Geo-spatial Event Detection in the Twitter Stream

Maximilian Walther and Michael Kaisser

AGT International,
Jägerstraße 41, 10117 Berlin, Germany
{mwalther,mkaisser}@agtinternational.com

Abstract. The rise of Social Media services in the last years has created huge streams of information that can be very valuable in a variety of scenarios. What precisely these scenarios are and how the data streams can efficiently be analyzed for each scenario is still largely unclear at this point in time and has therefore created significant interest in industry and academia. In this paper, we describe a novel algorithm for geo-spatial event detection on Social Media streams. We monitor all posts on Twitter issued in a given geographic region and identify places that show a high amount of activity. In a second processing step, we analyze the resulting spatio-temporal clusters of posts with a Machine Learning component in order to detect whether they constitute real-world events or not. We show that this can be done with high precision and recall. The detected events are finally displayed to a user on a map, at the location where they happen and while they happen.

Keywords: Social Media Analytics, Event Detection, Twitter.

1 Introduction

The rise of Social Media platforms in recent years brought up huge information streams which require new approaches to analyze the respective data. At the time of writing, on Twitter[1] alone, more than 500 million posts are issued every day. A large part of these originate from private users who describe how they currently feel, what they are doing, or what is happening around them. We are only starting to understand how to leverage the potential of these real-time information streams.

In this paper, we describe a new scenario and a novel approach to tackle it: detecting real-world events in real-time in a monitored geographic area. The events we discover are often on a rather small-scale and localized, that is, they happen at a specific place in a given time period. This also represents an important distinction to other work in the field (see Section 2) where event detection is often the same as trend or trending topic detection. In this paper, we are not interested in discussions about the US elections, celebrity gossip, spreading memes, or the fact that an earthquake happened in a distant country. We are interested

[1] http://twitter.com/

in, e.g., house fires, on-going baseball games, bomb threats, parties, traffic jams, Broadway premiers, conferences, gatherings and demonstrations in the area we monitor. Furthermore, independent from the event type, we want to be able to pinpoint it on a map, so that the information becomes more actionable. So, if there is an earthquake in the area we monitor, we want to know where it caused what kind of casualties or damages. We believe that such a system can be useful in very different scenarios. In particular, we see the following customer groups and use cases:

Police forces, fire departments and governmental organizations to increase their situational awareness picture about the area they are responsible for.

Journalists and news agencies to instantly be informed about breaking events.

Private customers that have an interest in what is going on in their area. Here, the particular nature of Twitter and its adoption by a younger, "trendy" crowd suggests applications along the lines of, e.g., a real-time *New York City party finder*, to name just one possibility.

2 Related Work

Current approaches on event detection in Social Media streams center around two focal points: event augmentation and trending topic detection. In the first case, the system receives input about an event from external sources and finds information on Social Media sites suitable to augment this input. In the second case, the event to be detected is on a large, often global scale, and receives wide-spread coverage on Social Media sites. In such cases, "event" is often used interchangeably with "topic", "trend" or "trending topic".

In the area we have just categorized as event augmentation, [11] present an approach that gathers tweets for target events that can be defined by a user via keywords. The authors apply classification and particle filtering methods for detecting events, e.g., earthquakes in Japan.

Twitcident [1,2] enables filtering, searching, and analyzing Twitter information streams during incidents. It listens to a broadcast network which provides information about incidents. Whenever a new message comes in, it searches for related tweets which are semantically extended in order to allow for effective filtering. Users may also make use of a faceted search interface to dive deeper into these tweets.

The event detection system going by the name of TEDAS [6] employs an adapted information retrieval architecture consisting of an online processing and an offline processing part. The offline processing is based on a fetcher accessing Twitter's API and a classifier to mark tweets as event-related or not event-related. The focus of TEDAS is on so-called CDE events (crime- and disaster-related events). For classifying tweets as CDE events, content features (e.g., inclusion of lexicon words), user features (e.g., number of followers), and usage features (e.g., number of retweets) are taken into account.

