Robust Social Event Detection in Twitter

Bilkent University CS533 Spring 2016 Project Final Report

Fouad Amira
Bilkent University, CS Dept
fouad.amira@bilkent.edu.tr

Selim Eren Bekçe
Bilkent University, CS Dept
eren.bekce@bilkent.edu.tr

ABSTRACT

Twitter is used by millions of people every day around the world and people tend to report on their surroundings more than ever. In this paper, we investigate solutions to detect political social events by analyzing large number of tweets, which may prove to be faster and more reliable than some news agencies. We will be investigating the recent social events in Turkey with a input data size of 160 million tweets. Robustness is a vital element in the analysis of such big number of tweets so we look at ways to optimize processing speed and filter out deficient data. We convert filtered data into statistical models and employ an anomaly detection algorithm to detect such events. Evaluation is done by looking at the precision of signaled events and improving the detector parameters.

1. INTRODUCTION

Twitter is a microblogging platform people use for expressing themselves and their surroundings. It recently acquired important place in many people's lives as it is also commonly used for communication around particular topics and people. Another very important use for Twitter is emergencies and events. It is very convenient to use from smartphones, it is also very easy to attach photos, and get instantaneous feedback from their followers. This convenience allows people to capture and report the events happening in their surroundings for other people to see and understand about the event. Events can be very broad in sense it can be anything from political protests to artistic occasions. Some events are more prominent than others and the urge to share them with other people will be more prominent.

Recently in Turkey, there were many political events all over the country. There was also a suppression on news agencies so they were blocked by the government to share news on national television and newspapers, which are the prominent source of information for many of the population, most people simply were unaware of those protests for an extended period of time because there was no sign of them on the news.

With the help of social media platforms, including Twitter, it was possible for vast majority to understand and get information on those events. People were sharing photos and updates on their surroundings and it allowed tracking the events on the Twitter in near real time. The reliability of those news is subject to debates but as more number of people tend to share same information, it becomes more likely to be correct. This proved that social media platforms can actually be used for real time news source. In this work, we inquired whether it is possible to detect those events automatically by using Tweets. We employed statistical models to detect and report significant rises in number of event related tweets.

2. LITERATURE REVIEW

We have looked at two papers related to Event Detection using Twitter.

"TEDAS: A Twitter-based Event Detection and Analysis System" [1]

This is an intuitive Twitter event detection infrastructure which mainly focuses on the detection crime and disaster related events with respect to spatial and temporal locality. They developed keyword based tweet streaming, keyword rule generator, related tweet classification, location prediction heuristics for this task. Their infrastructure has both offline and online components.

"Twitter earthquake detection: earthquake monitoring in a social world" [2]

They ask the question whether it is possible to detect earthquakes by looking at tweets with respect to spatial and temporal locality. They developed an online anomaly detection algorithm which signals an event if there is an abnormal increase in targeted tweets. They then compare signaled events to real earthquakes for validation.

Earthquake detection and social event detection are very similar topics at their core. For example, earthquake related tweet is likely to have 'earthquake' in its content. A social event related tweet is also likely to have related keywords. Earthquakes can be detected by the increase of such matching tweets in the incoming stream and social events are also like that.

There are several papers on Event Detection topic. We have picked these papers because in some extent, their methods and way of thinking have good correlation with our proposed methodology for this task.

3. WORK DONE

There are many steps involved when doing "a Twitter experiment". Since tweets are basically unstructured human written text with only a few metadata, we need to make use of Information Retrieval methods for detecting events. We will describe a series of steps to transform collection of tweets into list of detected events. Robustness is a very important factor here, as any data created by humans are prone to errors, empty and erroneous values. Since we are vastly applying text processing methods on tweets, robustness and data cleaning become very important.

Our methodology can be roughly summarized as:

a) Serialization and compression to process raw data, filter deficient data out and create semi-processed data form to efficiently apply processing steps on.

b) Cleaning to filter on desired date range and tweet types, discarding non-relevant ones.

c) Preprocessing and tokenization steps applies IR and text transformation operations for robustness, such as lowercase conversion and stemming. Vital for free text operations.

d) Classifier is the step where the last piece of filtering happens. It selects 'eventful' tweets and discards the rest. Its logic depends on the output of tokenization part.

e) Event Detector part is the main processing part where the events are detected out of the list of tweets generated from the classifier part.

We have obtained 160 million raw Turkish tweets collected over years 2013 to 2015 [3]. This particular period was important because there were many social events in Turkey.

