

SNAF: Observation filtering and location inference for event monitoring on twitter

Yihong Zhang¹ · Claudia Szabo² · Quan Z. Sheng³ ·
Xiu Susie Fang²

Received: 1 February 2016 / Revised: 28 December 2016 /
Accepted: 8 March 2017 / Published online: 24 March 2017
© Springer Science+Business Media New York 2017

Abstract Twitter has recently emerged as a popular microblogging service that has 284 million monthly active users around the world. A part of the 500 million tweets posted on Twitter everyday are personal observations of immediate environment. If provided with time and location information, these observations can be seen as sensory readings for monitoring and localizing objects and events of interests. Location information on Twitter, however, is scarce, with less than 1% of tweets have associated GPS coordinates. Current researches on Twitter location inference mostly focus on city-level or coarser inference, and cannot provide accurate results for fine-grained locations. We propose an event monitoring system for Twitter that emphasizes local events, called SNAF (Sense and Focus). The system filters personal observations posted on Twitter and infers location of each report. Our extensive experiments with real Twitter data show that, the proposed observation filtering approach can have about 22% improvement over existing filtering techniques, and our location inference approach can increase the location accuracy by up to 36% within the 3km error range. By aggregating the observation reports with location information, our

✉ Yihong Zhang
yihong.zhang@ntu.edu.sg

Claudia Szabo
claudia.szabo@adelaide.edu.au

Quan Z. Sheng
michael.sheng@mq.edu.au

Xiu Susie Fang
xiu.fang@adelaide.edu.au

¹ School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798, Singapore

² School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia

³ Department of Computing, Macquarie University, Sydney, NSW 2109, Australia

prototype event monitoring system can detect real world events, in many case earlier than news reports.

Keywords Twitter · Social sensor · Message filtering · Location inference · Event detection

1 Introduction

A micro-blogging service such as Twitter allows its users to conveniently create and publish *tweets*, which are short messages of maximum 140 characters and are immediately available for anyone to read online. As Twitter has gained enormous popularity over the past five years, the number of users and published tweets has increased significantly. Nowadays, 284 million active Twitter users generate 500 million tweets every day¹. Moreover, as a direct effect of the prevalence of smartphone, 80% of Twitter users are using their mobile phones to create tweets[1]. The use of mobile platform for tweeting implies that users can conveniently report current events and objects in their physical vicinity.

In this paper, we assume that a user is an ordinary person, as opposed to a community or a corporation. A user's tweet can be a description of the user's mood, a message to a friend, or a link to an online article the user wants to share. A user may also post observations of surrounding environments, such as "*the air is fresh*", or "*I just saw a homeless man outside of the church*". We call this type of tweets *immediate observations*, because they describe the state of the observed object, person, or event, at the time and location when and where the user posts the message.

The immediate observations of personal Twitter users can be seen as reading values of some sensory devices [23]. Take, for example, a user posting a tweet in Times Square on Tuesday morning, saying "*the air is fresh*". This can be seen as an air quality sensor installed in Times Square that just recorded a good reading on Tuesday morning. An immediate observation posted on Twitter can be converted to sensory values, if the time and location of the tweet are known. Given the high density of Twitter users in urban areas, the perspective of Twitter users as sensors instantly provides a dense virtual sensor network, by which some events and objects of interest can be monitored. Several works have proposed using tweets as sensory values to monitor environmental disasters [17, 23].

Using personal observations published on Twitter as sensory values, however, has two challenges. The first is to filter the personal observation reports from other types of messages. For instance, in one experiment we examined 100 tweets containing the keyword "hailstorm", but only 35% were found as actual personal observations of hailstorm. Tweets that are not immediate observations include hailstorm reported by news agency, and metaphoric use of the word hailstorm. Previous works has shown filtering Twitter messages is a challenging task, and existing approaches only achieved unsatisfactory accuracy [23, 28]. Current work on microblog and short text analysis mostly relies on supervised machine learning methods [4, 24], which require the manual preparation of training samples. This has several drawbacks, such as the significant manual effort for annotating examples, and a lack of quality guarantees of the classification solutions, when the classifier is applied to a wider pool of tweets beyond its training data. In this paper, we thus aim to develop an unsupervised filtering method.

¹<https://about.twitter.com/company>

The other challenge is to provide the missing location information. As opposed to time information, which is readily available for all tweets, location information is scarce. Current works for obtaining the geo-location information of the tweets rely heavily on the tweets' GPS data [12, 23]. In one of our experiments, we sampled one thousand random tweets, but found that only 0.9% had GPS data. This implies that if only considering GPS-enabled tweets, most of the observations posted on Twitter will not be noticed. Suppose there is an air pollution around a certain corner of the city, even with ten Twitter users reporting it, the probability of overlooking the pollution can be as high as 0.91^2 . To increase the availability of location information, tweet messages and other publicly accessible information on Twitter need to be treated as location sources [16, 17].

Extraction of location information from tweets has been studied by several works [6–8, 15–17, 26]. Many of these works focus on the city-level location [6, 7, 16]. However, finer-grained location information is often vital for local events, such as shooting incidents and vehicle crashes. Previous studies show that, exact location extraction from tweet texts often leads to large errors [8, 26]. The issues include the use of informal names, mentioning of a placename other than the place where the user is located, and the lack of a comprehensive gazetteer. In tweet messages, “can” could mean Canada, “philly” could mean Philadelphia, and “central park” could mean the Central Park in New York or Sydney. It is difficult to find out all associations between locations and their informal names, whether using a learning model [8] or a gazetteer [26]. If a comprehensive gazetteer is generated to cover all informal names, on the other hand, many words that do not mean placenames will be mistakenly captured as placenames.

In this paper, we propose SNAF (Sense and Focus), an event monitoring system that captures local events reported on Twitter and accurately infers event locations. SNAF has three components, the *personal observation filter*, the *location estimator*, and the *event detector*. An overview of the system is shown in Figure 1.

For filtering the personal observation reports, SNAF uses lexical analysis and user profiling based on statistics generated from the Twitter data. Filtered reports are sent to the location estimator for location inference. The location estimator is based on a large gazetteer and distance-based data cleaning algorithms. It also takes the poster's past tweets as sources to increase the availability of location information. Given the accurately classified personal observation reports and inferred location information, our event detector is able to successfully capture local events, even before news reports. The main contributions of our work are summarized as follows:

- We propose a method that converts personal observations of events and objects of interest reported on Twitter into sensory values. This is achieved by first filtering personal observation reports from other type of tweets, then providing the missing location information.
- We propose three outlier removal methods that significantly improve the location inference accuracy. Our experimental results with real Twitter data show that, comparing with the existing approaches, our method improves the accuracy of location inference by over 30% in smaller error ranges, and over 20% in larger error ranges.
- We show the potential of further use of converted sensory values from observation tweets, by designing and implementing an event monitoring system, which aggregates personal observations of an event or an object of interest based on their locations. We

²As $(1 - 0.009)^{10} \approx 0.91$

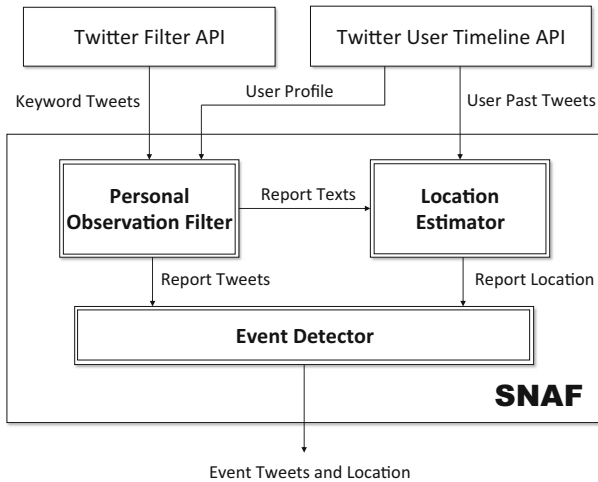


Figure 1 Sense and focus overview

verify the system by comparing the detected events to relevant news articles, and find that the event detected by our system are in many cases several hours faster than news articles, while the locations inferred are accurate comparing to the locations reported in the news.

This paper includes findings and materials from two of our conference papers [36] and [37], with expanded explanations and literature reviews. In addition, we introduce a new location inference technique, which is shown to produce significant improvement over existing results. We also present a prototype web application that incorporates proposed techniques for realtime event monitoring and display.

2 Related work

Our work follows a current research trend of converting microblog messages to sensory values for the purpose of monitoring events and objects of interest [16, 23, 38]. Sakaki et al. [23] proposed and investigated the idea of using tweets as sensory values for environmental event monitoring. They collected earthquake-related tweets and filtered them to generate reports about ongoing earthquakes. Using this information, they built a system for predicting the movements of earthquakes in Japan. Their results show that the use of tweets as sensory values is feasible, and that their system can detect earthquakes within a minute after they occur, five minutes faster than announcements from meteorological authorities. In a later work, they extended the system to predict typhoon movements, which achieves a similar accuracy and response time [24]. Machine learning approach is also used in other works for filtering observations of particular events [14, 22, 28, 29, 38]. Sriram et al. [28] proposed a machine learning-based filter for classifying tweet categories such as news, opinions, events, and private messages, using features including author name, the use of opinioned words, currency signs, and mention signs. Kwon et al. [14] identified effective lexical and

temporal features for distinguishing rumors in event-related tweets. Olteanu et al. [22] proposed keyword selection methods for filtering tweets related to particular natural disasters. While these works identified tweets that can be used for monitoring particular events or objects of interest, they are not sufficient for converting tweets into sensory values, as the majority of tweets are not explicitly associated with location data.

Current researches in location inference on Twitter focus on different granularities. A category of research aims at inferring city-level locations [6, 7, 16]. Li et al. [16] inferred tweet location by first inferring the user's home location, relying on the home location entries in Twitter's user profiles, which are usually entered as cities. Graham et al. [6] also exploited home location entries in user profiles for inferring the location of the tweets and users.

Another category of research aims at inferring finer-grained locations [8, 15, 17, 26]. Li et al. [15] identified placenames in tweet messages based on Foursquare data, but does not link the location to coordinates. Another Foursquare data application is shown by Ji et al. [10], in which structured Foursquare location profiles are exploited. Ikawa et al. [8] inferred tweet locations by matching the tweet with tweets with known locations using cosine similarity, and also based on Foursquare data. Foursquare is a location-based service that provides accurate place-coordinate association for commercial places such as hotels and restaurants, but in a recent business operation, the service has been terminated³. Schulz et al. [26] leveraged DBpedia for identifying places in tweet messages, but resulted in large errors, with the median error distance of *1,100km*. DBpedia is a large user-contributed database for name-entities used in the Web [1]. Using a generation tool called DBpedia Spotlight [5], we can generate a gazetteer containing more than 800,000 places with respective GPS coordinates, which covers a large number of street-level places. However, as we will discuss below, such large gazetteer will make location information very noisy.