In the area we classified as trending topic detection, [9] present an approach dealing with streaming first story detection. The presented system decides for every incoming tweet if it belongs to an already existing story. This is done with the help of so-called locality-sensitive hashing (LSH). The computed hash is compared with available stories. If the difference is below a predefined threshold, the tweet is added to the story. Otherwise, it is marked to be a new story. Since not all clusters created this way are actual stories, a follow-up component measures how fast the different stories grow. Only the fastest growing ones are collected since they are assumed to be the stories that attract the most public attention. In a follow-up publication [10] introduce the extension of using paraphrases to improve the first story detection.

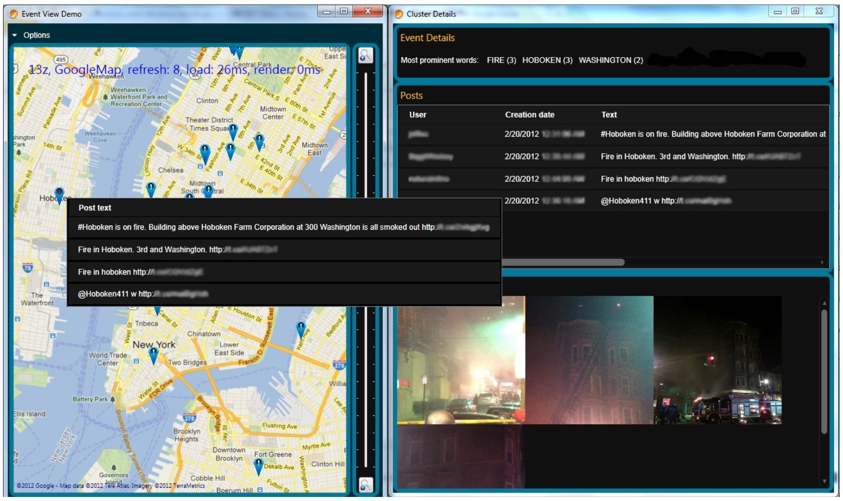
[3] are concerned with real-time trending topic detection in order to retrieve the most emergent topics currently discussed by the Twitter community. A term life cycle model is used to detect terms that are currently more frequently used than they were in the past. The importance of a source is assessed via a version of the Page Rank algorithm. As a last step, a keyword-based topic graph connecting emerging terms with co-occurrent terms is displayed to the user.

[7] describe "TwitterMonitor", a system which performs trend detection on the Twitter stream. In order to achieve this, the system looks for keywords that show up in the stream at an unusually high rate at a given point in time. Trending keywords are grouped into disjoint subsets with a greedy algorithm, each indicating a topic of a current discussion.

In contrast to the above mentioned approaches, we focus on a novel scenario concerned with detecting geo-spatial, real-world events, many of which are of a fairly small scale, e.g., house fires or parties, and thus are often covered by only few tweets. We are not interested in global discussions, trending memes and the like. In fact, we need to make sure that such tweets (and there are a lot of them) are disregarded by our system. We also do not rely on any user or external input (the only input to the system are Social Media streams, in this paper the Twitter stream exclusively), and our goal is to not only detect such real-world events, but also to know precisely where they happen, so that they can be presented to a user on a map while the event happens or shortly after.

3 System Overview

We aim to detect real-world events, often of rather small scale, in a given monitored geographic area and conduct the experiments described in this paper with tweets from the New York metropolitan area. We receive more than three million tweets in any 24 hour period which from a processing-time perspective significantly narrows down the set of potentially applicable real-time algorithms. An interesting approach to solve the problem at hand would, for example, be to compute textual similarity of the tweets with a vector space model where location and time information could, in one way or another, be included as additional dimensions of the vector space. It is clear, however, that with the large amount of posts at hand, it is unfeasible to compute the distance between all tweets,



Post text

#Hoboken is on fire. Building above Hoboken Farm Corporation at 300 Washington is all smoked out http:[redacted]

Fire in Hoboken. 3rd and Washington. http:[redacted]

Fire in hoboken http:[redacted]

@Hoboken411 w http:[redacted]

Event Details

Most prominent words: FIRE (3) HOBOKEN (3) WASHINGTON (2) [redacted]