We will first describe an offline algorithm which works on high volume of tweets in an efficient manner. It would be used to detect previous events occurred. We will then describe a novel online algorithm design which works through connecting to Twitter Streaming API and report events as they naturally occur.

These two algorithms are very similar in nature and they share the preprocessing part, the only noticeable difference is the event detector part.

3.1 Offline Algorithm

First algorithm we've designed is an offline algorithm. In following subsections, we explain some details about the major components of our proposed methodology. Figure 1 shows the components and steps in the Offline Algorithm.

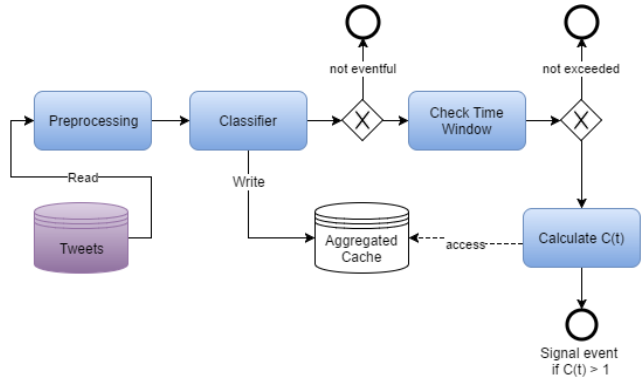

Figure 1: Offline Algorithm steps

3.2 Serialization and Compression

We obtained our dataset with 160 million tweets was stored in raw JSON form. It was difficult to run processing on because of the huge file size and parsing overheads. It also contained lots of raw and redundant data so we discarded unnecessary raw parts then applied binary serialization and compression to obtain semi-processed tweets to efficiently store and perform processing on, as can be seen on Figure 2. This process took half a day on a single computer but it was very effective such that the data size was tremendously reduced from 500 GB to 9.7 GB.

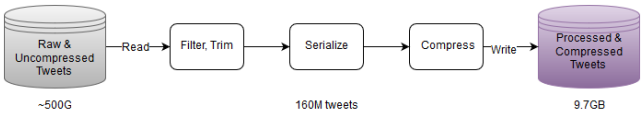

Figure 2: Serialization and compression steps

3.3 Cleaning

'Retweet's in Twitter is like carbon copies of other (original) tweets, which usually mean that the 'retweet' author has same or similar opinion on related tweet so that he or she chose to 'retweet' it. This brings unneeded redundancy to the model and it amplifies the number of false positives in case an unrelated tweet is a popular one. Therefore, we chose to discard retweets and decided only to take original tweets into account while detecting events.

In our findings, we've found that nearly 30% of tweets are retweets, which is a huge number.

A 'Reply' to another tweet is also possible in Twitter. Replies usually indicate a direct conversation between some parties or one-side mentions to particular notable person or hub in order to ask a question or express opinion about a common subject. These kind of tweets (both replies and mentions) are also not relevant to our subject as we are seeking immediate responses to social events. Thus, we also discard these kind of tweets.

3.4 Preprocessing and Tokenization

After cleaning out unneeded tweets, we apply following preprocessing on the tweet text, respectively.

Replace all URLs with <URL> identifier: The content of URLs is not relevant in our experiment. Moreover, since we will tokenize the tweet text with non-alphanumeric characters, the characters in URLs ':', '/' will get replaced by and it will be decomposed into multiple tokens, which is not desired.

Since replied tweets were dropped in previous stage, we are not doing anything special for the user mention tags.

Convert tweet text into lowercase: This is important to neutralize the text. Note that since our target language is Turkish, we do this with respect to Turkish characters ('I' → 'ı').

Convert non-alphanumeric characters to spaces: Our classifier looks at concrete words, not punctuation or other non-printable characters. Therefore, we replace all non-alphanumeric characters with spaces. Note that we are using a Unicode regex class for this purpose so that it does not replace accented or language specific characters like 'ğ' or 'â'. Then we remove excess spaces from the tweet text so multiple spaces become one space. Note that this process also removes hashtags '#' from the tweet text. This is OK as hashtag context is not relevant in our event detector.

Next we use Zemberek [4] for stemming and tokenizing purposes. It tokenizes given (Turkish) text with respect to its stems so that it is much robust for our classifier to find relevant keywords in the corpus ('protestolar' → 'protesto')

3.5 Classifier

Our classifier is the final step to decide whether a tweet is 'eventful' or 'not eventful'. We look at the generated tokens and look for following tokens, which are relevant for protests and social events: [protesto, eylem, toma, saldırı, direniş, barikat].