The earthquake and typhoon detection system proposed in [24] takes all tweets containing the keyword as the tweets for a single event, assuming that earthquakes and typhoons do not occur frequently. Local events such as vehicle crashes occur in much higher frequencies with smaller impact areas. Local event detection thus requires clustering, preferably with location information [16, 19, 33]. Unankard et al. [33] proposed an event detection method based on clustering of words. They relate events to locations, but only at a country-level. Li et al. [16] proposed a classification-based event detection method for crime and disaster events, with a location extraction component focuses on city-level locations. The lack of accurate and fine-grained location information is the main issue that prevents existing event monitoring systems from obtaining local details.

Noisy sensor readings have been extensively studied in sensor networks. Basic techniques include temporal and spatial aggregation using mean or median [9, 34, 35]. Particularly, the median is effective for avoiding extreme outlier values in the dataset [34]. More advanced techniques identify and remove outliers in the dataset, based on distance measures or clustering. Since it is straightforward to interpret the distance between two location points, the distance-based outlier detection is particularly suitable for location data. Subramaniam et al. [30] and Branch et al. [2] both used *k*-Nearest Neighborhood (KNN) for identifying outliers. An issue of using KNN is how to choose the proper distance threshold that excludes the outliers. Sheng et al. [27] proposed two KNN-based outlier detection algorithms, using a fixed distance threshold and a relative distance threshold, respectively.

³<http://time.com/3024078/foursquare-swarm/>

3 Filtering personal observation reports

Our method filters observations of objects and events from personal accounts, by implementing the following steps. First, we identify observations from collected tweets for a specific keyword. Second, using the collected tweets, we distinguish personal accounts from other types of accounts. A personal account is a Twitter account employed for personal use, and is assumed to be free from business or propaganda interests. Our insight is that tweets from personal accounts often contain realtime and localized observations of objects and events. Finally, from the observation tweets identified in the first step, we retain only those made from personal accounts. These personal observations of objects and events have proved useful in previous works for scenarios such as disaster location and rumor detection [14, 24].

An overview of our method is shown in Figure 2. To identify observation tweets, we run lexical analysis on tweet texts based on the par-of-speech (POS) tagging, objectivity analysis, and originality test. To identify personal accounts, we first analyze four attributes for each user, namely, *objectivity*, *interactivity*, *originality*, and *topic focus*. Then we use a clustering algorithm for classifying personal accounts based on the attribute values. We describe our method below.

3.1 Observation filtering

After using the Twitter Filter API⁴ to obtain tweets that contain the object or event keyword such as “rainbow” or “car accident”, our lexical analysis method focuses on extracting observation tweets. Not all tweets containing the keywords are observations of objects and events, since in some cases the keywords can have another semantic, context-based meaning, and the objects and events can be mentioned in general comments instead of specific observations, e.g., “I dislike car accidents”. We address this by utilizing three techniques, namely, *par-of-speech* (POS) tagging, *objectivity analysis*, and *originality test*. POS tagging allows filtering of messages in which the object or event keyword is not used as a subject of observation. Objectivity analysis allows filtering of uncertain messages, such as questions and general comments. Originality test removes messages that are not originally created by the user, such as retweets or quotations.

3.1.1 Filtering based on part-of-speech tagging

Our insight is that the objects and events mentioned in an observation are most likely to be nouns and gerunds, such as in “I just saw a rainbow”, or “A shooting outside my home”. On the other hand, keywords not used as nouns and gerunds often indicate that the tweet is not a specific observation. Some examples of non-observation tweets are shown in Table 1, with the role of the keyword determined by POS tagging.

POS tagging is a technique that matches words in a text with their part-of-speech categories, such as modal, noun, verb, and adverb [25]. We use a filtering rule on top of POS tagging to effectively remove a portion of tweets that are clearly not observations. After performing POS tagging for a tweet, we accept it if the POS tag for the keyword is **NN** (Noun, singular or mass), **NNP** (proper noun, singular), and **VBG** (verb, gerund or present participle). The tweet is rejected if the keyword has other POS tags.

⁴<https://dev.twitter.com/streaming/public>

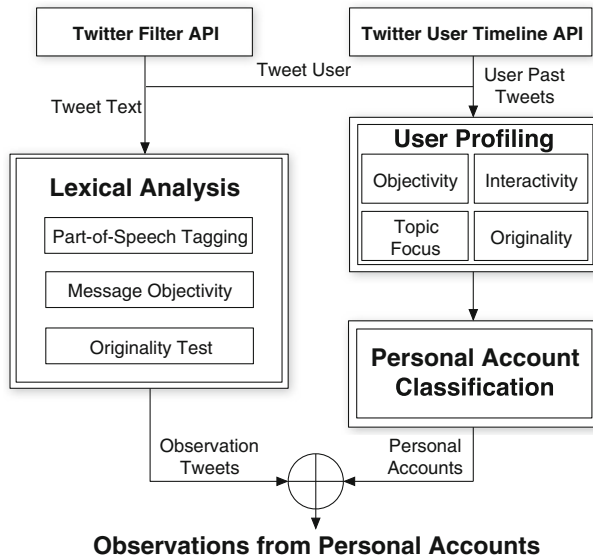


Figure 2 Personal observation filtering method overview

3.1.2 Filtering based on objectivity analysis

Our insight is that a specific observation of an object or event usually is written in a more objective tone than a general tweet. Generally, the objectivity of a message is affected by sentimental words and uncertain words, such as “great”, “bad”, “maybe”, “anyone”. Sentimentality and uncertainty as factors for determining message objectivity has already been proposed in existing works [3, 21]. We calculate tweet objectivity based on both sentimentality and uncertainty, using the following formula:

$$objectivity(t) = 1 - [senti_p(t) + 0.5 \times senti_n(t)] \times (1 - \sqrt{uncertainty(t)})$$

where $senti_p$ is the positive sentiment and $senti_n$ is the negative sentiment. In our previous works, we have found that negative sentiments have a large presence in observation messages [38]. We follow this insight here and we weight down the effect of negative sentiments

Table 1 Non-observation tweets filtered by POS tagging, for monitoring flight delay, shooting incidents, and rainbows

Tweet Text	POS
Keep praying for the typhoon to magically delay my flight a day	VB
Can we pretend that airplanes, in the night sky, are like shooting stars?	JJ
This guy got on a rainbow colored LV belt	JJ

VB=base form verb, JJ=adjective

on reducing the objectivity in the formula. Furthermore, since uncertainty plays an important role in determining the objectivity of a message, as discovered in [21], we increase the effect of uncertainty by scaling it to a larger value.

For sentiment analysis, we employ previously proven effective methods, which employ a positive/negative words dictionary and the slang sentiment dictionary [32]. The positive and negative sentiments of a tweet text t are measured as:

$$senti_p(t) = \frac{count_p(t)}{count_w(t)}$$

$$senti_n(t) = \frac{count_n(t)}{count_w(t)}$$

where $count_p(t)$ and $count_n(t)$ are the word count for positive and negative words in t , and $count_w(t)$ is the word count of t .

For uncertainty analysis, we use a dictionary of uncertain words based on the LIWC category of hesitation words [31]. To measure the uncertainty of tweet t , we consider the number of uncertain words in the text, and whether it is a question.

$$uncertainty(t) = \begin{cases} 0.5, & \text{if } t \text{ ends with a question mark} \\ \frac{count_u(t)}{count_w(t)}, & \text{otherwise} \end{cases}$$

where $count_u(t)$ is the word count for uncertain words in t .

3.1.3 Originality test

Our analysis of various datasets show that sometimes personal users may repeat some messages created by other users, which do not count as their own observations. The repeated messages not only produce redundancy, but also generate noises for analysis. Thus it is crucial to determine message originality. We proposed a set of rules to determine non-original messages based on message content, as shown in Table 2. A message satisfies any of the rules in the table is considered non-original, and will be filtered out.

Some repeated messages are easy to identify, such as retweets, which have “RT” at the beginning of the messages. Other forms of repeated messages can be more difficult to spot, such as indirect quotes, which often but not necessarily contain the word “says” or “claims”. Given the various ways a message may be repeated, the rules listed in Table 2 do not cover all non-original messages. Nevertheless, we found these rules to filter out most of the repeated messages.

Table 2 Originality Test Rules

Rule	Explanation
retweet	contains the word RT
quotation	contains quotation marks
speech	mention or capitalized word before colon
news title	all words capitalized before link
repeat	contains “says”, “claims”, “via”, or “according to”
news mention	mention contains “news”, “radio”, or “breaking”
news agent	mention contains news agent name such as “ABC” or “CNN”

3.1.4 Lexical analysis algorithm

Algorithm 1 describes our lexical analysis method. The input is a keyword w , and a set T of tweet texts containing the keyword. The output is a set of predictions of whether each tweet text $t \in T$ is an observation, O . In line 7, we use a parameter θ to control the level of objectivity a tweet requires to meet to be considered an observation. The default value for θ is the first quartile of overall objectivity in the tweet set.

Algorithm 1 Lexical Analysis on Single Tweets

INPUT: keyword w , tweet set T , objectivity threshold θ
OUTPUT: observation labels O

```

1: set all  $o \in O$  as false
2: for each  $t \in T$  do
3:   run POS tagging for  $t$ 
4:   if POS tag for  $w \in \{\text{NN, NNP, VBG}\}$  then
5:      $pp \leftarrow \text{true}$ 
6:   end if
7:   if  $\text{objectivity}(t) > \theta$  then
8:      $po \leftarrow \text{true}$ 
9:   end if
10:  if  $t$  fails all rules in Table 2
11:     $pt \leftarrow \text{true}$ 
12:  end if
13:   $o_t \leftarrow pp \wedge po \wedge pt$ 
14: end for

```

3.2 User profiling for personal account classification

Previous works have shown that news generated from personal observations on Twitter can be much faster than traditional media, and the implicitly-associated location data can be used for localizing the object or the event [24, 36]. However, there are many Twitter accounts that are not for personal use, and do not have the same time and location association for their observation messages, and while they add noises to the data collected, it is usually difficult to distinguish them from personal accounts. The main issue is that all accounts on Twitter uses the same format to store data, and usually there is no effective way to judge the type of account other than looking at the content of the account posts directly. These accounts include news, business, activist and advertisement accounts.

