Getting There First: Real-Time Detection of Real-World Incidents on Twitter

Miloš Krstajić
University of Konstanz
Germany

Christian Rohrdantz
University of Konstanz
Germany

Michael Hund
University of Konstanz
Germany

Andreas Weiler
University of Konstanz
Germany

ABSTRACT

Social networking and micro-blogging services such as Twitter have become a valuable source of information on current events. Widespread use of Twitter on mobile devices and personal computers enables users to share short messages on any subject in real-time, thus making it suitable for early detection of unexpected events where fast response is critical. In this paper, we present an online method for detection of real-world events in Twitter data, such as natural disasters or man-made catastrophes, by analyzing Twitter data. Our method combines different textual and frequency components that represent or approximate interesting semantic aspects of an event. We use visualization as a validation vehicle, which allows us to understand which components are relevant and what impact the parameters have on the results of our event detection algorithm.

INTRODUCTION

Social media sites for short messages such as Twitter have become a powerful tool for real-time information sharing on a large scale. Twitter currently produces 340,000,000 tweets per day from more than 140,000,000 active users[1]. Many users post messages related to specific real-world events as they happen, or shortly after. This huge resource can potentially be a valuable body of information about these events, which can significantly differ by type, location, and scale. Unexpected events, such as natural disasters or catastrophes, are local events with global impact, where real-time event detection is crucial in preparing the appropriate response.

Current event detection methods in social media streams can be classified into three categories: clustering-based, model-based, and those based on signal processing, see [2] for an overview. A well-known approach that belongs to the third category has shown that Twitter can be used to alert the world about natural disasters faster than traditional media [6]. An in-depth analysis of how breaking news spread on Twitter is provided in [3].

The detection of events on Twitter in real-time is challenging for several reasons. First, the volume of the messages is huge and unpredictable and, second, the message content makes most of the state-of-the-art text mining techniques unsuitable. In 2009, a short-term study by an online marketing company stated that a large portion of the generated information can be considered as pointless babble (40% according to pearanalytics [4]), making it difficult to separate signal from noise. Twitter messages are short (140 characters at most) and can often contain typos, grammatical errors, and cryptic abbreviations.

In this paper, we present our method for real-time detection of real-world events. The method detects potential events by identifying keywords, so-called *event-term candidates*, whose frequency suddenly becomes significantly higher than expected. We then use all tweets containing the term candidate to compute the scores that aim to capture different characteristics of the event. In our approach, visualization is used as a validation tool, which provides a direct insight into the algorithm output. We evaluate our techniques in a case study using real Twitter data and discuss our findings and future work.

PROBLEM DESCRIPTION

In this section, we provide details about the Twitter data, describe the analysis problem, and outline our approach.

Data

The Twitter platform provides direct access to the public live stream of Twitter messages and offers an API[2] to application developers for receiving a large portion of the total number of daily produced tweets. By using the Streaming API of Twitter with the so-called "gardenhose" level, we are able to collect 10% of the public live stream. It covers more than 30 million tweets per day and more than one million tweets per hour. Each tweet object is streamed in the semi-structured JSON format containing 67 data fields. A tweet object includes the short message itself along with detailed metadata on the tweet (e.g. count of retweets) and on the user's profile (e.g. count of followers).

To cope with the massive data stream and to facilitate the access to the data we use an extended version of a native XML database [7] to store the designated incoming data in a standardized format and access the incoming items on-the-fly. With this solution we are able to keep up with the live data stream and can support real-time access to the incoming data objects.

The Problem

Allan et al. [1] define an event as "something that happens at some specific time and place along with all necessary preconditions and unavoidable consequences". While this definition does not include duration, we can still use it to describe all real-world events. In the context of events in Twitter, we define an *unexpected event* as a sequence of Twitter messages,

[1] https://business.twitter.com/basics/what-is-twitter/

[2] https://dev.twitter.com/start

Published at the 2nd IEEE Workshop on Interactive Visual Text Analytics "Task-Driven Analysis of Social Media" as part of the IEEE VisWeek 2012, October 15th, 2012, Seattle, Washington, USA.

whose content is similar and related to a keyword, whose volume is larger than usual. The term *interesting* refers to a combination of textual and statistical features, which will be described below in more detail.

