On a Keyword-Lifecycle Model for Real-time Event Detection in Social Network Data

Tamás Matuszka\textsuperscript{2}, Zoltán Vincellér\textsuperscript{1,2} and Sándor Laki\textsuperscript{1,2}

\textsuperscript{1}Inter-University Centre for Telecommunications and Informatics, Debrecen, Hungary
\textsuperscript{2}Department of Information Systems, Eötvös Loránd University, Budapest, Hungary
Email: \{tomintt, vzoli, lakis\}@inf.elte.hu

Abstract—Social networks like Twitter and Facebook have gained a significant popularity with people from all parts of the society in the past decade, providing a new kind of data source for novel social-aware applications. A great majority of the users are online all the time, posting real-time information on various topics including unpredictable events. An accident or a natural disaster is often posted on social networks hours before appearing in traditional news. In this paper, we outline a framework for real-time event detection in Twitter data. In contrast to prior works where the absolute or relative changes in the frequencies of some predefined keywords are taken into account, we introduce a lifecycle for each keyword to be observed, expressing their average behavior (e.g. average frequency changes) over time. As a motivation, we show that some keywords exhibit periodic behavior that can be handled by our model. The proposed lifecycle model enables us to define novel temporal features used by our framework in real-time event detection.

Keywords—social networks; twitter; framework; event detection; keywords;

I. INTRODUCTION

Various social media (Twitter, Facebook, Google+, etc.) sites have emerged in the past decade, creating a radically new way of communication between people. These platforms enable users to share and exchange information on real-world events from local to world scale. These events can vary on a wide range: popular concerts, festivals, demonstrations, accidents, traffic jams, disasters, etc. They differ from each other in many aspects, but they have in common that they affect the life of a great mass of people. Since different social networks are also available from mobile phones, a great many people are online all the time, reporting about all the happenings like an accident or a natural disaster in almost real-time. Sometimes these events are posted on Twitter sooner than appearing in traditional news media. This makes social media a good data source for real-time event detection.

However, one can also observe that people generally use Twitter and other sites to share their ideas and thoughts, and not only to report events. In addition, different users may use different languages and phrases to express the same concept. These information sources suffer from noise and one of the key challenges in real-time event detection is to distinguish a post on a real-world event from other messages. In the past few years, some studies \cite{1,2,3,4} on event-detection in social network data have been published, and there are public services like Twitter Monitor \cite{7} that can be applied to identify unusual activities as well. Most of the keyword-based methods are searching for sudden increases in the frequency of some keywords regarded as an evidence of the event’s occurrence.

The aim of our paper is twofold: First, we outline our Twitter-based real-time event detection framework. Second, we introduce a novel keyword-lifecycle model for deriving temporal features constituting the basis of our event classifier. This model is based on the observation that the frequency of some keywords over time shows periodic temporal behavior. For example, if we focus on New York area, the word ‘alarm’ shows a significant increase every morning, resulting in very similar symptoms to an unforeseen real-world event. However, in this case, this sudden increase is related to the lifecycle of people, indicating that most of them wake up in almost the same time as if they are synchronized.

The rest of the paper is organized as follows. In Section II we briefly overview the existing event detection solutions based on social networks. Section III outlines our event detection framework that can process Twitter streams in real-time. Our keyword-lifecycle model is described in Section IV where we also show examples on the periodic behavior of some keywords. In Section V an use case scenario is presented as a preliminary validation of our methodology. Finally, Section VI concludes the results and describes our further plans.

II. RELATED WORK

As described earlier, several social network-based services have emerged in the past few years. The Twitter Monitor \cite{7} is one of them, providing an online monitoring system that allows the detection of sharp increases (bursts) in the frequency of sets of keywords found in Twitter messages. For each burst detected, it provides a keyword-based query represented in a form $A \land B$ can be submitted by users where $A$ is the conjunction and $B$ is the disjunction of keywords or hash-tags. For example, the $Q = ((cinco \land mayo) \land (mexican \party CELEBRATE))$ query was generated by the tweets referring to the „cinco de mayo” Mexican ceremony. The method used by Twitter Monitor has the problem that we need to know the event to be recognized and the related keywords in advance.
Furthermore the primary goal of Twitter Monitor is to detect anomalies that may be related to non-event bursts as well.