Our study of personal and specific-purpose accounts leads to the following observations:

- News accounts tweet about various topics in a strictly objective tone. Their tweets usually contain links to Web articles. Depending on the specialty, a media account can cover a wide range of topics.
- Business accounts contain conversations, observations, and product promotions, but the range of topic is limited to the specific business.
- Activist and advertisement accounts rarely use objective tone, and their range of topics is also limited.

A personal account, however, does not have such clear-cut characteristics as specific-purpose accounts, and usually contains a mix of information sharing, conversation with other users, and original content that covers various topics. We propose that:

Conjecture 1 A personal account has moderate levels in objectivity, interactivity, originality, and topic focus.

We use various statistics generated from Twitter data to calculate the levels of *objectivity*, *interactivity*, *originality*, and *topic focus* for Twitter users. We assume these user qualities are consistent over time and do not easily change. There are rare cases that the profile of a user changes drastically, for example, caused by a job change, but currently we do not consider such cases. To profile a user, first we collect a set of past tweets made by the user, H . Then we select the original tweets in H based on the rules described in Table 2, as $OH = \{oh_1, oh_2, \dots, oh_l\}$, where $|OH| = l$.

The objectivity of a user is calculated based on the objectivity of each tweet in OH :

$$u_{objectivity} = \frac{\sum_{i=1}^l objectivity(oh_i)}{l}$$

The interactivity of a user is calculated based on the number of tweets containing mention mark “@” in H :

$$u_{interactivity} = \frac{count@(H)}{|H|}$$

The originality of a user is calculated based on the fraction of original tweets in H .

$$u_{originality} = \frac{l}{|H|}$$

To calculate a user’s topic focus, we count the frequency of each topic word for all topic words appearing in OH . For simplicity, we consider a topic word as a word that starts with a capital letter. The first word in a sentence is ignored. Once we have a descendingly-sorted list of topic word occurrences $\{nt_1, nt_2, \dots, nt_k\}$, the topic focus of a user is calculated based on the fraction of the first quarter of the most frequent topic words:

$$u_{focus} = \frac{\sum_{i=1}^{n/4} nt_i}{\sum_{j=1}^n nt_j}$$

A user is thus profiled by the quadruple:

$$u = \{u_{objectivity}, u_{interactivity}, u_{originality}, u_{focus}\}$$

3.3 Personal account classification with profiles

We propose an algorithm for automatically identifying personal accounts based on the user profile. First we define the difference between two user profiles u_1 and u_2 as the Euclidian distance between two profiles:

$$d(u_1, u_2) = \sqrt{\sum (u_1 - u_2)^2}$$

where

$$\begin{aligned} \sum (u_1 - u_2)^2 = & (u_{objectivity1} - u_{objectivity2})^2 \\ & + (u_{interactivity1} - u_{interactivity2})^2 \\ & + (u_{originality1} - u_{originality2})^2 \\ & + (u_{focus1} - u_{focus2})^2 \end{aligned}$$

Following Conjecture 1, we see that the attributes of a personal account are usually closer to a set of mean values while a specific-purpose account usually holds more extreme values. Therefore we propose that:

Conjecture 2 Given a set of user profiles U , which contains personal account profiles P and specific-purpose account profiles S , there exists a mean profile \bar{u} , such that $\sum_{p \in P} d(p, \bar{u}) <$

$$\sum_{s \in S} d(s, \bar{u}).$$

While it is difficult to prove Conjecture 2, we find it generally true in our analysis, as we will show with our experiments. Given a set of user profiles U , and a mean profile \bar{u} , we can separate from U a subset C that is more likely to contain personal accounts, by selecting profiles that have shorter distance to \bar{u} .

We devise an iterative algorithm for finding the mean profile \bar{u} . Intuitively, we can use the mean attribute values of all profiles in U . However, the extreme attribute values of the specific-purpose account profiles can bias the mean significantly, making it inaccurate for deciding personal accounts. In Algorithm 2, we use an iterative approach and a cluster size threshold δ for selecting a cluster of $|U| \times \delta$ profiles that are close to an unbiased \bar{u} . Starting from an initial mean profile \bar{u}_0 , the algorithm alters between cluster updating (line 2, 6) and mean updating (line 4 and 5). In the cluster updating step, a number of profiles close to the mean are selected. In the mean updating step, a new mean is calculated based on the selected profiles. If there are extreme values that cause a bias in the cluster, the mean will move away from the bias, and replace the extreme value profiles with more average profiles in the cluster. The output of the algorithm, F , is a set of personal account predictions.

While Algorithm 2 generally finds a good mean profile that separates personal accounts and specific-purpose accounts. However, depending on the choice of the initial mean \bar{u}_0 , the algorithm sometimes produces undesirable results. To address this issue, we derive a particle swarm optimization (PSO) algorithm for finding the optimal \bar{u}_0 .

PSO is an optimization technique that takes a population of solutions, and iteratively improves the quality of the solutions by moving them toward the best solution in each iteration [11]. A solution in our PSO algorithm is an initial mean \bar{u}_0 to be given to Algorithm 2. A PSO algorithm requires the definition of the quality measure and the solution movement. To define the quality of a solution, we rely on our initial observation that personal accounts exhibit higher variance than any types of specific-purpose accounts. Therefore we propose that:

Algorithm 2 Predicting Personal Accounts

INPUT: user profiles U , mean profile \bar{u}_0 , selected cluster size δ

OUTPUT: F

```

1: set all  $f \in F$  as false
2:  $C \leftarrow |U| \times \delta$  profiles closest to  $\bar{u}_0$ 
3: while  $C \neq C'$  do
4:    $C' \leftarrow C$ 
5:    $\bar{u} \leftarrow$  mean attribute values of profiles in  $C$ 
6:    $C \leftarrow |U| \times \delta$  profiles closest to  $\bar{u}$ 
7: end while
8: for each  $u \in U$  do
9:   if  $u \in C$  then
10:     $f_u \leftarrow true$ 
11:   end if
12: end for

```

Conjecture 3 Given two user profile clusters C_1 and C_2 , if the profiles in C_1 are more diverse than C_2 , then C_1 is more likely to contain personal accounts.

We use pairwise profile differences to calculate the diversity of profiles in a cluster, $C = \{c_1, c_2, \dots, c_k\}$,

$$div(C) = \frac{2 \times \sum_{i=1}^{k-1} \sum_{j=i+1}^k d(c_i, c_j)}{k \cdot (k-1)}$$

For the solution movement in PSO, we set a moving speed v so in each iteration, a solution p moves towards the best solution p_b as:

$$p \leftarrow p + (p_b - p) \cdot v \quad (1)$$

Our PSO algorithm is shown as Algorithm 3. It starts with a number of random solutions (line 1) and for each solution, a profile cluster is generated using Algorithm 2 (line 2 to 4). Then iteratively, the PSO algorithm moves the best solution towards an optimal solution by comparing the cluster diversity with each solution (line 5 to 13).

Algorithm 3 PSO for Finding Optimal \bar{u}_0 **INPUT:** user profiles U , selected cluster size δ , number of particles n , speed v **OUTPUT:** p_b

```

1: randomly choose  $n$  solutions  $P$  in the solution space
2: for each  $p \in P$  do
3:   generate a cluster  $C_p$  using Algorithm 2
4: end for
5:  $p_b \leftarrow p$  with highest  $div(C_p)$ 
6: while  $p_b \neq p'_b$  do
7:    $p'_b \leftarrow p_b$ 
8:   for each  $p \in P$  do
9:      $p \leftarrow p + (p_b - p) \cdot v$ 
10:    generate a cluster  $C_p$  using Algorithm 2
11:   end for
12:    $p_b \leftarrow p$  with highest  $div(C_p)$ 
13: end while

```

The optimal initial mean produced by Algorithm 3 can then be used in Algorithm 2 for selecting the cluster of personal account profiles. Although Algorithm 3 requires two more parameters, during our experiments we find the effect of changing n and v negligible for any $n > 1,000$ and $v < 0.2$, as the solution already reaches optimal values. Therefore we can confidently set n and v to fixed values. The only parameter that still affects the classification result is the cluster size parameter δ , which controls the portion of profiles in the data to be selected as personal account profiles.

3.4 Overall algorithm

Algorithm 4 identifies observations from personal accounts. Given the input of a keyword w and a set of tweets M , and the control parameter θ and δ , the output is a set of predictions, R , of whether each respective tweet is an observation of the object or event of interest from personal accounts.

Algorithm 4 Filter Observations from Personal Accounts**INPUT:** keyword w , messages M , objectivity threshold θ , selected cluster size δ **OUTPUT:** R

```

1: set all  $r \in R$  as false
2:  $T \leftarrow$  tweet text from  $M$ 
3:  $U \leftarrow$  user profiles from  $M$ 
4:  $O \leftarrow$  run Algorithm 1 with  $w, T, \theta$ 
5:  $p_b \leftarrow$  run Algorithm 3 with  $U, \delta$ 
6:  $F \leftarrow$  run Algorithm 2 with  $U, p_b, \delta$ 
7: for each  $m \in M$  that has text  $t$  and user profile  $u$  do
8:   if  $o_t \wedge f_u$  then
9:      $r_m \leftarrow true$ 
10:  end if
11: end for

```

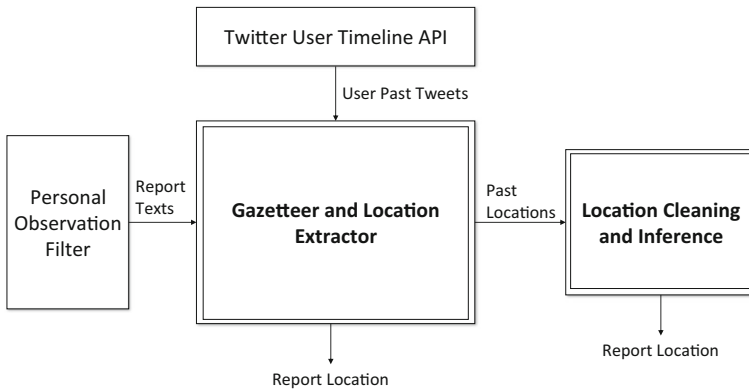


Figure 3 Report location inference method overview

4 Report location inference

A filtered report that does not have the GPS data will be sent to the location estimator. To infer the location of a report, we consider the tweet text as well as past tweets of the user who made the report as sources of locations. We follow a gazetteer-based approach. First we generate a gazetteer covering placenames in a dataset. Then we run name extraction for the report tweet and the user’s past tweets. In case past locations are considered, we run data cleaning algorithms to remove noises before inferring a likely location. An overview of the location inference is shown in Figure 3.