Posts

User	Creation date	Text
[redacted]	2/20/2012 [redacted]	#Hoboken is on fire. Building above Hoboken Farm Corporation at
[redacted]	2/20/2012 [redacted]	Fire in Hoboken. 3rd and Washington. http:[redacted]
[redacted]	2/20/2012 [redacted]	Fire in hoboken http:[redacted]
	2/20/2012 [redacted]	@Hoboken411 w http:[redacted]

Fig. 1. An event detected by the algorithm described in this paper, displayed by the system's GUI prototype. To the left, we see a map of New York where each detected event is displayed with a marker. A user can hover over the events to get a preview. If a marker is clicked, more details are displayed on the right. The tweets themselves are displayed on top while any potential pictures are displayed below.

so some kind of pre-selection needs to take place. While this could in theory be done with an inverted index, for example like it is implemented in Lucene [5], we opted for a different approach that first pre-selects a set of tweets based on their geographical and temporal proximity and then uses a Machine Learning algorithm to further evaluate these candidate clusters. We adopt this approach primarily for two reasons:

1. Geo-spatial and temporal indices are supported by many databases today, so this processing can be performed efficiently with off-the-shelf tools.
2. A Machine Learning approach gives us the flexibility to evaluate many facets of the cluster candidates by defining a wide range of features that analyze the textual content of the tweets as well as additional features that deal with other aspects of the candidate clusters.

We have chosen MongoDB[2] as the database system because it supports geo-spatial and temporal indices, has fast read and write times, and its document model is very suitable to store Social Media posts. Furthermore, MongoDB can easily be scaled horizontally, should the need arise. We keep all tweets issued in the last 24 hours in a table and constantly query it for *tweet clusters*, that is, tweets that were issued geographically close to each other in a given time frame. The architecture that supports this is described in more detail in Section 4.

[2] `http://www.mongodb.org/`

Each identified *tweet cluster* is treated as an *event candidate.*[3] It might potentially indicate a real-world event currently happening at the location where the tweets are issued. Alternatively, we might have picked up a bunch of unrelated tweets, coincidentally issued geographically and temporally close to each other. We implemented 41 features that address various aspects of the event candidates. These are used in a Machine Learning scheme to make a binary decision as to whether a tweets cluster constitutes a real-world event or not. This processing is done in real-time, and clusters classified as events are shown to a user in a GUI where they can interactively explore the events on a map, see Figure 1.

4 System Architecture

Requirements for Social Media Analytics architectures generally include support for large incoming data streams that have to be processed in real-time. In our case, because we focus on a specific geographic region, this challenge is somewhat reduced. Nevertheless, we deal with a significant number of tweets that have to be stored and processed. Additionally, we were from the outset considering to scale the scenario to multiple and/or larger regions. Our architecture supports real-time processing of tweets and centers around a MongoDB database in which we keep the tweets, intermediate results (event candidates), and final results (events), see Table 1. Several independent modules query and update this database. Each module is designed in a way so that multiple instances can be run in parallel, should it be necessary. The modules we use are the following:

TweetFetcher. The most recent tweets received from Twitter's public APIs are put in a queue and then written to a MongoDB table named *Posts.*

ClusterCreator. For each new incoming tweet, this module checks whether there were more than x other tweets issued in the last y minutes in a radius of z meters. For the experiments in this paper, we used the settings $x = 3, y = 30, z = 200$. Whenever a new cluster can be created this way, it is written to the table *EventCandidates.*

ClusterUpdater. This module updates existing clusters, i.e., if a new tweet is created in the area of an already existing cluster, it is added to that cluster. This module also merges clusters that overlap temporally or spatially if this overlap is above a certain threshold and if the resulting cluster does not exceed n tweets where currently $n = 50$. Clusters in which the newest tweet is older than 48 hours are deleted.

ClusterScorer. Each new or updated cluster candidate in *EventCandiates* is scored by this component. Here we use the Machine Learning setup described in more detail in Section 5.

