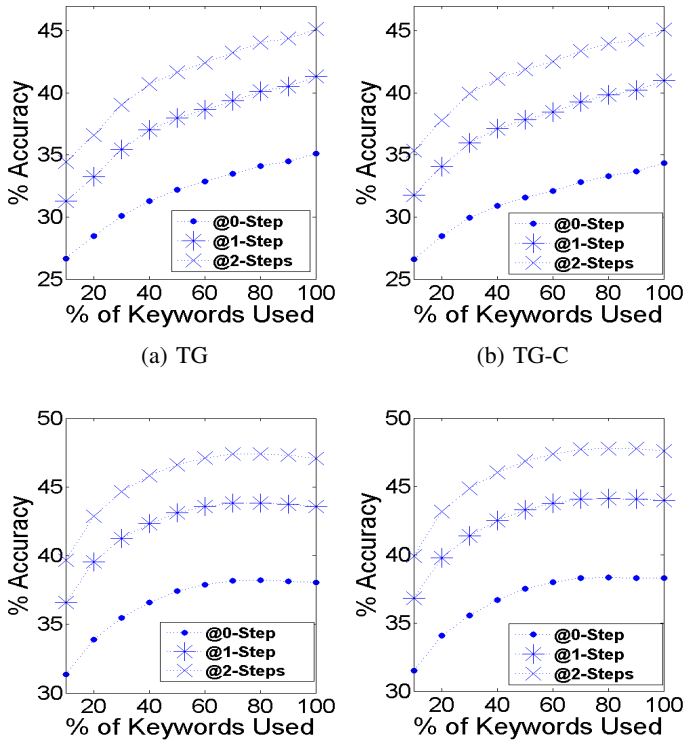
(a) TG (b) TG-TI-C

Fig. 6: F1 measure for Neighbourhood Level without and with threshold (Th) (@Top1 and @0-Step).



(a) @Top3 (b) @Top5

Fig. 7: Accuracy for Neighbourhood Level for TG, TG-C, TG-TI, and TG-TI-C (@0-Step).



(a) TG (b) TG-C

(c) TG-TI (d) TG-TI-C

Fig. 8: Accuracy for Neighbourhood Level (@Top1).

for the accuracy achieved by using the TG-C compared to the same method when using the exact answer case. Though the difference between the exact answer and the $1-Step$ is so big, the difference between the same methods when using $2-Steps$ is only up to 4% better. Probably this difference is due to the fact that neighboring squares share the same topic while the topic differs more comparing to the neighbors of the neighbors. Furthermore, when using up to 30% of the keywords, TG-TI and TG-TI-C for the exact answer have better accuracy compared to those that TG and TG-C have for the $1-Step$. The percentage of the keywords for which TG-TI and TG-TI-C of $2-Steps$ have better accuracy compared to the TG and TG-C of the $1-Step$ have, is even bigger coming up to 80% of the keywords, a fact that becomes even more interesting when taking into consideration the complexity of the methods when we have $n-Step$ and $(n-1)-Step$ evaluations. After analyzing the results in detail, we identified that in all the cases the best mean accuracy appears for TG-TI-C, in the cases of the exact answer or the $1-Step$ match when using 80% of the keywords, while for the case $2-Step$ we get the best accuracy when using 90% of the keywords. The second best mean accuracy in the one achieved with TG-TI, while the third best method is TG.

C. Discussion

Overall, our results show that using the correlation between local and global activity can lead to better accuracy when the candidate geolocations do not share the same topics. For the $@Top1$ 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 a part of our keywords. This is due to the fact that pruning the keywords with the lowest weight we mainly remove stopwords (while by using 100% of the keywords, we include the stopwords, and thus reduce the 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 when using different percentage of keywords, while maintaining high accuracy.

Regarding the difference in accuracy 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 geolocalising tweets referring to time-focused events (e.g., concerts), which appear suddenly and disappear soon afterwards.

VI. 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 geolocalising 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. 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.

References

[1] Twitter,https://twitter.com.

[2] Facebook,https://www.facebook.com/.

[3] Google+,https://plus.google.com.

[4] M. Tsytsarau and T. Palpanas, "Survey on mining subjective data on the web," *Data Mining and Knowledge Discovery*, 2012.

[5] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *WWW*, 2010.