LIVE DETECTION OF INCIDENTS IN TWITTER

We base our approach on the idea that people would use Twitter to quickly respond to an important event or incident. An event is then considered important if there is an unexpectedly large amount of messages in a short period of time that are related to the event. Here the challenge is twofold: first, we need to define what is "unexpectedly large", and, second, we have to identify messages that are related to an event. Since the messages are short, we assume that a word-level analysis of tweets might be suitable to detect events. We can expect that applying a term-frequency based approach on Twitter could deliver good results and performance since: a) it is easy to adapt it to incremental data sources, and b) sophisticated state-of-the-art text mining approaches do not work well with short texts.

We monitor the frequency of individual keywords (event-term candidates) and, for those keywords that have unexpected frequency values, we calculate additional scores that could help in describing and detecting an important real-world incident.

Algorithm

Our approach builds on previous work on feature-based sentiment analysis of product reviews [5]. The real-time version of this offline event detection algorithm is extended to work on the Twitter stream for emergency response tasks, taking into account the characteristics and the challenges of this text stream such as short messages and large data volume. The algorithm works directly on the real-time text stream and, in difference to other approaches, is able to process the documents as soon as they arrive, instead of processing and analyzing chunks of documents sequentially.

Step 1: Queueing and preprocessing of tweets. A queue is used to buffer the incoming document stream in order not to skip any document during extraordinary peaks in the overall data load. Afterwards, each document is preprocessed to filter out non-English tweets and extract keywords that may indicate an incident (*event-term candidates*). The preprocessing is optimized to cope with the real-time processing requirement and includes tokenization, lemmatization, and part-of-speech (POS) tagging. During our tests we found out that nouns, proper nouns, and hashtags are a good choice for event-term candidates as they usually refer to explicit entities or concepts. Due to grammatical inconsistencies, the accuracy of POS tagging of tweets is lower than POS tagging of general text data and, therefore, the preprocessing step produces some wrong event-term candidates.

2. Event-term candidate extraction. For each incoming tweet, the algorithm compares the newly extracted event-term candidate with the list of event-term candidates that were extracted in the past. If the candidate occurs for the first time, it will be stored in a special data table with two attributes: *last_seen* and *moving_avg*. The timestamp of the current tweet will be saved to the data field *last_seen*, because by now it is the last time the candidate has been observed. The data field *moving_avg* contains the average temporal distance between two occurrences of a candidate, which will be incrementally updated with each appearance of the candidate. When a candidate appears for the first time, the *moving_avg* will be set to the time that has passed since the stream started. The update step is then: $moving\_avg_t = moving\_avg_{t-1} \cdot (1-\delta) + dist\_to\_previous \cdot \delta$. The coefficient δ dampens the *moving_avg* in order to avoid abrupt changes caused by outliers. We are using the value $\delta = 0.01$.

3. Event-term episode identification, update and deletion. If the candidate has been stored earlier, we check if the relative temporal distance to the previous occurrence of the candidate is smaller than the threshold $\alpha \cdot moving\_avg$. If so, we create or update an episode for the candidate. An episode will contain all tweets containing the candidate for which the relative temporal distance is smaller than our threshold $\alpha \cdot moving\_avg$. Each episode has an $expiration\_date$, which is calculated by adding the *moving_avg* of the candidate to the timestamp of the last tweet added. Expired episodes are removed to keep the storage load low.

4. Score calculation. When the number of documents in one episode exceeds β, which is a fixed threshold, we calculate the overall score of the episode. If this score is higher than the the third threshold γ, we consider this episode to be important. Then we update the visualization and show the event to the user. The parameters α, β, and γ are decisive for the number of retrieved events and can be tuned by the user trading off recall and precision. We analyzed multiple different real-world events and experimentally came up with parameter settings, which retrieve most of the relevant episodes for a real-world event ($\alpha = 0.25$ and $\beta = 15$). Next, increasing the parameter γ we minimize the number of falsely identified events.