4.1 Location inference for a single tweet

To extract location, we run a gazetteer-based name extraction by comparing the words in the tweet text with gazetteer entries. As expected, the accuracy of this approach depends on the quality of the gazetteer. When testing a gazetteer, we found two potential issues, *coverage* and *noise*. The coverage problem occurs when a placename is present in the tweet but not in the gazetteer. The noise problem occurs when a word in the tweet matches an entry in the gazetteer but does not mean a placename in the context of the tweet. We take both issues into account when building our gazetteer.

We generated our base gazetteer using the latest version of DBpedia geo-coordinate dataset⁵ and the DBpedia Spotlight name generation tool [5]. DBpedia is a user-contributed name-entity database for Internet-of-Things data lookup, and in a recent version[5] includes 4.58 million names or things. The geo-coordinate dataset contains a subset of these names that are associated with a geo-coordinate. DBpedia Spotlight can track the usage of these names in Web documents such as Wikipedia pages, and produce a gazetteer containing formal and informal names. Our base gazetteer contains more than 800,000 entries and covers a large number of street-level places, including their formal and informal names, such as “*penn station*”, “*lion’s head*”, and “*east end*”. However, it also contains a considerable amount of names which are usually not referring to a place when people mention them on Twitter. Table 3 shows examples of four types of such names.

⁵<http://wiki.dbpedia.org/Datasets>

Table 3 Examples of non-placenames in the geo-coordinate dataset

Common Words	<i>in, can, bath, young, act, space, orange, king, cook, north, bucks, tower, paradise, marathon, racing, shape, corner, path</i>
Person's Names	<i>george, williams, georgia, lewis, nelson, anderson, stanley, monroe, katherine, clinton, reagan, douglas, harrison</i>
Brand and Organization Names	<i>nike, microsoft, nintendo, tesco, fbi, dell, nsa, red cross, mercedes, cadillac, java, absolute radio, chipotle, texas tech</i>
Symbolic Place Names	<i>troy, sparta, wall street, pentagon, third reich, green line, berlin wall, tiananmen square, pearl harbor, notre dame</i>

We use a heuristic consists of two rules to address the two issues mentioned above and refine the gazetteer. First we collect a set of GPS-enable tweets and extract locations from the messages using the base gazetteer. The extracted locations are compared with the GPS data. If the difference between the extracted location and the GPS data is larger than an acceptable threshold λ , we count a hit for the location. If the difference is larger than a rejection threshold θ where $\theta > \lambda$, we count a miss for the location. Usually a common location word will be extracted more than once. After the process, each extracted location will have a hit number and a miss number. We then pick the locations with at least one hit to put in the refined gazetteer, which ensures the *coverage* of the refined gazetteer. On the other hand, if the number of misses is more than half of the total extraction number, we remove it from the refined gazetteer. The purpose of this heuristic is to keep all the gazetteer entries relevant to the GPS data, with respect to the training dataset, which reduces the *noise* of the gazetteer. In our experiments we chose $\theta = 10\text{km}$ and $\lambda = 300\text{km}$. These numbers are relatively strict because we focus on small granularity, but different numbers can be chosen with different requirements in error tolerance.

After we had a refined gazetteer, we evaluated the precision of our gazetteer by comparing the coordinates of the location extracted from a single tweet with the GPS associated with the tweet. The results are shown in Table 4, where the precision for each distance range is the percentage of locations in all extracted locations that is within that distance range when compared to the GPS data. If we encounter a case that a name may refer to multiple locations, for simplicity, we use the first location found.

The precision of the name extraction on single tweets based on the gazetteer is relatively low. While we can use the general knowledge to distinguish, for example, a person's name from a placename, this judgement is not always accurate. Furthermore, a place mentioned by the user does not always reflect the location of the user or the tweet. For example, when a user tweets that he wants to go to *Gold Coast* for a holiday, this place is probably not related to the current location of the user or the tweet, even if the placename is successfully extracted from the tweet. Table 5 shows examples of placenames in tweets that do not indicate the current location of the user, where the error is the distance between the mentioned place and the tweet's GPS coordinates.

Therefore, even if we provide a gazetteer that accurately extracts placenames from tweets, we cannot avoid a certain level of noises, and these noises prevent location inference

Table 4 Precision of location extraction on single tweets using refined gazetteer

Error	$\leq 3\text{km}$	$\leq 5\text{km}$	$\leq 10\text{km}$	$\leq 30\text{km}$	$\leq 100\text{km}$
Precision	0.225	0.283	0.372	0.518	0.587

Table 5 Examples of placename in tweets not indicating user location

tweet	place	error(km)
Someone come with me to the South Shore Plaza	South Shore Plaza	8458
What are the chances of @bassnectar playing Outside Lands this weekend? #ibelieveinmiracles #bayjams	Outside Lands	12800
He was not armed when taken into custody. Just getting this info in from our CBS affiliate in Rockford.	Rockford	9926
going to see the new school	The New School	8666

from reaching high accuracy. More location information and further processing are required for inferring more accurate locations. We will obtain more location information by looking at users' past tweets.

4.2 Location resolution given past locations

When considering the past tweets from a user, we can generally extract a handful of past locations. The past tweets of Twitter users are publicly accessible through the timeline API provided by Twitter⁶, with a restricted availability of a maximum of 3,000 most recent tweets. About 5% tweets have a placename that can be identified using DBpedia-based gazetteer [26]. We conducted an experiment to analyze 1,000 random tweets and found that, while only 0.9% of tweets have GPS tags, 87% of the 982 users who made these tweets have at least one extractable placename in their past tweets, and 68% of them have more than ten. By using placenames extracted from past tweets to infer the current location, we can effectively associate location information with a significant portion of tweets.

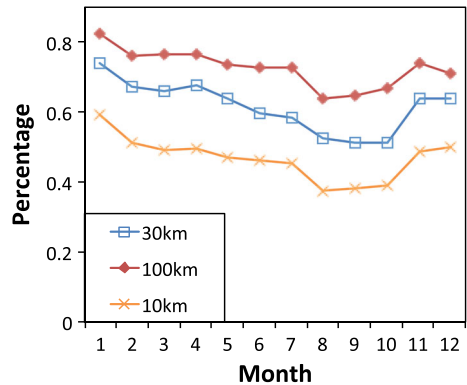
A general intuition is that the locations of a person are spatially correlated, and the locations appear in the past tweets can be used to infer the location of the current tweet. To verify this intuition, we investigated the temporal-spatial pattern of Twitter users based on GPS data. We extracted the past locations of 743 random users with at least 100 GPS-enabled tweets, and divided the location into months, based on number of days past until the last tweet. We then calculated the mean distance of locations in each month to the location of last tweet for each user. The results as the average of all users are shown in Figure 4.

The results showed that, more than 63% of the tweets in the past year are within 100km from the last tweet, regardless of how old they are. Also, even in the worst month, the eighth month, more than 50% of the tweets are within 30km from the last tweet, and 37% are within 10km. Therefore we conclude that past locations can be a good indicator for inferring current locations.

Various techniques have been proposed to infer the current location of an object based on past locations, such as Kalman filters and Particle filters. Since it is difficult to define a dynamic model for the movement of Twitter users, the Bayesian Filter-type inference techniques are ineffective. In sensor networks, a common way to aggregate spatial data is using the average. However, when the data are noisy and erroneous, the average will also deviate significantly from the correct location. As reported by Schulz et al. [26], using the

⁶<https://dev.twitter.com/rest/public/timelines>

Figure 4 Mean location differences and percentages for tweets in different past months



mean of locations found in tweet messages, the inferred location has an average error of 1,100km. Therefore it is quite crucial to clean the noisy and erroneous data before running average-based inference.

4.2.1 Distance-based outlier removal

A common way to clean noisy data is by removing outliers. Since it is straightforward to interpret the distance between location points, we choose two representative distance-based outlier detection techniques from the sensor network literature, called DK-Outlier and NK-Outlier [27].

Both outlier detection techniques are based on the *k*-Nearest Neighbor (KNN). Given a set of locations *H*, and a location $l \in H$, let $V(l)$ be the distances from *l* to all other locations in *H*, sorted in ascending order. Then $V_k(l)$ is the distance from *l* to its *k*-th nearest neighbor. Introduced in [13], given an acceptable distance threshold *d*, DK-Outlier is defined as:

Definition 1 A location *l* is a DK-Outlier if $V_k(l) \geq d$.

NK-Outlier is a new type of outlier proposed by [27]. NK-Outlier detection sets a relative distance threshold based on comparison between locations in the dataset. Given *n*, the number of other locations in the data a location needs to be comparable in order to be considered acceptable:

Definition 2 A location *l* is a NK-Outlier if there are no more than $n - 1$ other locations *p*, such that $V_k(l) < V_k(p)$.

The choice of *d*, *n* and *k* depends on the desired data accuracy and expected errors in the dataset. For example, we consider an acceptable past location is within 100km of the current location, and thus the maximum distance between two acceptable locations is 200km, so we set *d* as 200. When testing the accuracy of location extraction on single tweets, we found 58.7% extracted locations are within 100km error range, and 24% locations extracted had an error larger than 1,000km, so we set *k* as $0.587 \times |H|$, and *n* as $0.76 \times |H|$. The algorithms for removing DK-Outlier and NK-Outlier are shown below as Algorithm 5 and 6.

Algorithm 5 Removing DK-Outlier**INPUT:** H , extracted past locations of the user**OUTPUT:** H' , past locations after outliers removed

```

1:  $R \leftarrow null$ 
2: for each  $l \in H$  do
3:    $V \leftarrow$  distances from  $l$  to all other locations in  $H$ 
4:    $VS \leftarrow V$  sorted in ascending order
5:   if  $VS_k < d$  then
6:     add  $l$  to  $R$ 
7:   end if
8: end for
9:  $H' \leftarrow$  remove all  $l \in R$  from  $H$ 
10: return  $H'$ 

```

For removing DK-Outlier, we generate a distance list of one location to all other locations, V , for each location in the data (line 3). For each location, we sort the distance list in ascending order, and compare the k -th element with the predefined threshold d to decide if the location should be removed as an outlier (line 4 - 6).