H. Becker et al. [6] have presented an approach that does not require any a-priori knowledge on the event to be detected. Their method follows the idea that the majority of users respond to real events by writing tweets, retweets or comments. Based on the time pattern of these responses, real events can be separated from non-event messages. In addition, the users can be handled together with their relationships (friend, followers, etc.) and in this way the activity generated by the event can also be examined. Their approach takes into account the obtained patterns to distinguish relevant and irrelevant events.

T. Sakaki et al. [5] have implemented a Twitter-based earthquake detector, using a semantic approach. This method filters the set of messages with some specific keywords corresponding to the target event (e.g. earthquake, typhoon, traffic jam). This filtering is not sufficient, since irrelevant entries can also be included in the result set. However, the situation is even more complex, as some events may induce tweets long after their original occurrences, not carrying new information or misleading the event detection. To manage this problem, authors used an SVM [8] classifier that divides the tweets into two groups. The first group contains positive tweets related to real target events, while the second group consists of negative tweets that are not taken into account, being independent from the actual event. The authors have prepared well-defined training and testing data sets, where tweets have manually been classified by volunteers. These sets have been used to train and evaluate their event detector.

III. FRAMEWORK

In this section we briefly overview our Twitter-based event detector framework that collects, stores and processes tweets in real-time. The architecture of our framework is outlined on Figure 1. One can observe that different information sources are supported including user and public streams or previously collected streams replayed from our local databases for validation purposes. From the streams each twitter message is added to a processing queue which is fed by one or more processing units. A processing unit has two key roles: First, it assigns tweets to time slots and secondly calculates features for each time slots. We note that our current implementation uses one minute long timeslots, determining the time resolution of the event detection algorithm. Processing tweets from a given time slot can be carried out by multiple processing units in parallel which can also speed up the feature calculation. Features are both stored in a historical database and used for calculating keyword-lifecycle (standardized) features. The calculated feature vectors are then forwarded to an SVM-based classifier to make the final decision. If an event is detected, its location is determined from the geotagged messages by a simple majority voting method. Note that in some cases there is not enough information to reliably estimate the event’s position.

To satisfy real-time criterion, the tweet processing must be faster than the arrival rate of the messages. Figure 2 shows the number of messages temporarily stored in the processing queue over time. This experiment was carried on a simple desktop machine with Intel Core i5 2.50 Ghz Processor and 8 GB RAM using only one processing unit. The average incoming rate of the input stream was approximately 2500 tweets/minutes. One can observe that the messages arrive in bursts, resulting transient increases in queue size. However, after each burst the queue becomes empty, indicating that the processing units can deal with this load even on a single desktop computer.

A. Data collection

Twitter provides different ways of access to collect tweets. In this subsection, we examine how the different data collection approaches can be used for real-time event detection, revealing their advantages and disadvantages.
First, Twitter provides access to a public stream (Twitter Streaming API) through a permanent HTTP connection that enables us to obtain an approximately 1% sample of the full Twitter data set in real-time.

Second, Twitter also provides a REST API which in contrast to Stream API does not require a permanent connection. This access follows a request-reply model, returning most recent tweets in chunks of given size. The key disadvantages of this solution that it is not so flexible than the public stream, connections are continuously established and closed, and the request rate is also limited.

Besides the above two approaches, an alternative data collection approach is to build a Twitter bot that selects users systematically and follows them. This method would enable us to collect the tweets of the followed users. In this way, a very targeted data collection can be carried out by selecting users according to how valuable their activities/messages are. Furthermore, by using the public Twitter stream only 10% of the tweets have geographical coordinates. However, if we collect such users who are located in known place (e.g. the city or the country can be determined based on the previously tweets), we can get a more stable data source having appropriate location information. Unfortunately, according to the latest Twitter policy bots are not allowed anymore, so our framework now uses the public stream.

However, by analyzing the public stream we found that some keywords related to real-world occurs very rarely, providing statistically insufficient data for event detection. For example, the word ‘earthquake’ is generally rarely used, but according to previous studies [5] at time of an earthquake event there is expected to have a sharp increase in the frequency of this word. However, in the original public stream there is no evidence of such increase. However, Twitter public stream allows us to specify keyword-based filters and in this way the number of relevant tweets can significantly be increased, providing statistically enough information for event detection. Our framework currently uses more than 30 different preselected keywords related to specific events like earthquakes, fires and other disasters.

Note that several research projects have built infrastructures to collect, store and analyze tweets from the public stream, but our finding shed light on that this stream is not sufficient for keyword based analysis like event detection because of biased sampling. Furthermore, besides keyword filtering, by restricting the searching area the performance of local event detection may be improved, since much more relevant messages can be obtained in this way.