Scoring

Without considering the score of an episode, the algorithm detects a lot of event-term episodes that are actually false alarms. Very often, these false alarms are just conversations or new topics started by some user that, due to the high-connectivity of Twitter users, suddenly go viral. Therefore, it is necessary to use a score to distinguish between episodes indicating a real-world event and non-event episodes that are for example caused by frequent retweets.

Real-world incidents typically have different characteristic semantic aspects. We aim to capture these aspects for each event episode approximating them through different heuristic score components. Each component is calculated using all tweets in the active episode. The **sentiment score**, which is defined in equation 1 is used to identify whether the documents of an episode are negative or not. Then, each document of an episode is compared with a *trigger-word list* to calculate the **trigger-word score** in order to distinguish between episodes that indicate a real-world event and those that represent a popular conversation topic. A trigger-word list, for example, can be generated by taking all hyponyms of the word *event* from WordNet[3]. The **relative time distances** between two documents (in relation to the *moving_average*) indicates the strength of the accelerated occurrence of a term.

[3] http://wordnet.princeton.edu/

The **absolute time distance** between two documents specifies whether the term in general occurs frequently or not. A word that is very frequent usually does not indicate an event. The **similarity between documents** (equation 2) can be used to check whether tweets deal with the same topic or not. Furthermore, we calculate the average number of tweets that are **retweets**, which implicitly indicates the interestingness of a tweet. The average number of tweets containing an **URL** will be calculated since tweets dealing with news often contain an URL. We also calculate scores based on whether the event-term candidate is a **hashtag** or not, the **source of the tweet** (mobile phone, computer, button on homepage,...) and the importance of the **user** who published the tweet, which is defined as the number of the user's followers.
Most of our scores are ratio-based and we weight each score by the total **number of documents** in the episode. The usefulness of each individual score, as well as the design of the overall score depends on the type of events the user wants to find. For example, if we are looking for natural disasters such as floods or earthquakes, we might want to put more weight on the sentiment than on any other score.

$$sentiment\_score = \begin{cases} 0, & \text{if } avg\_senti > 0, \\ |avg\_senti|, & \text{else} \end{cases} \quad (1)$$

wherein avg_senti refers to the average sentiment score of all documents of the current episode.

$$document\_similarity = \frac{1}{\#ofdocs} \cdot \frac{|frequent_1| + |frequent_2|}{2} \quad (2)$$

wherein $|frequent_1|$ and $|frequent_2|$ refer to the number of documents containing the most and second most frequent event-term candidate, ignoring the candidate of the current episode.

VISUALIZATION AS A VALIDATION TOOL

Visualization for detection algorithms can have two different purposes. First, it can give algorithm developers insightful feedback about how novel methods work on real data and support systematic testing and validating capabilities. For example, they allow developers to learn more about the relevance of different components and the impact of different parameter settings. This is especially valuable in the case of text streams, because there are no suitable ground-truth data sets available for automated evaluation. Second, visualization can also be used to convey relevant information and grant interactive explorative access to the text stream to end users such as professional analysts. In this paper, we treat mainly the usage of the first scenario, as it provides us the possibility to evaluate our different scores and parameters in order to find an optimal configuration for real-world event detection. Besides, this research direction has not received a lot of attention until now.
Each of the frequent terms extracted by our algorithm occupies a row in our visualization, as shown in Figure 1. Time is mapped to the x-axis, while the term occurrences are represented by vertical bars, whose widths indicate the number of underlying documents. Each bar represents a three minute interval and is divided into several small rectangles, each representing an individual score. We use a colormap, going from blue (low score) over white to red (high score), to represent the value of the score. The scores have the following order: *Sentiment, Negative Event Words from WordNet, All Event Words, Triggerwords from News, Relative Time Distance, Absolute Time Distance, Retweets, URL, Candidate Hash Tag, Document Similarity*. We order the frequent terms according to their overall score (Figure 1), which is calculated according to the user-defined weighting of the individual scores and represented by the color of the large square at the left end of each row.
It is important to consider that event-term episodes collect documents of an event-term candidate until the score of the episode exceeds the threshold γ. Therefore, it can happen that a document occurs in a bar which is visualized at a later point in time (compared to the document's timestamp). This is inevitable, since an episode requires several documents to exceed the threshold γ, but the moving average of some event-term candidates can be larger than the three minute interval.