Algorithm 6 Removing NK-Outlier**INPUT:** H , extracted past locations of the user**OUTPUT:** H' , past locations after outliers removed

```

1:  $R \leftarrow null$ 
2: for  $i \leftarrow 1$  to  $|H|$  do
3:    $V \leftarrow$  distances from  $H_i$  to all other locations in  $H$ 
4:    $VS \leftarrow V$  sorted in ascending order
5:    $K_i \leftarrow VS_k$ 
6: end for
7: for  $i \leftarrow 1$  to  $|H|$  do
8:    $count \leftarrow 0$ 
9:   for  $j \leftarrow 1$  to  $|H|$  do
10:    if  $K_j \geq K_i$  then
11:       $count++$ 
12:    end if
13:   end for
14:   if  $count < n$  then
15:     add  $l_i$  to  $R$ 
16:   end if
17: end for
18:  $H' \leftarrow$  remove all  $l \in R$  from  $H$ 
19: return  $H'$ 

```

For removing NK-Outlier, we also generate a distance list of each location to all other locations (line 3), and then take the k -th element in the ascendingly-sorted list as the representative value (line 4, 5). Then we compare the representative value of one location to other locations, count how many other representative values are greater than the one

associated with this location (line 9 - 13), before deciding if the location should be removed as an outlier (line 14 - 15).

4.2.2 Probabilistic outlier removal

We also propose a probabilistic outlier removal algorithm, called *Simple Expectation Maximization* (SEM). This algorithm is inspired by the Expectation Maximization framework [20], and incorporates a latent variable indicating whether each location mentioned reflects the correct location of the user. Given a number of past locations $H = \{h_1, \dots, h_{|H|}\}$, the algorithm aims to find a latent variable $Z = \{z_1, \dots, z_{|H|}\}$, where $z_i = 1$ if h_i is a correct location, and 0 otherwise. The algorithm alternates between two phases. In the first phase, it calculates a most-likely correct location given Z . In the second phase, it updates Z with the new parameter. The algorithm terminates when Z is no longer changing. The details of the algorithm is shown in Algorithm 7.

Algorithm 7 Simple Expectation Maximization

INPUT: H , extracted past locations of the user

OUTPUT: H' , past locations after outliers removed

```

1:  $z_i \leftarrow 1$  for each  $z_i \in Z$ 
2:  $\theta \leftarrow \frac{1}{\sum Z} \sum_{i=1}^{|H|} (h_i \times z_i)$ 
3: while  $\theta \neq \theta'$  do
4:   for each  $h_i \in H$  do
5:      $likelihood \leftarrow \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{dist(h_i, \theta)}{2\sigma^2}}$ 
6:      $likelihood < \delta$  then
7:        $z_i = 0$ 
8:     else
9:        $z_i = 1$ 
10:    end if
11:  end for
12:   $\theta' = \frac{1}{\sum Z} \sum_{i=1}^{|H|} (h_i \times z_i)$ 
13: end while
14:  $H' \leftarrow$  remove all  $h_i \in H$  if  $z_i = 0$ 
15: return  $H'$ 

```

This algorithm is similar to an Expectation Maximization algorithm with the Expectation step and the Maximization step, but does not have rigidly defined maximum likelihood. The latent variable Z is uniformly assigned an initial number (line 1). The model parameter θ is the mean location and is calculated using all locations considered as correct (line 2, 12). With a mean location, we find the likelihood of a mentioned location, assuming the correct locations are distributed normally around the mean location (line 5). If the likelihood of a location h_i is too low, we consider it as incorrect and assign the respective z_i to 0 (line 6 - 10). The threshold δ can be chosen considering the relative likelihood values, and usually set to such a value that 30% $z_i \in Z$ will get 1. If a new mean location θ' calculated using the updated Z is the same as the previous θ then the algorithm terminates, returning outlier removal results based on Z . Since this algorithm does not strictly follow the EM framework, we have no proof that it will always terminate, but in experiments we find that the algorithm terminates in less than 10 iterations most of the time.

4.2.3 Burst compression

When examining the data, we notice that users may have a burst of interest in a certain location, and post messages about this location intensively within a short period of time. For example, after a football match, an user may engage in an intense conversation with a friend about the match, which involves mentioning of the name of the football team that will be detected as a placename. This phenomenon often distorts the location inference significantly. Therefore we propose a compression algorithm, called *burst compression*, which basically removes consecutive occurrences of a location within three days from the first occurrence, as shown in Algorithm 8. This algorithm can be added to the above outlier removal algorithms as a pre-processing step.

Algorithm 8 Burst Compression

INPUT: H , extracted past locations of the user

OUTPUT: H' , past locations after outliers removed

```

1:  $P \leftarrow$  days lapsed for each  $l \in H$ 
2:  $R \leftarrow null$ 
3:  $curLoc \leftarrow H_1$ 
4:  $curDay \leftarrow P_1$ 
5: for  $i \leftarrow 2$  to  $|H|$  do
6:   if  $H_i = curLoc$  and  $P_i - curDay \leq 3$  then
7:     add  $H_i$  to  $R$ 
8:   else
9:      $curLoc \leftarrow H_i$ 
10:     $curDay \leftarrow P_i$ 
11:   end if
12: end for
13:  $H' \leftarrow$  remove all  $l \in R$  from  $H$ 
14: return  $H'$ 

```

To remove repetitive locations, first we associate each past location with the number of days between the message containing the location and the current message (line 1). Then we move along the timeline, making each location as a cursor (line 5). If the following location is the same as the current location and the time lapsed is within three days, we mark the location as an outlier (line 6), otherwise we move the cursor to the following location (line 9, 10).

The final inference of the location is thus the median location of all remaining locations extracted from the user's past tweets, after outliers are removed. The median location is chosen as the location that has the smallest total distance to all other locations. This median location can also be considered as the *centroid* in the clustering problems.

5 Realtime event monitoring

Events such as car crashes and shooting incidents attract attention easily, especially if they happen in urban areas. In areas where Twitter is popular, such events usually trigger multiple

immediate observation reports posted to Twitter. By aggregating these reports, we can potentially detect the event and raise early alarms.

The final module of SNAF focuses on aggregation-based event detection, which uses the location inference described above to find out the correct location of the event. The aggregator is based on the *connected component* of the graph theory. In our approach, a connected component links reports that are geographically close to each other. Reports and their locations are added to the graph one-by-one. An alarm is raised to indicate that an event is detected, if the connected component after adding a new report now has enough connected nodes. Figure 5 illustrates this process. We set 5 as the number of reports needed for raising an alarm, because if we have 50% report classification accuracy, $1 - 0.5^5 > 0.95$, we can confidently alarm an incident with 5 reports.

The figure indicates observation reports and their geographical locations. (a) Adding report A. (b) A is connected to the nearest report. The connected component now has five nodes, so an alarm is raised, with location of the event calculated as the mean location of nodes A, B, C, D, and E.

The relevant information such as all reporting tweets for the event are written to a log file when an alarm is raised, which can be then exported to an online platform such as a website, making detected events publicly accessible in realtime. To maintain the timeliness of the monitoring system, we only store the most recent tweets, and reports older than a day are removed. Algorithm 9 shows details of our method.

To monitor an event, we incrementally added the detected reports from Twitter data stream to a graph (line 2). For each added report, we calculated the distance between the new report to the nearest report in the graph, and if the distance is smaller than a threshold, an edge is added between the two reports (line 3 - 5). Then we check the connected component the new report is connected to (in case no edge is created for the new report, the connected component has only one node), and if the number of nodes in the connect component reaches five, we raise an alarm and record the time, location and the reporting tweets (line 8 - 13). We also remove old reports once every hour to keep the information update (line 15 - 17).

Our event detection can leverage distance-based connected components because we are confident about the accuracy of our location inference at fine granularity. Since the reports are added to the system individually, SNAF is also capable of processing realtime streams, making it a realtime event detection system.

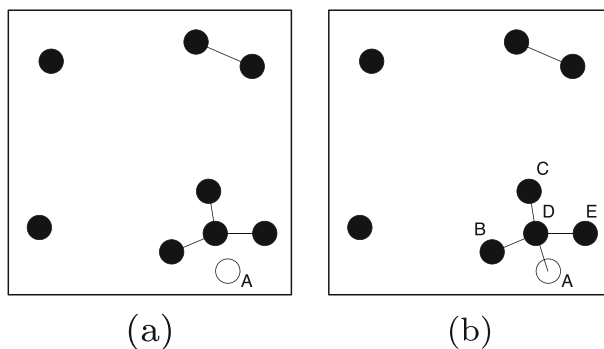


Figure 5 Event detection based on connected components

Algorithm 9 Event Detection and Alarming

```

1: initialize an empty graph  $G = \{V, E\}$ 
2: for each incoming report  $r$  do
3:   find  $p \in V$  geographically nearest to  $r$ 
4:   if distance from  $r$  to  $p$  is less than  $\theta$  then
5:     add  $\{r, p\}$  to  $E$ 
6:   end if
7:   add  $r$  to  $V$ 
8:    $CC \leftarrow$  the connected component  $r$  is connected to
9:    $s \leftarrow$  the number of nodes in  $CC$ 
10:  if  $s = 5$  then
11:    compute location as the mean of locations of all nodes in  $CC$ 
12:    output an alarm, record time, location, and connected nodes to the log file
13:  end if
14: end for
15: for once each hour do
16:   remove all nodes older than 24 hours from  $G$ 
17: end for

```

6 Experimental analysis

We verified our method of report filtering, location inference, and event monitoring in separate experiments. First we tested our report filtering method effectiveness by running the method on controlled and crowd-sourced datasets. Then we tested our location inference method accuracy by running location inference on two datasets. Finally we run our event detection algorithm and manually examine the results. In this section, we discuss our experimental setup and results.

6.1 Report filtering effectiveness

We prepared three datasets for testing the effectiveness of our report filtering method. First we manually collected and labelled two tweet sets, containing the keyword *hailstorm* and *accident*, respectively, according to whether each tweet is a personal observation of the event. The controlled dataset contains a total of 1,629 tweets. Then we used a publicly available crowd-sourced dataset that contains eyewitness and other relevant tweets about crisis events, such as flood and riot. The crowd-sourced dataset contains 3,646 tweets.

6.1.1 Experiment setup

For setting up our method, we deployed an existing implementation for POS tagging. After comparing several existing POS tagging implementations including OpenNLP and LingPipe, we chose StanfordNLP POS module to run our POS tagging because it is relatively fast and provides a high tagging accuracy of around 95% [18].