Even with these handful of words, the number of false positives is still high. An improvement is to give importance to the present tense clauses ('-yor') which implies something is going on. For example, the following sentence contains present tense clauses and it reports an ongoing social event in action:

"Ankara Kızılay'da madenci heykeli önünde Soma'daki iş cinayeti protesto ediliyor."

3.6 Histogram Generation

After Classifier part, we generate 'histogram' of 'eventful' tweets by aggregating them into fixed time windows and counting them in every interval. The time interval is called the Grouping Factor. The output is a very compact representation of the input data which only consists of the interval timestamp and the number of tweets in that interval. It will be consumed in the Event Detector part.

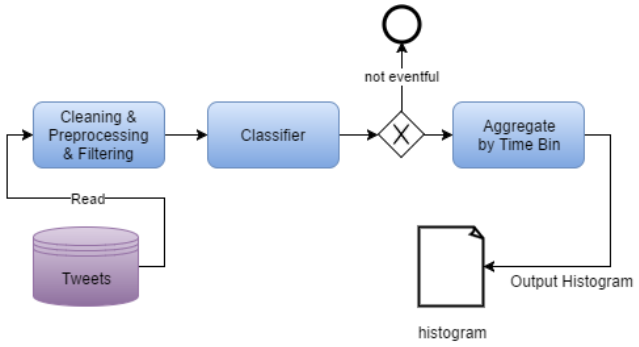

Figure 3: Histogram Generation steps

Processing all tweets and classifying takes more time than evaluating histogram. Therefore, the advantage of producing an output at this point is that we are now able to change the parameters of the Event Detector part without having to run Classifier again. We call these series of steps including cleaning, preprocessing, tokenization and classifier as Histogram Generation, whose details can be observed in Figure 3.

3.7 Event Detector

The core of our algorithm consists of detecting events. We have constructed statistical models by reading generated histogram and employ an anomaly detection algorithm and signals an event when a threshold in number of tweets is reached.

After reading the generated histogram of tweets we employ the characteristic function [2] which is defined as:

$$C(t) = STA/(mLTA + b)$$

A detection is declared when C(t) exceeds 1. The STA is the short term average taken 15 minutes before the time of the event and LTA is the long term average taken 5 hours before the event. m and b are tunable parameters used to determine the sensitivity of the detector. Low values result in a more sensitive detector.

LTA helps with approximating the background noise of Tweets, possibly because of old events that happened long time before the detection time, C(t) requires higher signal levels (STA) to trigger at higher noise levels (LTA).

We require C(t) to drop to a threshold until a new event can be detected. Without this threshold, an event spike signals many events until LTA stabilizes, which may happen several intervals later and until it stabilizes it will signal semantically same event once again, which is undesirable. We have used drop threshold value of 0.25 and 0.5.

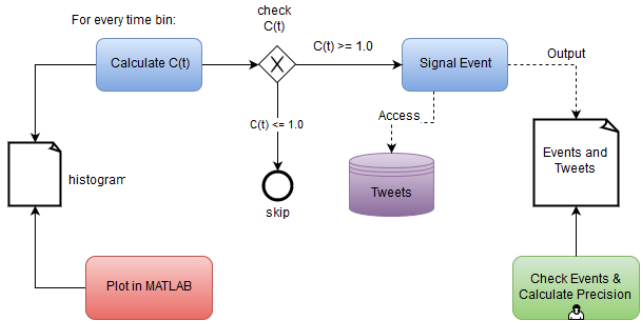

Figure 4: Characteristic Function and Event Detector steps

Figure 4 shows the final steps of our algorithm. It starts by reading the generated histogram and calculating histogram for every time window. It is important to note that since histogram file only contains non-zero intervals, it is customary to fill empty intervals before calculating C(t) values.

After an event is declared for an interval, we also output the related tweets by contacting the tweet store for evaluation to be applied later in the chain.

3.8 Online Algorithm

Online Algorithm was also designed to prove that the core of the event detection algorithm can be applied effectively to online case. The major difference between the online and offline algorithms is that the source of the tweets is not stored on local storage; instead it is received in real time from a Twitter streaming API. The STA/LTA semantics work nicely with streaming data; we only need to store the last LTA window as shown in Figure 5. At this phase it is important that the preprocessing and tokenization phases are optimized to work efficiently so that they can sustain a good amount of real life data.