B. Feature calculation

The processing units are responsible for assigning tweets to time slots and calculating features for them. The incoming tweets first are grouped into one minute long disjoint timeslots. We then calculate the following keyword-related features to characterize the input stream:

For timeslot \( t \) and keyword \( k \),

- Tweet frequency of keyword \( k \) (denoted by \( f_k(t) \)) is the number of tweets containing \( k \) in timeslot \( t \).
- Joint tweet frequency of keyword \( k_1 \) and \( k_2 \) is the number of tweets containing both \( k_1 \) and \( k_2 \) in timeslot \( t \).
- Average tweet length for keyword \( k \) is the average length of tweets containing \( k \) in timeslot \( t \).

These features are not directly used by our event detector framework, but standardized features are calculated from their actual values according to our keyword-lifecycle model described in Section IV. For example, standardized tweet frequency of keyword \( k \) also takes into account the daily behavior (mean and standard deviation) of \( k \).

C. Event detection

Each standardized features results time series with transient excursions denoting unusual behaviors that may be an evidence of unpredicted real events. The input of our event detector is vectors of standardized features for two consecutive time slots.

An SVM [8] classifier is used to assign event or non-event flags to the actual time slot. To calibrate and validate the classifier, a good reference data set is needed. Unfortunately, for most of the real events manual reference data collection is the only option. However, in Section V we show that in some cases e.g. for natural disasters like earthquakes there are other information sources that can be used for training and validation purposes as well.

If an event is detected, its location is estimated by applying a majority vote to the coordinates of the related geotagged tweets. Note that in some cases the event cannot be localized because of the lack of information.

IV. KEYWORD-LIFECYCLE

In contrast to existing solutions our framework assumes a periodic temporal behavior of different features. To handle this, besides the actual values of various features we also take into account their average temporal behavior. In this section, we briefly analyze the temporal properties of the keyword-related features listed in the previous section and introduce a keyword-lifecycle model to filter out false-positive excursions caused by e.g. the daily behavior of Twitter users.

Figure 3 illustrates the daily behavior of the keyword ‘alarm’ for two days. We note that similar trend can be seen for all other days. One can observe that the number of tweets containing the keyword ‘alarm’ shows periodic temporal behavior, following a specific trend related to how people use the given word. Since most of the existing methods are searching for local increases or decreases in the time series, these periodic changes may be recognized as a clear evidence of unpredicted events. We note that Figure 3 exhibits all the tweets with the given keyword from all over the world. However, if we narrow the observation area, the excursions become much sharper, making the event detection more difficult. Similar behavior can be seen for many keywords that have various meaning and can be used in different situations. As described in Section III-A, without keyword filtering the input stream does not contain statistically enough information for reliable event detection and what is more it carries too much irrelevant tweets.
For keyword ‘alarm’, the excursions are generated by a morning wave. This word is more frequently used in the morning hours in English-speaking countries than at other times of the day. By analyzing these tweets, we also found that the significant portion of these messages is simply related to ‘alarm clocks’. Similar phenomenon can be shown for numerous keywords, but we have to note, that in some cases e.g. for the keyword ‘earthquake’ the periodic changes are totally missing. This finding inspired us to take into account not only the actual frequencies of different keywords but their average daily behavior, called keyword-lifecycle as well.

To model the daily behavior of keyword $k$, we first split the day into 1440 one minute long timeslots. For a timeslot $t$, the mean ($\mu_k(t)$) and standard deviation ($\sigma_k(t)$) of a long observation period (e.g. a week or longer) are calculated. Figure 4 depicts the average daily behavior (the keyword-lifecycle) of the keyword ‘alarm’.

To take into account the keyword-lifecycle, the standardized values (also known as z-scores) are calculated for each keyword-based features. For keyword $k$ and timeslot $t$, the z-score can be defined as follows:

$$z_k(t) = \frac{f_k(t) - \mu_k(t)}{\sigma_k(t)},$$

where $f_k(t)$ represents the actual tweet frequency of keyword $k$. Although the mean and the standard deviation may vary for different keywords, using z-scores makes the actual observations comparable, and eliminate the daily periodic changes that do not carry valuable information on real unpredicted events. The z-score values for the keyword ‘alarm’ are depicted on Figure 5.