For parameter θ in Algorithm 1, we chose the first quartile of overall objectivity in the dataset for all experiments, which generally provides good results. For parameter δ , we compared three different values, including 0.9, 0.8 and 0.5. To ensure the consistency of the experiments, instead of randomly choosing initial values for the particles in Algorithm

3, we chose combinations of evenly distributed values for the four attributes as the initial values, i.e., 0.2, 0.4, 0.6, 0.8, and 1. Our analysis shows that randomly initialized particles provide similar results. For user profiling, up to 1,000 recent tweets were collected for each user using Twitter Timeline API.

6.1.2 Baseline methods and comparison metrics

We compared our approach with three baseline filtering strategies, namely, *Accept All*, *Sakaki filter*, and *Sriram*. *Accept All* takes all tweets in the dataset as the positive for personal observations. *Sakaki classifier*, proposed by Sakaki et al. in [24], is a supervised method that deploys a Support Vector Machine (SVM) classifier with linear kernel built on manually annotated training data. Among the three feature sets proposed in [24], we implemented the reportedly most effective set, Feature Set A, which is based on word counts and keyword positions. We deployed an existing SVM implementation in an R language package called *e1071*⁷. We used a weighting function according to class imbalance to ensure optimal performance of the classifier. The performance of the Sakaki classifier was measured using the three-fold cross validation. One drawback of the Sakaki classifier is that it requires the presence of a keyword. The user profiling in our approach, though, does not have this requirement.

The *Sriram classifier*, proposed by Sriram et al. in [28], is also a supervised method that is based on eight features and the Naive Bayes model. The eight features include author name, use of slang, time phrase, opinionated words, and word emphasis, presences of currency signs, percentage signs, mention sign at the beginning and the middle of the message. The evaluation is based on the five-fold cross validation. The *Sriram classifier* is shown to be effective in classifying tweets into categories such as news, opinions, deals and events, but has not been tested in other applications.

All datasets for evaluation were manually annotated according to whether each tweet is a personal observation of an object or event of interest, which were considered ground truth in our experiments. The output of the filtering methods were compared with the manual annotations. If a filtering output is positive in manual annotations, it is considered a *true positive*. We use *precision*, *recall* and *f-value* as the measurements of filtering accuracy, where given the set of positive filtering results P and the set of true positives in the dataset TP , The $precision = \frac{|P \cap TP|}{|P|}$, $recall = \frac{|P \cap TP|}{|TP|}$, and $f\text{-value} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$.

6.1.3 Effectiveness on controlled datasets

We first tested our method on two controlled datasets. We collected a dataset of around 5,000 tweets containing the keyword *hailstorm* during August, 2015, and a dataset of around 5,000 tweets containing the keyword *car accident* during September, 2015. After removing retweets and tweets containing links, we manually labelled the remaining tweets as positive or negative examples, according to whether the message is about a direct observation of a hailstorm or a car accident. The resulted *hailstorm* dataset contains 675 tweets, with 251 positive examples and 424 negative examples. The labelled *accident* dataset contains 954 tweets, with 347 positive examples and 607 negative examples.

We tested the filtering methods on the two datasets. Accuracy results for the baseline methods, lexical analysis-only filtering (LX), and lexical analysis combined with personal

⁷<https://cran.r-project.org/package=e1071>

Table 6 Filtering accuracy for hailstorm and car accident datasets

	Accept All	Sakaki	Sriram	LX ($\delta = 0.9$)	PA ($\delta = 0.8$)	PA ($\delta = 0.5$)	PA
hailstorm dataset							
precision	0.37	0.43	0.37	0.53	0.62	0.64	0.70
recall	1	0.70	0.98	0.80	0.76	0.71	0.46
f-value	0.54	0.53	0.54	0.63	0.68	0.67	0.56
car accident dataset							
precision	0.38	0.50	0.44	0.53	0.58	0.59	0.60
recall	1	0.73	0.84	0.76	0.74	0.69	0.43
f-value	0.55	0.60	0.57	0.63	0.65	0.64	0.50

account filtering using three δ values, $PA(\delta = 0.9)$, $PA(\delta = 0.8)$, and $PA(\delta = 0.5)$, are presented in Table 6.

As shown in the table, the Accept All strategy captured all the positives in the annotations and had the maximum recall of 1. All other methods improved the precision by sacrificing the recall to some degree. Personal account filtering with δ set to 0.9 achieved the highest overall performance, indicated by the highest f-value. Using lexical analysis only and PA with $\delta = 0.9$ and $\delta = 0.8$ all performed better than the Sakaki classifier and the Sriram classifier, the latter of which provided almost no filtering effect in the hailstorm dataset. Setting δ to a lower value improved the precision but also lowered the recall. When setting δ to 0.5, PA achieved the highest precision, while still held a relatively high f-value. The performance of all methods were consistent across two datasets, with LX improving precision from the Accept All strategy by around 15% and $PA(\delta = 0.9)$ further improved it by 5%-7%.

6.1.4 Effectiveness on crowd-sourced dataset

We tested our approach on a publicly available dataset produced by Castillo et al. [22], and is available online⁸. The dataset contains around 20,000 tweets related to crisis events, such as the Colorado wildfires and the Pablo typhoon in 2012, and the Australia bushfire and New York train crash in 2013. These crisis tweets were manually annotated by hired workers on Crowdfunder, a crowdsourcing platform⁹. The tweets were labelled according to their relevance to the crisis event, and the type of information they provide into four categories, namely, *related and informative*, *related but not informative*, *not related*, and *not applicable*. The seven information types include Eyewitness, Government, NGOs, Business, Media, Outsiders, and Not applicable.

We consider that the Eyewitness-type tweets in the dataset are personal observations, while other types of tweets are not. Hence we expect our approach to filter Eyewitness tweets from other tweets. With this goal, we re-organized the dataset. First we selected two categories of related tweets from the dataset. Then we selected five information types of tweets: Eyewitness, Government, NGOs, Business and Media. We then produced a list of

⁸<http://crisislex.org/>

⁹<http://www.crowdfunder.com/>

labels, with Eyewitness tweets as positives, and other types of tweets negatives. We also removed retweets from the data. Our labelled dataset had 3,646 tweets with 528 positives.

Since the tweets do not contain a specific keyword, we did not run POS and objectivity analysis. The Sakaki classifier is not applicable without a keyword. As such we ran the originality test in the lexical analysis and the personal account classification, and compared only to the Sriram classifier (Table 7).

The results are similar to previous experiments, where $PA(\delta = 0.9)$ achieved the highest f-value and $PA(\delta = 0.5)$ achieved the highest precision. The lexical analysis was particularly effective for this dataset, improving the precision by around 38%, mainly because the dataset includes a large portion of news messages, which failed the originality test. After the lexical analysis, $PA(\delta = 0.9)$ further improved the precision by 12%. Both LX and $PA(\delta = 0.9)$ significantly outperformed the Sriram classifier.

6.2 Location inference accuracy

We used two other datasets for testing our location inference method. We collected two sets of tweets by containing the keywords *homeless* and *shooting*. The homeless dataset contains around one million tweets dated from August 5 to September 9, 2014. The shooting dataset contains around two million tweets dated from September 24 to October 17, 2014. We run our filtering method on both datasets. Filtering the homeless dataset generated 20,229 reports, 1,331 of which contained GPS data. Filtering the shooting dataset generated 10,216 reports, 359 of which contained GPS data.

6.2.1 Measurements and baseline methods

We set five distance error buckets of 3, 5, 10, 30, and 100 kilometers, and counted an inferred location in a bucket if the difference between the inferred location and the GPS data of the report was within that error range. If the error of the inferred location was more than 100 kilometers, we counted it as failed. The accuracy of a method in an error range is calculated as the percentage of reports counted in that distance error bucket, in all reports considered. If no location information was found for a report, we still counted it as failed. Since we count all the reports, the accuracy we use here is the same as *recall* in information retrieval. Because we can extract a location from most of the reports, the precision would be very close to the recall, and is not included in the measurement.

We tested three of our methods, DKO+BC, NKO+BC, and SEM+BC, as defined in the previous section. We also tested two baseline methods for comparison, the *Mean* method, and the method proposed by Ikawa et al. in [8], which we term *Ikawa* method. The Mean method, which uses the mean coordinate value of all past locations as the tweet location, was investigated in a work by Schulz et al. [26]. The Ikawa method associates messages

Table 7 Filtering accuracy for the crisis dataset

	Accept All	Sriram	LX ($\delta = 0.9$)	PA ($\delta = 0.8$)	PA ($\delta = 0.5$)	PA
precision	0.14	0.32	0.52	0.64	0.64	0.65
recall	1	0.52	0.47	0.50	0.48	0.27
f-value	0.24	0.40	0.47	0.56	0.55	0.38

with locations using messages containing location words, and then infers the location of a new message by matching the message with the trained messages, based on cosine similarity and term frequency. The Ikawa method has two variations, *trained by all data*, and *trained by user*. We chose the *trained by user* variation, because it performs better in smaller error ranges.

6.2.2 Location accuracy results

We applied the location inference methods on 1,331 homeless reports and 359 shooting reports, and generated a location for each report. Figure 6 shows some of the locations identified around North America, with dense areas roughly corresponding to large cities that have dense populations and high Twitter usage.

The accuracy of the proposed inference methods for each of the two datasets is shown in Table 8. The highest accuracy in each distance error bucket is highlighted in bold font. From the table, we can see that for the homeless dataset, SEM+BC achieved the highest accuracy in all error ranges. For the shooting dataset, DKO+BC achieved the highest accuracy in smaller error ranges, while within 100km error range, SEM+BC was more accurate.

Comparing to the mean method, our methods, which essentially cleaned the data before applying median, successfully improved the accuracy by over 30% in smaller error ranges. For the 3km error range the improvement was 36%, and for the 100km error range the improvement was 12%. Our best achieving method was also more accurate than the Ikawa method in all error ranges. The accuracy of the Ikawa method was lower in smaller error ranges than the numbers reported in their paper, because the Foursquare data, which they relied on to get accurate location DB extraction, are no longer available, and our automatically generated gazetteer based on DBpedia was used instead. The accuracy of Ikawa method in larger error ranges was nevertheless higher than the numbers reported in their paper, because we extracted locations from more messages using the gazetteer approach, thus allowed the locations of more tweets to be found, even though the errors are larger.