OUR FIRST RESULTS

One of the events that our algorithm detected on the live Twitter stream was the "Aurora shooting"[4], which took place on July 20, 2012. An armed man fired indiscriminately in a theatre in Aurora (near Denver, Colorado, USA) during the premiere of the new Batman movie "The Dark Knight Rises". The shooting started shortly after the beginning of the movie, at 00:38 local time (06:38 UTC), killing 12 people and injuring 58 more. Figure 1 shows the output of our tool for the Aurora shooting. The algorithm detected the first frequent term *aurora* at 07:57 UTC, about 1 hour and 20 minutes after the incident had started. Other important terms, such as *victim*, *gunman*, and *wound*, are detected about 20 minutes later, after 8:20 UTC. These terms might indicate a new information as it appears (that the incident involved a gunman, and the first reports about the wounded people). It is interesting to note that terms such as *kill* and *shoot* were not detected as unexpected events. Looking into the data, it can be seen that these terms are very often used as jargon among Twitter users and this incident did not cause a noticeable increase in their frequency. Another term, *theatreshooting*, appears later in the visualization (after 8:45 UTC). It is a hashtag and this example actually shows how the hashtags are created much later than our detected event.

Discussion

At first sight, the time gap between the incident and the detection seems quite long for a real-time event detection algorithm. We therefore looked directly into the tweets between the beginning of the event and the time of our detection. The first two tweets dealing with the Aurora shooting occur at 07:13 UTC and 07:18 UTC. There are no other tweets about the shooting until 07:45 UTC, when they start appearing at a rate of about 3 tweets per minute and then continuously increasing. Why is the event not represented in Twitter, right after the incident? We can speculate that this might be caused by the local time (most of the people in the area are asleep and can not tweet about the event) and also by our data stream