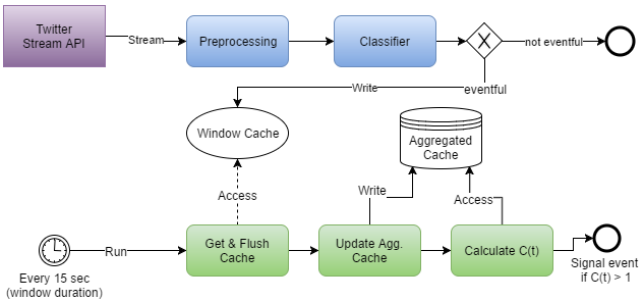

Figure 5: Online Algorithm steps

After storing an LTA worth amount of data in the cache, the algorithm continues the same way as in the offline algorithm. However, the cached data needs to be flushed after processing it in order to make space for new data to take place and be analyzed.

4. EXPERIMENTAL RESULTS

4.1 The Test Procedure

We tried to detect the events in the time period between 01.04.2013 – 01.01.2014. we chose this period because our twitter dataset is Turkish and the specified period is a hot period in Turkey and many political events happened in that time frame.

In order to visualize the data, we wrote a Matlab script that prepares the histogram's data for a proper visualization as it can be seen in Figure 3. The script also implemented the C(t) function and plotted the results and placed the events that are above a certain threshold on top of the C(t) function values.

During the run, we processed 25.710.914 Tweets, 91.085 of these Tweets were related to political events within the previously mentioned time frame. Using a grouping factor of 5 minutes and LTA and STA values of 5 hours and 15 minutes respectively, we obtained the histogram shown in Figure 6.

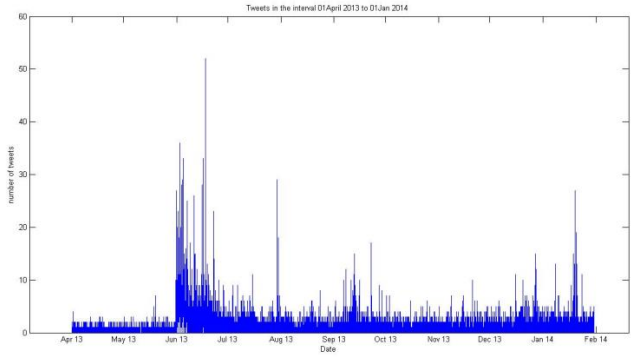

Figure 6: Histogram of tweets between 04.2013 – 01.2014

As it can be seen in Figure 6, the number of eventual events is high in the period between May 2013 and September 2013 since it's a hot period in the Turkish history and a lot of political events and protests happened in that period.

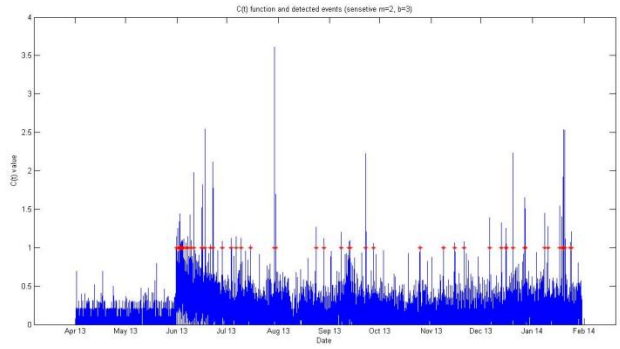

Figure 7: C(t) function with m=2 b=3

After generating the histogram, we applied the C(t) function over the grouped tweets we obtained in the previous step. The result of the C(t) function decides whether an event happened or not, so if the value of the C(t) function gets higher than one then an event is triggered. Please note that, for two consecutive events, the C(t) value after the first one must drop to a certain threshold before the second event is considered a valid event. This is done because we want to guarantee that the two detected events are separate events and any remaining tweets from the first event have no significant influence in triggering latter events. Figure 7 shows the result of applying the characteristic function with m= 2 and b = 3 (sensitive detector). Figure 8 shows the zoomed version of Figure 4 on the period of time with maximum number of events.