6.3 Event detection results

In this experiment, we tested the event detection of the SNAF system using the *Shooting* dataset. In particular, we sorted 10,216 reports in chronological order, sent each report to SNAF, from the oldest to the newest, and SNAF generated a number of alarms. We

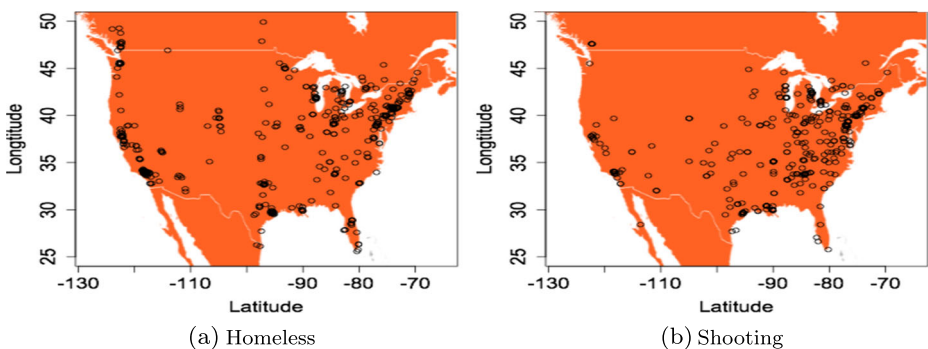


Figure 6 Identified report locations in North America

Table 8 Accuracy of location inference methods

error	$\leq 3\text{km}$	$\leq 5\text{km}$	$\leq 10\text{km}$	$\leq 30\text{km}$	$\leq 100\text{km}$
Dataset: Homeless					
Mean	0.086	0.113	0.192	0.327	0.432
Ikawa	0.084	0.113	0.184	0.284	0.363
DKO+BC	0.109	0.142	0.247	0.374	0.481
NKO+BC	0.113	0.151	0.259	0.383	0.480
SEM+BC	0.117	0.152	0.259	0.392	0.488
Dataset: Shooting					
Mean	0.091	0.116	0.194	0.298	0.387
Ikawa	0.066	0.089	0.155	0.256	0.334
DKO+BC	0.108	0.147	0.233	0.331	0.423
NKO+BC	0.103	0.144	0.233	0.334	0.420
SEM+BC	0.103	0.142	0.222	0.331	0.426

manually examined 100 alarms and found that 54 of them contained reports of actual shooting incidents. Due to the difficulty of finding corresponding news article, we did not evaluate all detected events. Table 9 shows three instances of the detected shooting incidence compared to corresponding news articles.

For each instance, the *time alarmed* is the time of the last report that triggered the alarm. The *earliest news article* is the earliest online news article about the event we found using our best effort. We found these news articles mostly on *Google News*¹⁰, making use of their “sort by date” feature. The *news article time* is the time of the publication of the news. The *time before news article* indicates how much faster our alarm was than the publication of the earliest news article, measured in hours. A positive number means the alarm is faster than news, while a negative one means the alarm is slower. All times are converted to the local time where the event happened. *Supported tweets* shows tweet messages of the reports in the connected component that triggered the alarm.

In the *Indiana State University Shooting* case, although the news reacted quickly and reported less than one hour and a half after the incident, our system was able to capture the incident 9 minutes earlier than the news. From the supporting tweets we also identified a mistake in the news, that the incident happened not at 6:30, but at 16:30. In the *Fern Creek High School Shooting* case, although we were able to detect it just one hour after the shooting, it was half an hour slower than the news. In the third instance, the *Marysville Shooting* on the night of October 15, 2014, which happened around midnight, was not reported by the news until next morning. However, our monitoring system was able to detect it at midnight, only an hour after the shooting. In all three instances, the event locations the system inferred were very close to the reported location, with distance errors between a few hundred meters and 1.5 kilometers. The inferred location for the last event was less accurate because the event involved a series of sub-events each had a different location.

In addition to the three examples discussed, we also found five other instances for which we found corresponding news articles. Based on these instances, on average, our system

¹⁰<http://news.google.com>

Table 9 Examples of detected events

Detected event: Indiana State University Shooting, 27/09/14	
Time alarmed	27/09/14 17:43
Location inferred	39.469222,−87.41235737
Earliest News article	http://www.wthr.com/story/26644661/2014/09/27/isu-reports-shooting-near-lincoln-quad-residence-hall
News article time	27/09/14 17:52
News article location	Indiana State University (39.471345, −87.408071)
Time before news article	+9 minutes
Location error	0.43km
Reporting Tweets	
27/9 16:28 <i>Shooting on campus.. #MoreToBlue</i>	
27/9 16:26 <i>An actual shooting on campus. On family day. What has happened to Indiana State.</i>	
27/9 16:42 <i>ok basically witnessed someone die earlier and now there's a shooting on campus.</i>	
27/9 16:58 <i>Niggas really out here shooting on campus in terre haute</i>	
27/9 17:43 <i>WTF is terre haute coming to there was just a shooting at isu..</i>	
Detected event: Fern Creek High School Shooting, 30/09/14	
Time alarmed	30/09/14 14:09
Location inferred	38.1597,-85.5877
Earliest News article	http://www.tristatehomepage.com/story/d/story/shooting-confirmed-at-louisville-high-school/27354/izZd6FZn8kaM9yx3V2PFoQ
News article time	30/09/14 13:37
News article location	Fern Creek High School (38.1563, −85.5923)
Time before news article	−32 minutes
Location error	0.5km
Reporting Tweets	
30/09 13:28 <i>Damn shooting at fern creek</i>	
30/09 14:09 <i>One student suffered injuries not believed to be life-threatening, WLKY reports.</i>	
30/09 13:57 <i>Fern Creek High still on lockdown after reports of shots fired. Student text parent that there was someone shooting at another person.</i>	
30/09 13:47 <i>Damn, kids shooting at fern creek. Smh</i>	
30/09 13:51 <i>damn, shooting at fern creek</i>	
Detected event: Marysville Shooting, 15/10/14	
Time alarmed	16/10/14 00:38
Location inferred	48.062,-122.163
Earliest News article	http://www.marysvilleglobe.com/news/279423142.html
News article time	16/10/14 07:00
News article location	64th Ave, Marysville (48.059219, −122.144871)
Time before news article	+6 hours 28 minutes
Location error	1.38km
Reporting Tweets	
16/10 00:22 <i>Damn, I guess there was a guy shooting at police and and drove through marysville</i>	

Table 9 (continued)

16/10 00:09	<i>Gunman shot at Granite Falls and Lake Stevens police stations, shooting at pursuing officers in Marysville.</i>
16/10 00:05	<i>Another lake Stevens shooting going on as we speak.</i>
16/10 00:38	<i>That's so scary. I told my dad I heard like 15 gun shots go off and it was that guy shooting up the lake Stevens police department</i>
16/10 00:29	<i>There's a police chase on state and the guy is shooting at the cops #MarysvilleProbs</i>

alarmed the incident 2.8 hours before the earliest news report, with a mean location error of 4.1km. We found that a shooting incident usually will not make it to news article unless a sensitive location or person is involved, such as shooting happened in a school or a university. There are many shooting incidents, while clearly happened based on examining tweets, have no related news reports.

7 A prototype realtime event monitoring system

We designed and implemented a prototype realtime event monitoring system for monitoring shooting incidents, called *SNAF-Shooting*, based on the methods and techniques proposed in this article. We implemented the components shown in Figure 1 using Java 1.7. The Java program continuously listens to Twitter Filter API, and upon capturing a tweet containing the keyword *shooting*, it classifies, infers the location, and aggregates with previous tweets, and outputs an event record if an event alarm is raised. Except for the final event information output, the program runs entirely in memory. We tested the program on a computer with a pair of Intel Core 2 Duo CPU, 7.9 GiB Memory, 621 GiB hard-drive, and runs a 32-bit Linux-based operating system. Setting 1024 MB memory for the Java Virtual Machine is proven to be sufficient for running the program continuously over a week.

We then setup a webpage that contains a PHP code that reads the event record file and displays them on the webpage. The PHP code converts the coordinates of the inferred event location to pixel values, and displays a marker on top of a map shown on the webpage, as can be seen in Figure 7. The numbers above the markers indicates event numbers, and the details of each event, including time, exact location, and reporting tweets, can be found below the map, as shown in Figure 8. We put a “Report Faulty Alarm” button alongside each event to allow the user to provide feedback for faulty event alarms. Each time the webpage is reloaded, the latest information on detected shooting incidents will be displayed on the webpage.

We set 5 as the number of required tweets for raising an event alarm, so the time needed for event detection after the event happened depending on the time of the emergence of the 5th report, and may vary between a few minutes and several hours. Classifying, inferring location, and aggregation for a single tweet, though, generally takes around 10 seconds. Most of this time is spent on waiting on Twitter API's rate limit. Twitter sets a rate limit of 180 calls in 15 minutes, or five seconds per call, for its timeline API. Each timeline API allows retrieval of up to 200 past tweets. We made two calls to retrieved 400 past tweets for location inference of each tweet, and we set a delay of four seconds between each

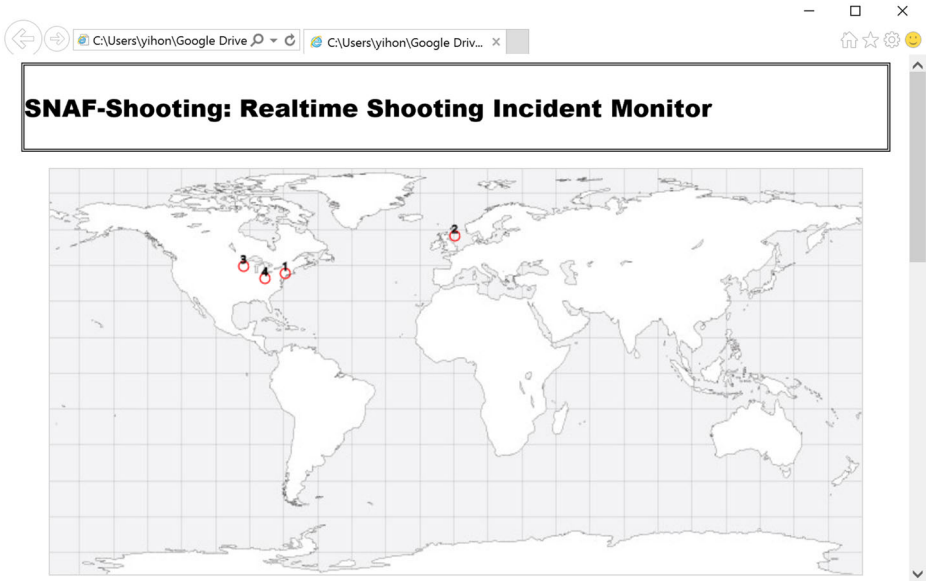


Figure 7 The map view of the event detection system

call, assuming that the location inference itself generally takes more than one second to complete. Actual statistics shows that, for 1,000 location inferences, the average time taken for each inference was 10.8 seconds. The time needed for classifying and aggregation was neglectable, comparing to the time needed for location inference.

