Online clustering and classification for real-time event detection in Twitter

Federico Angaramo
Istituto Superiore Mario Boella, Italy
angaramo@ismb.it

Claudio Rossi
Istituto Superiore Mario Boella, Italy
rossi@ismb.it

ABSTRACT

Event detection from social media is a challenging task due to the volume, the velocity and the variety of user-generated data requiring real-time processing. Despite recent works on this subject, a generalized and scalable approach that could be applied across languages and topics has not been consolidated, yet. In this paper, we propose a methodology for real-time event detection from Twitter data that allows users to select a topic of interest by defining a simple set of keywords and a matching rule. We implement the proposed methodology and evaluate it with real data to detect different types of events.

Keywords

Event detection, Social Media, Clustering, Machine Learning, Twitter

INTRODUCTION

There is an unprecedented number of personal mobile devices (smartphones, tablets) that can offer high-end capabilities in terms of computational power, connectivity, and real-time collection of geolocalized data including multimedia contents such as audio, images, videos. Such technological innovations are significantly expanding the sensing power of communication networks, giving momentum to a massively distributed and user-driven data collection process known as "crowdsensing". According to the International Telecommunication Union (Sanou 2013) there are more than 6.8 billions mobile phone subscriptions in the world. This translates in almost one mobile phone subscription per person, and this figure is globally rising each year. People equipped with mobile devices act as a mass of multimedia sensors, which generate a significant amount of real-time data, especially via social media platforms such as Facebook, YouTube, Flickr, and Twitter. The use of on-line social media platforms, coupled with the ubiquity of mobile devices capable of providing geolocated multimedia contents, offers the opportunity to exploit the generated data in order to detect in real-time the occurrence of an event. This capability can be very useful, especially when unexpected events such as natural hazards or terroristic attacks occur. Such events require prompt detection in order to activate response procedures. Also, tracking other kinds of events such as recurrent daily phenomena or planned gatherings can provide insights that can be exploited in commercial domains such as targeted advertisement. Twitter is the most studied social network in the emergency domain (Simon et al. 2015), probably due to the ease of sending and extracting information and to its open data policy. Twitter is categorized as a micro-blogging service, which is a form of communication that allows users to send brief text messages (formerly up to 140 characters, recently updated to 280), also known as Tweets, or media such as photographs or audio clips. By default, user posts are public, and they can be automatically retrieved using Twitter's Application Program Interfaces (API), which can be freely used under the limitations specified in the terms of service (*Twitter REST APIs* 2018). As shown in (Vieweg et al. 2010), Twitter is also used to give situational information during emergency events: during the Boston Bombing in 2013 it has been estimated that 27,800,000 Tweets have been written about that event. Furthermore, the information provided by Twitter can become viral (i.e., spread rapidly and on a vast scale across the Web) very easily thanks to Retweets, which are generated when a user re-posts (forwards) a message from another user. For all the aforementioned characteristics, we select Twitter as the social media platform to investigate.

However, including data from social media in emergency management processes poses several challenges, including the availability of location, the truthfulness and accuracy of the shared information, as well as the big volume, velocity, and variety of data. In this paper we propose a methodology aimed to detect in real-time events from an online social media network, and that can run on commodity hardware. The proposed methodology exploits machine learning algorithms such as clustering and classification and it is general with respect to the event type to be detected. Our methodology allows the end-user to define a set of relevant keywords and a matching criteria for each event of interest.

In the remainder of this paper we initially discuss related works and then we present our methodology. Next, we detail the implementation of the proposed methodology and show the results achieved with real Twitter data. Finally, we outline conclusions and future works.

RELATED WORKS

During the last decade, a number of studies have been done on event detection from social media data. Those who were most successful are volumetric approaches based on pre-selected keywords, like the one implemented in SensePlace (Maceachren et al. 2011), TEDAS (R. Li et al. 2012) and Twitcident (Abel et al. 2012), which enrich the emergency information using text semantics. However, these approaches use a predefined set of keywords to gather the initial data, hence they can detect only the event type for which they are designed for. Similar approaches are presented in (Marcus et al. 2011; Hila Becker 2011) where a Tweets' similarity metric based on a reference taxonomy is used. Another kind of approach is based on the Tweets' position, i.e. (Chen and Roy 2009) where a spatio-temporal analysis has been used to cluster Flickr photos, but this method is not designed to work with real-time data. Another method is the Topic Detection and Tracking (TDT), which had a lot of contributions in the past years such as (Yang et al. 1998; Allan et al. 1998). However, TDT works best with big documents and not with short messages like Tweets. Our work uses some ideas from the previous categories but it is different from all of them because it is based on a pipeline that involves: an online-clustering algorithm that uses a text similarity metric to group similar Tweets, a subsequent classifier to detect if an event happened, a scoring mechanism to rank the clusters containing the event, and a matching module to determine whether the event is among the user's interests. Such pipeline is generic and completely agnostic with respect to the event topic. Moreover, our approach can easily run on commodity hardware.