[6] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *SIGMOD*, 2010.

[7] M. Tsytsarau, T. Palpanas, and K. Denecke, "Scalable discovery of contradictions on the web," in *WWW*, 2010.

[8] V. Frias-Martinez, V. Soto, H. Hohwald, and E. Frias-Martinez, "Characterizing urban landscapes using geolocated tweets," in *SocialCom-PASSAT*, 2012.

[9] M. Balduini, E. Della Valle, D. DellAglio, M. Tsytsarau, T. Palpanas, and C. Confalonieri, "Social listening of city scale events using the streaming linked data framework," in *ISWC*, 2013.

[10] A. Crooks, A. Croitoru, A. Stefanidis, and J. Radzikowski, "# earthquake: Twitter as a distributed sensor system," *Transactions in GIS*, vol. 17, no. 1, pp. 124–147, 2013.

[11] M. Balduini, S. Bocconi, A. Bozzon, E. Della Valle, Y. Huang, J. Oosterman, T. Palpanas, and M. Tsytsarau, "A case study of active, continuous and predictive social media analytics for smart city," in *ISWC Workshop on Semantics for Smarter Cities (S4SC)*.

[12] M. Tsytsarau and T. Palpanas, "Nia: System for news impact analytics," *KDD Workshop on Interactive Data Exploration and Analytics (IDEA)*, 2014.

[13] M. Tsytsarau, T. Palpanas, and K. Denecke, "Scalable detection of sentiment-based contradictions," *DiversiWeb, WWW*, 2011.

[14] M. Tsytsarau, S. Amer-Yahia, and T. Palpanas, "Efficient sentiment correlation for large-scale demographics," in *SIGMOD*, 2013.

[15] P. Paraskevopoulos, T.-C. Dinh, Z. Dashdorj, T. Palpanas, and L. Serafini, "Identification and characterization of human behavior patterns from mobile phone data," *NetMob*, 2013.

[16] M. Tsytsarau, T. Palpanas, and M. Castellanos, "Dynamics of news events and social media reaction," in *SIGKDD*, 2014.

[17] K. Leetaru, S. Wang, G. Cao, A. Padmanabhan, and E. Shook, "Mapping the global twitter heartbeat: The geography of twitter," *First Monday*, vol. 18, no. 5, 2013.

[18] V. Murdock, "Your mileage may vary: on the limits of social media," *SIGSPATIAL Special*, 2011.

[19] B. Han, P. Cook, and T. Baldwin, "Text-based twitter user geolocation prediction," *JAIR*, 2014.

[20] H.-w. Chang, D. Lee, M. Eltaher, and J. Lee, "@ phillies tweeting from philly? predicting twitter user locations with spatial word usage," in *ASONAM*, 2012.

[21] S. Kinsella, V. Murdock, and N. O'Hare, "I'm eating a sandwich in glasgow: modeling locations with tweets," in *SMUC*, 2011.

[22] P. S. Earle, D. C. Bowden, and M. Guy, "Twitter earthquake detection: earthquake monitoring in a social world," *Annals of Geophysics*, vol. 54, no. 6, 2012.

[23] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet: Online localized event detection from twitter," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, 2013.

[24] C. Li and A. Sun, "Fine-grained location extraction from tweets with temporal awareness," in *SIGIR*, 2014.

[25] S. Van Canneyt, O. Van Laere, S. Schockaert, and B. Dhoedt, "Using social media to find places of interest: a case study," in *SIGSPATIAL (GEOCROWD)*, 2012.

[26] E. Malmi, T. M. T. Do, and D. Gatica-Perez, "From foursquare to my square: Learning check-in behavior from multiple sources," in *ICWSM*, 2013.

[27] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Who, where, when and what: discover spatio-temporal topics for twitter users," in *SIGKDD*, 2013.

[28] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: a content-based approach to geo-locating twitter users," in *CIKM*, 2010.

[29] J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing, "A latent variable model for geographic lexical variation," in *EMNLP*, 2010.

[30] S. M. Paradesi, "Geotagging tweets using their content." in *FLAIRS Conference*, 2011.

[31] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, 2003.

[32] P. Serdyukov, V. Murdock, and R. Van Zwol, "Placing flickr photos on a map," in *SIGIR*, 2009.