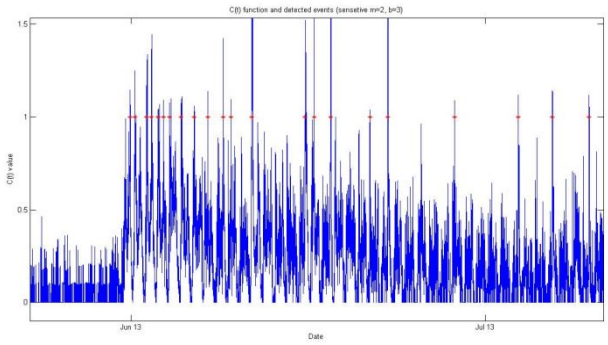

Figure 8: C(t) function with m=2 b=3 (zoomed version)

The characteristic function contains two tunable parameters: m and b, these parameters control the liberty of the detector, the smaller the value of m and b the more sensitive the detector is and as a result of that the number of detected events increases with a higher portion of false positive events. However, when the value of these parameters is high, then the detector is more conservative and the detected events have higher probability of being true positive. Figure 9 shows the C(t) function and the triggered events for the values of m and b of 4 and 6 respectively.

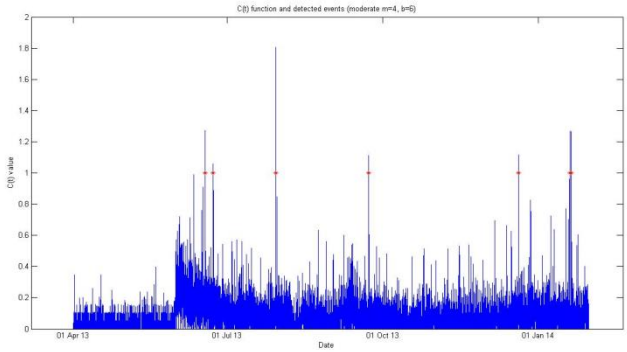

Figure 9: C(t) function with m=4 and b=6 (moderate)

Table 1 shows the count of detected events for different values of m and b and different threshold values.

Table 1: Number of generated events with different m and b values

M	b	C(t) drop thr	Count
2	3	0.5	69
2	5	0.25	24
4	6	0.25	7

4.2 Variable Tuning and Optimization

4.2.1 Tuning the Grouping Factor

In order to optimize the result, we ran the algorithm again with different grouping factors to generate different histograms and hence different results of the characteristic function. The grouping factor is the time period in which the Tweets are grouped and showed in the histogram. Figure 6 shows the histogram for the tweets using a grouping factor of 5 minutes, Figure 10 shows the C(t) function for tweets with grouping factors of 10.

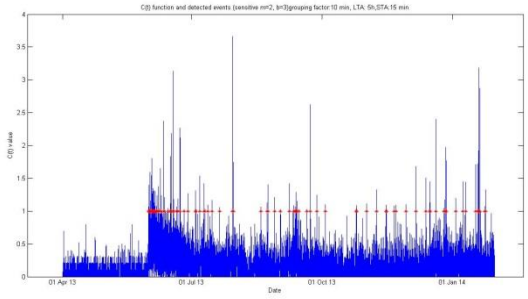

Figure 10: C(t) with groping of 10 min and m=2 b=3

In Figure 10 we can notice that the number of detected events for the same period has risen from 68 to 134 meaning that the detector became more sensitive after increasing the grouping interval from 5 minutes to 10 minutes. However, when we decreased the grouping window to 1 minute, the detector was not able to detect the events properly and only little number of events was detected as shown in Figure 11.

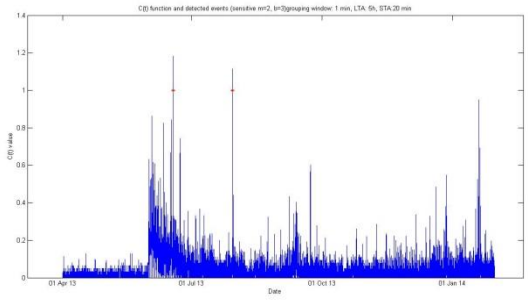

Figure 11: C(t) function with groping of 1 minute and m=2 b=3

4.2.2 LTA and STA

As we mentioned earlier, LTA is the long term average which tries to eliminate the noise in the background while STA is the short term average for checking the events with high count values. We tried to use different values for LTA and STA and compare the results. Figure 7 shows the C(t) function with LTA = 5 hours and STA = 15 minutes. Figure 12, you can see the value of the characteristic function with LTA = 2 hours and STA = 10 minutes. By increasing the value of LTA and STA to 25 hours and 1 hour respectively we obtain the result shown in Figure 13. Note that the number of detected events for using the new values of LTA and STA and the same values of m and b has dropped to 18 compared to 68 which was the count of detected events of the detector with LTA = 5 hours and STA = 15 min as shown in Figure 7.