[4] http://en.wikipedia.org/wiki/2012_Aurora_shooting

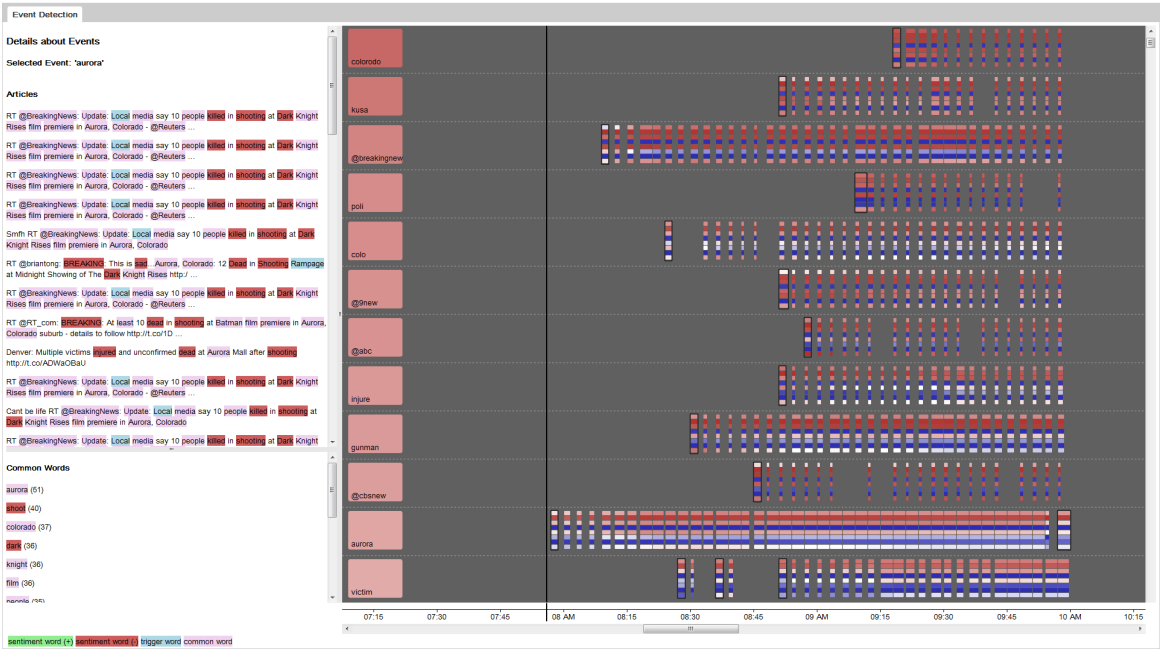

Figure 1. Screenshot of our validation tool for the Aurora shooting. On the right side, a row for each frequent term is shown containing vertical bars which indicate the occurrence of the term. Each bar represents the incoming documents (containing the term) in three minute intervals and is divided into multiple small rectangles, which represent different scores. The colormap goes from blue (low score) over white to red (high score). The width of the bar is mapped to the number of documents. The user can click on a bar and see all underlying tweets and their common words on the left side.

source itself, since we have access to random 10% of the whole Twitter stream. Therefore, it might be possible that the whole stream contains more tweets about the event.

Nine out of ten frequent words with the highest overall score, retrieved by the algorithm, are explicitly or implicitly related to the Aurora shooting. For this result, we use the following scores for the ranking: *sentiment*, *trigger words*, *retweets* and *urls*. Additional preliminary experiments showed that very similar results, with a slightly higher number of false positives, could be achieved by excluding the trigger-words or sentiment score. Summary of the most important findings:

1. Our word-level event detection approach works in real-time. The implementation can easily process the 10% live stream without batch processing on a laptop with Intel Core i7 CPU with 8 GB of RAM and would also scale up for the whole Twitter stream.
2. Generally infrequent terms (like *Aurora*) are more suitable to point to events than frequent terms (like *shoot*).
3. Hashtags evolve after the event was already detected.
4. The trigger-words score supports and improves the detection process, but the results are also acceptable without them.
5. The relative time distance score is always high and therefore it does not seem to be useful.
6. The URL score, similar to the hashtags, filters out some noise, but generates many false positives.

CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the initial results of our real-time visual analytics approach for automatic identification of unexpected real world incidents in Twitter. Our algorithm takes into account different detection components, which represent or approximate semantic aspects of the events. Visualization is used as part of the validation process, where we can see how different components and parameter settings influence the output of the algorithm. Currently, keywords relating to the same event are treated independently and can be understood as part of the same event by looking at the similar time intervals and scores. As part of our future work, we will work on developing a visualization that can be used to allow explorative analysis to end users.

REFERENCES

1. Allan, J., Ed. *Topic detection and tracking: event-based information organization*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.
2. Bontcheva, K., and Rout, D. Making Sense of Social Media Streams through Semantics: a Survey. *Semantic Web* (2012).
3. Hu, M., Liu, S., Wei, F., Wu, Y., Stasko, J., and Ma, K.-L. Breaking news on twitter. In *Proc. CHI 2012*, ACM (2012), 2751–2754.
4. Pearanalytics. Twitter study. `http://www.pearanalytics.com/`, 2009.
5. Rohrdantz, C., Hao, M. C., Dayal, U., Haug, L.-E., and Keim, D. A. Feature-based visual sentiment analysis of text document streams. *ACM TIST 3*, 2 (2012), 26.
6. Sakaki, T., Okazaki, M., and Matsuo, Y. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proc. WWW 2010*, ACM (2010), 851–860.
7. Weiler, A., Mansmann, S., and Scholl, M. H. Towards an advanced system for real-time event detection in high-volume data streams. In *Proc of 5th workshop for Ph.D. students PIKM 2012*, ACM (2012).