EVENT DETECTION METHODOLOGY

The concept of event has multiple definitions: it can be "a thing that happens or takes place, especially one of importance", or "a planned public or social occasion", or "each of several particular contests making up a sports competition", etc. We use the definition provided within the Information Retrieval (IR) community and reported in (Allan 2002): "an event is a flow of remarkable volume of temporally ordered messages" and we add to that a further constrain, i.e., that the ordered messages have to belong to the same topic.

In order to detect an event, we discretized the time in slots of size T, and denote by M the associated set of messages, each of which we assume to be composed by a multiset of words. We assume to have access to a given source, which produces a real-time stream of short text messages that we gather and process in a specific pipeline aimed to detect the occurrence of events matching with user-defined topics.

First, we apply to each message common pre-processing functions in order to remove not relevant contents. Next, we run an online clustering algorithm that groups similar messages together into a set of clusters C, where each cluster c_j is represented by its centroid r_j. At the end of each slot, we analyze C with a binary classifier in order to detect whether it contains an event or not. For the scope of this paper we assume that each slot contains at most one event, but the methodology can be easily extended to multiple events per slot. If the classifier detects an event, we assign a score to each cluster c_j and we select the highest in rank to identify the cluster that contains the event. Finally, we compare its content with a set of user-defined topics in order to decide whether the event (cluster) matches or not.

In the following subsections we detail each of the aforementioned steps of our methodology, which are represented in Fig. 1.

Text preprocessing

First, we apply common text pre-processing functions such as tokenization, stopwords and URL removal. Next, we apply a methodology to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For this goal we can use either stemming or lemmatization. Stemming usually refers to a crude heuristic process that cuts the ends of words, and often includes the removal of derivational affixes. Lemmatization uses a vocabulary and morphological analysis of words, aiming to return the base dictionary form of a word, which is known as the *lemma*. A *stemmer* is normally faster but less accurate than a *lemmatizer*. The selection among such methods does not affect the validity of our methodology, and it can be made according to a trade-off between computational speed and accuracy depending on the application needs and on the cost of computational resources. An in-depth analysis of the pros and cons of such methods is outside the scope of this work.

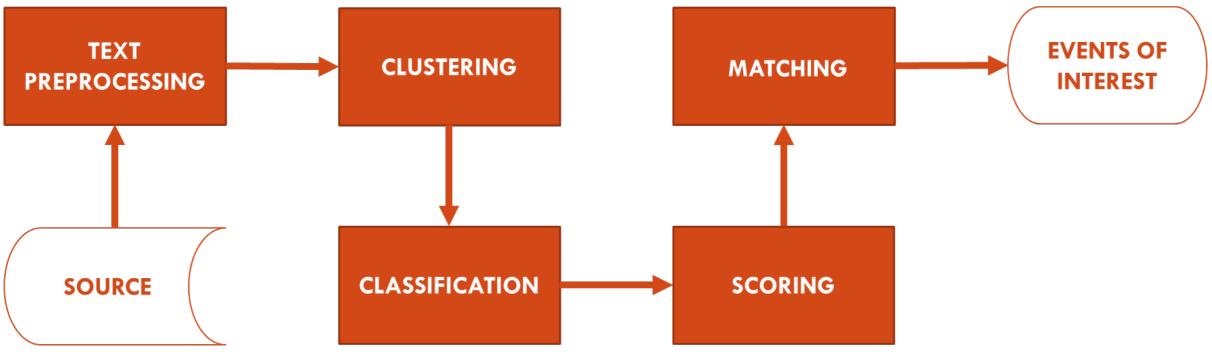


Figure 1. Block diagram of the proposed methodology

Clustering Algorithm

Some online (i.e. without initial knowledge) algorithms have been proposed in literature, e.g., (Liberty et al. 2014; Choromanska and Monteleoni 2011). However, the majority of them are based on k-means (Na et al. 2010), meaning that the maximum number of clusters have to be known a-priori. This assumption does not hold in our case because the number of clusters is assumed to be equal to the number of topics, which is unknown and can be very large. Other online algorithms, such as (Allan 2002; K. Li et al. 2011), can work with arbitrary number of clusters but they foresee heavy maintenance operations on clusters, e.g. merging, which prevent their real-time use in case of high throughput data such as the scenario we are considering. For the aforementioned reasons, we design a lightweight online and incremental (i.e. able to process real-time data) clustering algorithm that is structured in two phases: *attach* and *age*. The *attach* phase runs continuously with the objective to cluster incoming messages, while the *age* phase is executed every T, i.e., at the end of each time slot, with the aim to remove old and not relevant clusters. In the following paragraphs we detail such two phases.