[3] The terms *tweet cluster* and *event candidate* represent different perspectives of looking at the same data structure. In the following we use both terms interchangeably.

Table 1. MongoDB tables used to store tweets and event clusters

Table name	Description	Life Time
Posts	Contains all tweets received by Twitter API	24 hours
EventCandidates	Clusters of tweets issued close to each other in terms of time and location	48 hours
ScoredEvents	Evaluated tweet clusters	7 days

5 Evaluating Event Candidates

This section describes the *ClusterScorer* component of the system. Each newly created or updated cluster is evaluated individually and a binary decision is made as to whether this cluster constitutes an event or not. An overview of the 41 features we extract from the tweet clusters can be seen in Table 2, broken down into *textual* features, concerned with analyzing the textual content of the tweets, and *other* features, encompassing all other aspects.

Table 2. Overview of features used by the system. The first column lists the name of the feature group, the second lists the number of features in that group, and the third column gives a brief description. See text for more details.

Textual features

Feature Group	#	Brief Description
Common Theme	1	Calculates word overlap between different tweets in the cluster.
Near Duplicates	1	Indicates how many tweets in the cluster are near-duplicates of other tweets in the cluster.
Positive Sentiment	3	Indicates positive sentiment in the cluster.
Negative Sentiment	3	Indicates negative sentiment in the cluster.
Overall Sentiment	2	Indicates the overall sentiment tendency of the cluster.
Sentiment Strength	3	Indicates the sentiment strength of the cluster.
Subjectivity	2	Indicates whether tweeters make subjective reports rather than just sharing information, e.g., links to newspaper articles.
Present Tense	2	Indicates whether tweeters talk about the here & now rather than making general statements.
# Ratio	1	Number of hashtags relative to the number of posts in the cluster.
@ Ratio	1	Number of @s relative to the number of posts in the cluster.
RT Ratio	1	Fraction of tweets in the cluster that are retweets.
Semantic Category	13	Indicates whether the cluster belongs to certain event categories, e.g., "sport event" or "fire".

Other features

Feature Group	#	Brief Description
Link ratio	1	Indicates the number of posts that contain links.
Foursquare ratio	1	Fraction of tweets originating from Foursquare.
Tweet count	1	Score based on how many tweets are included in the cluster.
Poster count	2	Score based on how many different users posted the tweets in the cluster.
Unique coordinates	2	Score based on how many unique locations the posts are from.
Special location	1	Fraction of tweets that are from certain known "bad" locations, e.g., airports or train stations.

5.1 Textual Features

These features are concerned with analyzing the textual content of the tweets. In the following, we describe a few important features in more detail:

Common Theme. For each event candidate, we compute a list of the most frequent words it contains by using binary term counts on tweet level (but not on cluster level). As a result we have a list of words $w_1 \dots w_n$ and for each word its frequency $f(w)$. The formula to compute the *commonTheme* feature is:

$$commonTheme(w_1 \dots w_n) = \frac{1}{cm} \sum_{i=1}^{n} \begin{cases} 0 & \text{if} f(w_i) = 1 \\ f(n) & \text{otherwise} \end{cases} \quad (1)$$

where m is the number of tweets in the cluster, and c is a constant set to 3 for the experiments in this paper.

Near-Duplicates. We compute the length of the longest common substring (lcs) between all pairs of tweets $t_1 \dots t_n$ in a cluster, divide each value by the length of the shorter tweet, and compute the mean of all quotients:

$$D(t_1 \dots t_n) = \overline{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{len(lcs(t_i, t_j))}{\min(len(t_i), len(t_j))}} \quad (2)$$

Sentiment Features. This is a dictionary-based feature that uses a selection of sentiment dictionaries to detect a cluster's sentiment. We use the data provided in [8] and [12] and a few dictionaries that were manually created by us and contain, for example, frequent emoticons. We use separate scores indicating positive or negative sentiments from the different dictionaries. We also combine the positive and negative scores into an overall sentiment score.

Subjectivity. This is another dictionary-based feature where the dictionary contains first-person personal pronouns. It is designed as an indicator as to whether tweeters talk about personal experiences or about general issues.