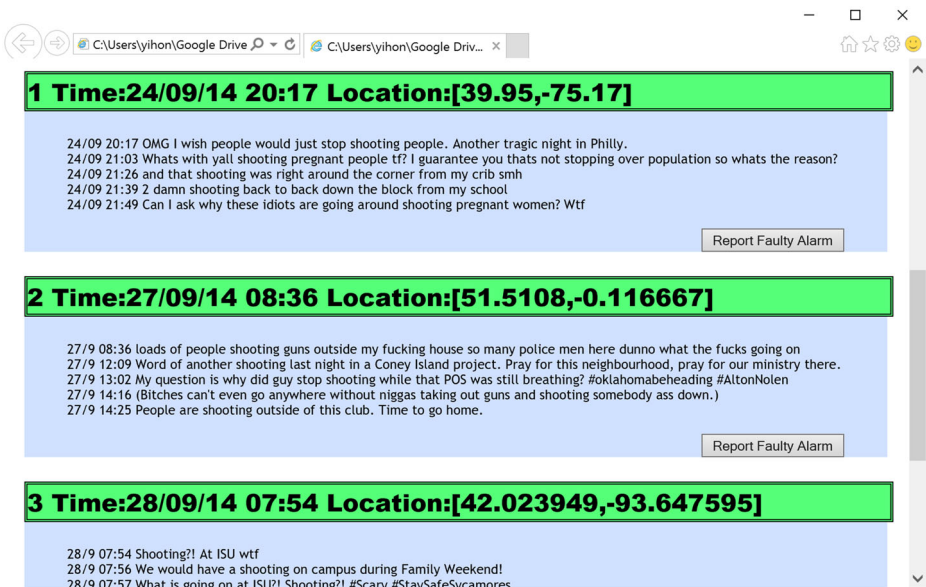


Figure 8 The tweet view of the event detection system

8 Conclusion

Twitter can be seen as a dense sensor network in urban areas, and each Twitter user can be seen as an individual sensor. Such a sensor network can be used for monitoring events and objects of interest, as Twitter users post observations of their immediate environments continuously and constantly. This information can be collected for detecting local events, potentially much earlier than any news reports or official announcements. For such usage, we identify two challenges. The first is to filter observations from other unrelated messages. The second is to fill the missing location information. Unfortunately, the portion of unrelated message on Twitter is large and the location information is sparse. Current works on both message filtering and location inference have not achieved satisfactory accuracy.

In this paper, we propose Sense and Focus (SNAF), an event monitoring system that classifies personal observation reports, and infers fine-grained locations for each report, based on a comprehensive place gazetteer and users' past tweets. By using lexical analysis and user profiling, we improve the report filtering accuracy by around 22%. For location inference, we increase the availability of location information by using location names in tweet messages and users' past tweets. By applying sensor data cleaning techniques, we remove the noises in the location data and improve the accuracy of the location inference over existing location inference approaches. For various error ranges between 3km and 100km, our method improves the location accuracy between 12% and 36%. Taking shooting incidents as the target event, our prototype event monitoring system captures local events and provides accurate location information based on the inferred locations, in some cases with less than 1km error. Based on the effective report classification and message location inference, the event detected by our system are actual event of interest more than half of the time. In the future, we plan to extend our approach for monitoring events with multiple, continuously changing locations.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: *DBpedia: A nucleus for a web of open data*. Springer (2007)
2. Branch, J., Szymanski, B., Giannella, C., Wolff, R., Kargupta, H.: In-network outlier detection in wireless sensor networks. In: *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, p. 51 (2006)
3. Carroll, T.Z.J.: Unsupervised classification of sentiment and objectivity in chinese text. In: *Third International Joint Conference on Natural Language Processing*, p. 304 (2008)
4. Castillo, C., Mendoza, M., Poblete, B.: Information credibility on Twitter. In: *Proceedings of the 20th International World Wide Web Conference*, pp. 675–684 (2011)
5. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: *Proceedings of the 9th International Conference on Semantic Systems*, pp. 121–124 (2013)
6. Graham, M., Hale, S.A., Gaffney, D.: Where in the world are you? geolocation and language identification in Twitter. *Prof. Geogr.* **66**(4), 568–578 (2014)
7. Hong, L., Ahmed, A., Gurumurthy, S., Smola, A.J., Tsioutsoulis, K.: Discovering geographical topics in the Twitter stream. In: *Proceedings of the 21st International World Wide Web Conference*, pp. 769–778 (2012)
8. Ikawa, Y., Enoki, M., Tsubori, M.: Location inference using microblog messages. In: *Proceedings of the 21st International World Wide Web Conference Companion*, pp. 687–690 (2012)
9. Jeffery, S.R., Alonso, G., Franklin, M.J., Hong, W., Widom, J.: *Declarative support for sensor data cleaning. Pervasive Computing*. Springer (2006)
10. Ji, Z., Sun, A., Cong, G., Han, J.: Joint recognition and linking of fine-grained locations from tweets. In: *Proceedings of the 25th International Conference on World Wide Web*, pp. 1271–1281 (2016)

11. Kennedy, J.: Particle swarm optimization. *Encyclopedia of Machine Learning*, pages 760–766. Springer (2010)
12. Kinsella, S., Murdock, V., O’Hare, N.: I’m eating a sandwich in glasgow: modeling locations with tweets. In: *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, pp. 61–68 (2011)
13. Knox, E.M., Ng, R.T.: Algorithms for mining distance based outliers in large datasets. In: *Proceedings of 24th International Conference on Very Large Data Bases*, pp. 392–403 (1998)
14. Kwon, S., Cha, M., Jung, K., Chen, W., Wang, Y.: Prominent features of rumor propagation in online social media. In: *Proceedings of 13th International Conference on Data Mining*, pp. 1103–1108 (2013)
15. Li, C., Sun, A.: Fine-grained location extraction from tweets with temporal awareness. In: *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 43–52 (2014)
16. Li, R., Lei, K.H., Khadiwala, R., Chang, K.-C.: TEDAS: A Twitter-based event detection and analysis system. In: *Proceedings of 28th International Conference on Data Engineering*, pp. 1273–1276 (2012)
17. Lingad, J., Karimi, S., Yin, J.: Location extraction from disaster-related microblogs. In: *Proceedings of the 22nd International World Wide Web Conference Companion*, pp. 1017–1020 (2013)
18. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford coreNLP natural language processing toolkit. In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60 (2014)
19. McMinn, A.J., Moshfeghi, Y., Jose, J.M.: Building a large-scale corpus for evaluating event detection on Twitter. In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pp. 409–418 ACM (2013)
20. Moon, T.K.: The expectation-maximization algorithm. *IEEE Signal Process. Mag.* **13**(6), 47–60 (1996)
21. Mukherjee, S., Weikum, G., Danescu-Niculescu-Mizil, C.: People on drugs Credibility of user statements in health communities. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 65–74 (2014)
22. Olteanu, A., Castillo, C., Diaz, F., Vieweg, S.: CrisisLex: A lexicon for collecting and filtering microblogged communications in crises. In: *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media*, pp. 376–385 (2014)
23. Sakaki, T., Okazaki, M., shakes, Y.M.: Earthquake Twitter users: Real-time event detection by social sensors. In: *Proceedings of the 19th International World Wide Web Conference*, pp. 851–860 (2010)
24. Sakaki, T., Okazaki, M., Matsuo, Y.: Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Trans. Knowl. Data Eng.* **25**(4), 919–931 (2013)
25. Santorini, B.: Part-of-speech tagging guidelines for the penn treebank project (3rd revision). Technical Report MS-CIS-90-47 University of Pennsylvania Department of Computer and Information Science Technical (1990)
26. Schulz, A., Hadjakos, A., Paulheim, H., Nachtwey, J., Mühlhäuser, M.: A multi-indicator approach for geolocalization of tweets. In: *Proceedings of the Seventh International Conference on Weblogs and Social Media*, pp. 573–582 (2013)
27. Sheng, B., Li, Q., Mao, W., Jin, W.: Outlier detection in sensor networks. In: *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 219–228 (2007)
28. Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., Demirbas, M.: Short text classification in Twitter to improve information filtering. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 841–842 (2010)
29. Starbird, K., Maddock, J., Orand, M., Achterman, P., Mason, R.M.: Rumors, false flags, and digital vigilantes: Misinformation on Twitter after the 2013 boston marathon bombing. In: *Proceedings of the iConference 2014*, pp. 654–662. iSchools (2014)
30. Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Online outlier detection in sensor data using non-parametric models. In: *Proceedings of the 32nd International Conference on Very Large Data Bases*, pp. 187–198 (2006)
31. Tausczik, Y.R., Pennebaker, J.W.: The psychological meaning of words: LIWC and computerized text analysis methods. *J. Lang. Soc. Psychol.* **29**(1), 24–54 (2010)
32. Unankard, S., Li, X., Sharaf, M., Zhong, J., Li, X.: Predicting elections from social networks based on sub-event detection and sentiment analysis. In: *Proceedings of 15th International Conference on Web Information Systems Engineering, Part II*, pp. 1–16. Springer (2014)
33. Unankard, S., Li, X., Sharaf, M.A.: Emerging event detection in social networks with location sensitivity. *World Wide Web* **18**(5), 1393–1417 (2015)
34. Wen, Y.-J., Agogino, A.M., Goebel, K.: Fuzzy validation and fusion for wireless sensor networks. In: *Proceedings of the ASME International Mechanical Engineering Congress*, pp. 727–732 (2004)

35. Zhang, Y., Meratnia, N., Havinga, P.: Outlier detection techniques for wireless sensor networks A survey. *Communications Surveys Tutorials*, IEEE **12**(2), 159–170 (2010)
36. Zhang, Y., Szabo, C., Sheng, Q.Z.: Sense and focus: Towards effective location inference and event detection on twitter. In: *Proceedings of the 16th International Conference on Web Information Systems Engineering Part I*, pp. 463–477 (2015)
37. Zhang, Y., Szabo, C., Sheng, Q.Z.: Improved object and event monitoring on twitter through lexical analysis and user profiling. In: *Proceedings of the 17th International Conference on Web Information System Engineering* (2016)
38. Zhang, Y., Szabo, C., Sheng, Q.Z., Fang, X.S.: Classifying perspectives on twitter Immediate observation, affection, and speculation. In: *Proceedings of the 16th International Conference on Web Information Systems Engineering Part I*, volume, pp. 493–507 (2015)