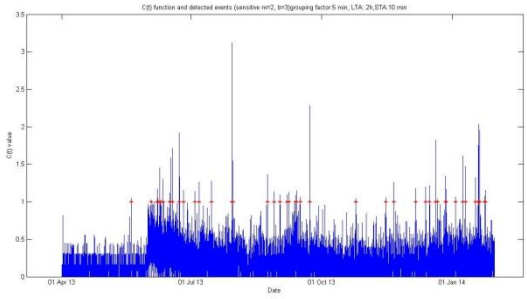

Figure 12: C(t) function for LTA = 2 hours and STA = 10 min

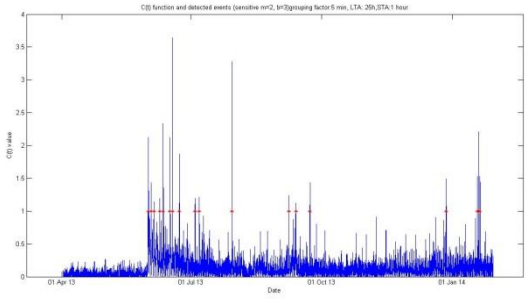

Figure 13: C(t) function for LTA = 25 hours and STA = 1 hour

5. EVALUATION

After detecting the events in our dataset, we evaluated the precision of our results by checking if the events actually happened in real life, we did that by checking the dates of the events manually and we counted the true positive events against false positive events.

We evaluated precision for one result set (m=2, b=5, STA=15mins, LTA=5hrs) by manually validating every signaled event. There were 16 valid events out of 24 reported, which results in a precision value of 2/3.

It is observed that some reported events contain spam tweets and hashtags. By applying some spam filtering techniques, as stated in Future Work section, the precision can be improved furthermore.

Table 2: Tweet samples from one event

Id	Tweet Text
364845589659254784	sükrü saraçoglunda her yer taksim her yer direnis sloganlari
364845474114576385	her yer taksim her yer direnis sükru saraçoglu inliyor
364845494712803328	kadiköy de her yer taksim her yer direnis sesleri halkin takimiyiz
364845693296312320	sükrü saraçoglu nda her yer taksim her yer direnis sesleri helal olsun fenerbahçe salzburg kadiköy
364845710258077697	fenerbahçe maçinda bütün stad her yer taksim her yer direnis diye inliyor

Some sampled tweets from one particular event (happened on 2013-08-06 23:25:00 EEST) can be observed in Table 2, which mentions an ongoing protest in Şükrü Saracoğlu stadium.

6. CONCLUSION AND FUTURE WORK

This project proves that event detection via Twitter is possible via statistical analysis methods and it can produce credible results.

The results we obtained were affected by the spam Tweets and we could increase the precision of our system by employing an algorithm for detecting spam tweets and filtering them out of our result set. This seems like a good chance for integrating more than one project discussed on the class and they handled the issue of spam Tweets. The selection of words used for filtering would highly affect the results and the precision of the system so in the future we can apply learning techniques for selecting the keywords that would yield the best precision.

It is also important to calculate recall by finding a list of social events which occurred at a certain time frame and then investigate how much of these events have been detected by our system. Calculating the recall is a challenge in our case because we need to collect a reliable list of all events happened in a certain time frame, which can be done by crawling some news archives and agencies which can be topic for another project.

It may also be interesting to explore events with respect to spatial locality, in addition to temporal locality. However, most tweets do not have location information as it is a usually a privacy concern for users. Locations can possibly be estimated by considering moving average location of user's last several geo-tagged tweets, user's location tag in his or her profile and locations of the network of the user [1]. Then, those locations may be used to determine the impact radius of detected events and generate event heat maps for better visibility.

7. REFERENCES

[1] Li et al. "TEDAS: A Twitter-based Event Detection and Analysis System", 2012 IEEE 28th Conference on Data Engineering. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6228186&tag=1

[2] Earle et al. "Twitter earthquake detection: earthquake monitoring in a social world", Annals of Geophysics, 54, 6, 2011. http://www.annalsofgeophysics.eu/index.php/annals/article/view/5364/5494

[3] 160M Tweets used for evaluation were obtained from Hakan Ferhatosmanoğlu.

[4] Zemberek Library by Ahmet Akın et. al. https://github.com/ahmetaa/zemberek