Present Tense. This dictionary-based feature contains common present tense verbs and auxiliaries. It is designed as an indicator as to whether tweeters talk about the here and now or about past events.

Semantic Category. While we deliberately do not want to limit our algorithm to detect events from known categories only, our experiments have shown that there are certain categories which are frequently found. In order to assist the algorithm in detecting these categories, we use 13 small dictionaries that contain n-grams which are indicative for these categories. A *sport events* category would for example contain terms like "match", "vs", and also names of sport teams from the New York region. Other categories we use are, e.g., "entertainment event", "music event", "traffic jam", and "violence".

All of the dictionary-based features listed above follow the same straightforward scoring scheme to compute their feature value. Each dictionary entry contains an associated weight v (where $0.0 < v \leq 1.0$). In most cases, this weight is learned, in other cases it is manually assigned. The score for each word w in a tweet t is then computed as follows:

$$wordScore(w) = \begin{cases} v(w) & \text{if word w is in dictionary} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Based on all word scores in a tweet, we compute the tweet score:

$$tweetScore(w_1 \dots w_n) = \min\left(\frac{1}{c}\sum_{i=1}^{n} wordScore(w_i), 1\right) \tag{4}$$

where c is a constant set to 3 for the experiments in this paper. All tweets in a cluster are combined into a cluster score which is the final feature value:

$$clusterScore(t_1 \dots t_n) = \left(\frac{1}{n}\sum_{i=1}^{n} tweetScore(t_i)\right)^{\frac{1}{2}} \tag{5}$$

5.2 Other Features

Besides the textual features, we also define a number of additional features which also cover important aspects of the tweet cluster. Some examples are listed in the following.

Tweet Count. A score indicating the number of tweets in a cluster.

Poster Count. The two poster count features are concerned with the number of posters in the cluster (which is often different from the number of tweets). In particular, we often see strings of tweets from the same person issued at the same location. Such monologues typically do not describe a real-world event. One of the two features we compute is relative to the number of tweets in the tweet cluster, the other one is absolute.

Unique Coordinates. These features evaluate how many tweets from unique coordinates the tweet cluster contains. This is important because a cluster containing many tweets with exactly the same coordinates might indicate that they originate from bots rather than from humans. If several persons independently witness an event and decide to tweet about it, we would expect the coordinates to be similar but not exactly the same.

Special Locations. People frequently tweet while they are bored and/or waiting. For this reason, we get a high number of tweets from locations like train stations and airports. For this feature, we defined a list of locations in NYC which are problematic in this regard. This feature's value is the fraction of tweets in a cluster being posted from such locations.

5.3 Ranking Posts

A frequent observation that we have made while building the system is that a cluster constituting a real-world event often still contains a few “stray” tweets that are not related to the event. One can hope that the Machine Learning setup is robust enough to deal with this (after all, the training data contains such examples), but even then the problem persists which tweets should be displayed to the user, and in which order. As common in IR, we present the user with a ranked list of documents, in our case tweets. This gives us the opportunity to move stray tweets to the bottom of the list–possibly below the cut-off point of what is displayed–and to put tweets that are representative for the whole cluster at the top. Based on these assumptions, here is how we rank tweets:

$$tweetRankingScore(w_1 \ldots w_n) = \sum_{i=1}^{n} wordCount(w_i) \tag{6}$$

where $wordCount(w_i)$ is a score indicating how often this word in the tweet occurs in the whole cluster (again using binary tweet counts on tweet level, which are summed up to a numeric score on cluster level). All tweets in the cluster are then sorted by their $TweetRankingScore$ and displayed in descending order. Figure 1 shows an example. The displayed tweets were ranked using the described approach. We can see that a tweet which provides a good summary of the event is displayed on top, while the text of the last tweet contains very little information.