Figure 3. Absolute frequencies for the keyword 'alarm' over time for two different days. Each red dot represents the number of relevant tweets containing the keyword for an one minute time-window. The blue curve represents the daily average frequency ($\mu_{\text{alarm}}(t)$).

Figure 4. The keyword-lifecycle of the word 'alarm', expressed by the mean and the standard deviation of its twitter frequency.

Figure 5. Z-Score values for keyword 'alarm' for 24 hours. By using the mean and the standard deviation the irrelevant excursions related to the daily behavior can be filtered out.

Instead of the actual tweet frequencies for different keywords, the standardized values are used by our event detection framework as keyword-related features. Note that z-scores can be defined for all other basic features introduced in Section III.

V. CASE STUDY – EARTHQUAKE DETECTION

In this section we show some preliminary results on the validation of our event-detection framework and methodology. Collecting reliable reference data for validation purposes poses many challenging since there are no public databases providing reliable information on local events like traffic jams, accidents, etc. This is why most of the existing studies use volunteers to manually classify event and non-event related tweets. In this section, we first show an automatic way to generate proper reference data set for earthquake related events and then describe how our framework performs in this use case scenario.

We have chosen earthquake detection for our use case, since there are some publicly available data sources (e.g. USGS) providing detailed information on natural disasters like earthquakes that can be used as ground truth for training and validation in our study. In addition, it should be also noted that many earthquake indicator bots including USGS exist on
Twitter that report on earthquakes in almost real-time. These bots could indicate the location of epicenter, the magnitude and the time of an earthquake. Such type of messages typically looks like the following: "#Earthquake of M 5.0, 78km SSW of Unalaska, Alaska http://t.co/mCobEli9ED"

One can also observe that these specific messages can mislead our detector since a high number of such messages are published minutes after a real event by different bots. Since we want to avoid that our classifier learns the pattern of artificial tweets, we have to filter out all the tweets submitted by bots. During our performance analysis, tweets by USGS’s earthquake detector bot and the USGS database have been used as reference data. For validation purposes, two week long data has been collected. The first week has been used to train our framework, including the calculation of keyword-lifecycles and the calibration of the SVM classifier, while the second week has been applied for validation. Based on these tweets, our earthquake detection method can automatically be validated.

In the one week long test data set 1324 earthquakes were reported by USGS from which mere 36 had a magnitude at least 5. In 29 cases out of 36, our framework had successfully recognized the earthquakes 3 minutes before they were reported by USGS in average. We also have to note that there were no false positive hits during this experiment. For weaker earthquakes having a magnitude at least 5. In 29 cases out of 36, our framework had successfully recognized the earthquakes 3 minutes before they were reported by USGS in average. We also have to note that there were no false positive hits during this experiment. For weaker earthquakes having a magnitude at least 2, 102 earthquakes were identified out of the total 261. We also have to note that in the one-week test data set, 32% of the detected events did not have GPS coordinates, so it was impossible to locate them based on geotagging. We are planning to use further localization information for the event localization process like the location of friends, etc.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a social network-based event detector and outline its architecture. In contrast to prior works where the absolute or relative changes in the frequencies of some predefined keywords are taken into account, we introduce a lifecycle for each keyword to be observed, expressing their average daily behavior over time. We also show that some keywords exhibit periodic behavior that can be handled by our model. Based on the proposed lifecycle model novel temporal features are defined that are used by our real-time event detector. We also show some preliminary results on the validation of our framework. In the future, we are planning to test the correctness of our methodology on a larger dataset and we would like to extend the scope of events to be detected by local ones like accidents, traffic jams, etc.

ACKNOWLEDGMENT

The publication was financed by the project TÁMOP-4.2.2.C-11/1/KONV-2012-0001 supported by the European Union, co-financed by the European Social Fund. The authors also thank the partial support of the EITKIC 12-1-2012-0001 project, partially supported by the Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund, and the MAKOG Foundation. Tamás Matuszka also thanks the partial support of the EITKIC 12-1-2012-0001 project.

REFERENCES


6. Figure – An example for detected earthquakes.

In case of earthquake events, unfortunately it has been found that the only a small percentage of relevant tweets was geotagged (i.e. the tweet has location information). Although this is not enough for filtering errors, in most cases can be used for localization earthquake. The location of an event was selected from the coordinates of geotagged tweets by majority voting. The stronger decision is, the better the reliability of location estimation is. Figure 6 shows an example for earthquake detection on a daily measurement basis. However,