6 Experiments and Evaluation

6.1 Experimental Setup

For our evaluation we use a data set containing 1,000 event candidates sampled from all event candidates (tweet clusters) the system created during two separate one-week periods (February 10-16, 2012 and April 17-23, 2012; Section 4 explains how these tweet clusters are created in more detail). The clusters were manually labeled in a binary fashion as to whether they constitute a real-world event or not. 319 clusters were labeled as positives (they do describe a real-world event) while 681 were labeled as negatives (they do not describe a real-world event). For all experiments reported below, we use a 10-fold cross-validation approach. The question we seek to answer is: Can we reliably detect whether a tweet cluster describes an event or not?

6.2 Overall Performance

Table 3 gives the results from our evaluation for three different Machine Learning algorithms. We used the implementations provided by Weka [4]. As can be seen, with a pruned C.45 decision tree (called "J48" in Weka), a precision of 0.858 and a recall of 0.856 is achieved. Its kappa statistic is 0.79. All classifiers outperform a ZeroR baseline that classifies every instance as an event. The differences between the baseline and the classifiers are statistically significant according to a sign test ($p < 0.05$).

Table 3. Precision, recall, and F_1-score for different Machine Learning algorithms when binarily classifying tweet clusters as events. A ZeroR baseline is also given.

Algorithm	**Precision**	**Recall**	**F_1-score**
Naive Bayes	0.783	0.871	0.825
Multilayer Perceptron	0.829	0.837	0.833
Pruned C4.5 decision tree	0.858	0.856	0.857
ZeroR	0.319	1.000	0.484

6.3 Textual vs. Other Features

Table 4 presents the evaluation results with the difference that a) only textual features and b) only other features have been employed (see Table 2). As we can see, the features which analyze the textual content of the tweets deliver a slightly higher precision, recall, and F_1-score than the other features. However, according to a sign test ($p < 0.05$), these differences are not statistically significant.

Table 4. Precision, recall, and F_1-score when when using a pruned C4.5 decision tree and only using once all textual features and once all other features, see Table 2

Features used	**Precision**	**Recall**	**F_1-score**
Textual features	0.827	0.765	0.795
Other features	0.799	0.749	0.773
All features	0.858	0.856	0.857

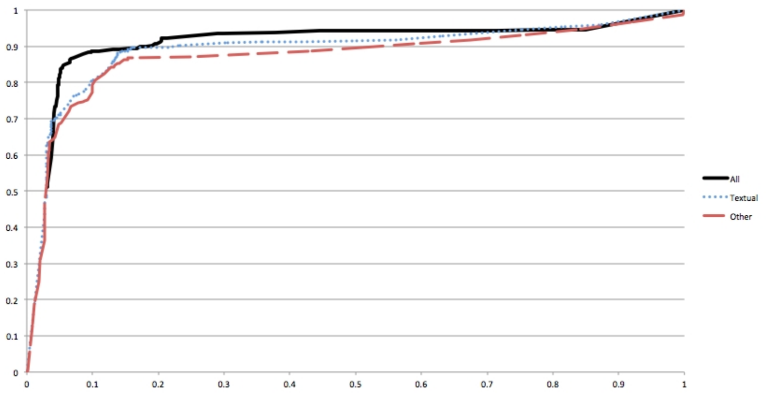


Fig. 2. ROC curves for J48 classifier with all features, only textual features, and only other features. Areas under the curve are 0.908, 0.893, and 0.873, respectively.

6.4 Most Effective Features

We furthermore evaluated which of our features have the biggest impact on the J48 classifier's performance. Hence, we repeated the evaluation 41 times, each time with a different feature. The 10 best-performing features are listed on the left side of Table 5. We also performed an exhaustive search to determine

Table 5. Precision, recall, and F_1-score when using the 10 best-performing features individually (left side), and for groups of two, three, and four features that work best in combination (right side)

Features (individual)	Prec.	Recall	F_1-score
Unique Posters (Total)	0.901	0.655	0.759
Common Theme	0.604	0.856	0.708
@ Ratio	0.546	0.931	0.688
Unique Coordinates (Total)	0.670	0.668	0.669
Ratio of Foursquare Posts	0.708	0.624	0.663
No. Tweets in Cluster	0.594	0.724	0.653
SemCat Sport Event	0.941	0.398	0.599
Subjectivity	0.665	0.524	0.586
Positive Sentiment	0.592	0.533	0.561
Unique Posters (Ratio)	0.626	0.483	0.545
All features	0.858	0.856	0.857

Features (combined)	Prec.	Recall	F_1-score
Unique Posters (Total) & Common Theme	0.810	0.853	0.831
Unique Posters (Total) & Common Theme & Subjectivity	0.828	0.846	0.837
Unique Posters (Total) & Common Theme & Subjectivity & SemCat Sport Event	0.832	0.850	0.842

the groups of two, three, and four features which work best in combination. Corresponding results can be seen on the right side of Table 5.

Especially the two best performing features can intuitively be explained well: A cluster is good when it contains many tweets from different posters. In other words: The more people tweet from one place, the less likely it is that this is by pure chance, thus, we can be more confident that something interesting is going on there.[4] Common Theme is the second-best indicator: If people tweeting from the same place use the same words, it is likely that they talk about the same thing which probably is some noteworthy event. These two features also work well together and result in an F_1-score of 0.831.

7 Conclusion and Future Work

We have presented a novel approach to detect geo-spatial, real-world events in real-time by analyzing the Twitter stream. We have shown that we can reliably detect whether clusters of tweets issued temporally and spatially close to each other describe a real-world event or not. We furthermore have presented an analysis that looks into the features used and their impact on system performance. An open question we have not yet evaluated is the recall of the system with regards to actual real-world events, not candidate clusters. The central question here is: How many of the events happening in a specific geographic area (e.g., NYC) are a) reported in the Twitter stream and b) can be detected with our system?

As far as future work is concerned, we are planning to classify the detected events according to their category. We also plan to include TF-IDF weights when computing cluster overlap in the future. A cluster that overlaps in the terms *panic*, *fire*, and *help* for example should receive a significantly higher score than a cluster whose most prominent words are *think*, *also*, and *back*. When manually inspecting the classifier's results, we have seen quite a few false positives where the clusters overlap in words that are not descriptive for their content.

[4] Note that number of tweets is not such a good indicator. This is because this feature can be deceived by a series of tweets from one user in a short time frame. Such monologues typically do not constitute an event.

References

1. Abel, F., Hauff, C., Houben, G.-J., Stronkman, R., Tao, K.: Semantics + Filtering + Search = Twitcident. Exploring Information in Social Web Streams. In: Proceedings of the 23rd ACM Conference on Hypertext and Social Media, HT 2012 (2012)
2. Abel, F., Hauff, C., Houben, G.-J., Stronkman, R., Tao, K.: Twitcident: Fighting Fire with Information from Social Web Stream. In: Proceedings of the 23rd ACM Conference on Hypertext and Social Media, HT 2012 (2012)
3. Cataldi, M., Di Caro, L., Schifanella, C.: Emerging Topic Detection on Twitter Based on Temporal and Social Terms Evaluation. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining, MDMKDD 2010 (2010)
4. Hall, M., Frank, E., Holmes, G., Pfahringer, P., Reutemann, B., Witten, I.H.: The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
5. Hatcher, E., Gospodnetic, O., McCandless, M.: Lucene in Action. Manning. 2nd revised edn. (2010)
6. Li, R., Lei, K.H., Khadiwala, R., Chang, K.C.-C.: TEDAS: A Twitter-based Event Detection and Analysis System. In: Proceedings of the IEEE 28th International Conference on Data Engineering (2012)
7. Mathioudakis, M., Koudas, N.: TwitterMonitor: Trend Detection over the Twitter Stream. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, SIGMOD 2010 (2010)
8. Nielsen, F.A.: A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In: Proceedings of the ESWC2011 Workshop on Making Sense of Microposts (2011)
9. Petrovic, S., Osborne, M., Lavrenko, V.: Streaming first story detection with application to Twitter. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2010 (2010)
10. Petrovic, S., Osborne, M., Lavrenko, V.: Using paraphrases for improving first story detection in news and Twitter. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2012 (2012)
11. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake Shakes Twitter Users: Real-Time Event Detection by SocialSensors. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010 (2010)
12. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT 2005 (2005)