The attach phase We represent messages M as well as clusters centroids R using the Bag Of Words (BOW)(Huang 2008) approach. We define a message $m_i = \{w_k, tf(w_k, m_i), \forall w_k \in m_i\}$, where $tf(w_k, m_i)$ is the Term Frequency (Huang 2008) of word w_k in the message m_i. We represent cluster centroids R in a similar way but with the Document Frequency $df(w_k, c_j)$ (Huang 2008) instead of the Term Frequency, i.e., $r_j = \{w_k, df(w_k, c_j) : \forall w_k \in m_i, \forall m_i \in c_j\}$, where:

$$df(w_k, c_j) = |\{m \in c_j : w_k \in m\}|$$

The clusters C are formed according to a similarity metric between a message m_i and a cluster c_j, i.e., $sim(m_i, c_j)$(Eq. 1), which sums all document frequencies of words that are both included in the message m_i and in the centroid r_j. Additionally, document frequencies of words in the centroid that are prefix of at least one word in the message m_j are added to $sim(m_i, c_j)$. In case multiple prefixes are found, we aggregate the longest matching one. If $sim(m_i, c_j)$ is below a certain threshold Th_s, m_i forms a new cluster, otherwise it is attached to the cluster having the highest sim. Ties are solved by randomly selecting the destination cluster.

$$sim(m_i, c_j) = \sum_{w_k \in m_i} df(w_k, c_j) \tag{1}$$

After a message m_i is attached to a cluster c_j, its centroid r_j is updated. In order to reduce the number of words in the centroids, we aggregate the document frequencies of words that are prefix of other ones, keeping the shortest word.

The age phase At the end of each temporal slot, the clusters are analyzed in order to understand if there is an event. Our goal is to have one event for each cluster and vice-versa (one cluster for each event) but it can happen that messages belonging to one event are split across more than one cluster. The age phase aims to avoid clusters growing too large and generic, thus attracting many topics, and to prevent the proliferation of very small clusters that represent topics no longer active. To tackle the first problem we remove the clusters older than a certain number Th_{a1} of slots, while to address the second one we remove the clusters which did not receive messages in the last Th_{a2} slots. When a cluster is removed, all its messages are deleted.

Classification

At the end of each time slot we have a set of clusters C and we want to remove the entire group if it does not contain any event. We achieve this goal by training a binary classifier and considering different set of features that can be grouped into four categories:

- Textual features: text metrics calculated using the words belonging to the set of centroids R, namely the Term-Frequency Inverse-Document-Frequency ($TFIDF$), which is computed according to its standard definition, normalized and averaged, considering each centroid r_j as a document and each word w_k as a term. The normalization is done with respect to the max tf and max idf.
- Temporal features: the Age of the cluster set $Age(C)$, which contains the cluster ages in terms of number of slots since their creation, and the cluster last updates $LU(C)$, which represents the most recent message attach time of all clusters.
- Dimensional features: they include the number of clusters $|C|$, the number of words of their centroids $size(R)$, how many new messages were attached to each cluster in the slot t ($\Delta_{c_j} = |c_j(t)| - |c_j(t-1)|$) over the initial cluster size $|c_j(t-1)|$ and over the total number of messages $|M|$ in the slot .
- Twitter specific features: for each cluster we count the number of followers (FR) and followees (FE) of the users that generated the messages included in the cluster.

We report in Table 1 the summary of the feature sets used.

Table 1. Feature sets used in the classification phase.

Textual	Dimensional
$TFIDF = \{\sum_{w_k \in r_j} tfidf(w_k, r_j, R) \forall r_j \in R\}$	$\lvert C\rvert$
$norm(TFIDF)$	$Size(R) = \{\lvert r_j\rvert, \forall r_j \in R\}$
$\mu(TFIDF)$	$\Delta_C = \{\Delta_{c_j}, \forall c_j \in C\}$
	$\Delta_{\%new} = \{\Delta_{c_j}/\lvert M\rvert, \forall c_j \in C\}$
	$\Delta_{\%new_c} = \{\Delta_{c_j}/\lvert c_j\rvert, \forall c_j \in C\}$
Temporal	**Twitter**
$Age(C)$	$FE = \{\lvert followee_{c_j}\rvert, \forall c_j \in C\}$
$LU(C)$	$FR = \{\lvert followers_{c_j}\rvert, \forall c_j \in C\}$

To train the binary classifier, we actually use as features the minimum, maximum, mean and variance of all the aforementioned features sets.

Scoring

If the binary classifier detects the presence of one event in the cluster set C, we rank the clusters in order to find the one that contains the event. Assuming that an event generates event-specific messages, i.e., messages containing a set of specific words in relation with the event topic, we have to spot the most specific cluster. In order to do so, we use the average tfidf of the words in each cluster. Also, assuming that the occurrence of an event corresponds to an increase in the volume of event-specific messages, we have to find a cluster that has substantially grown in the last slot, without favoring clusters of a specific size. For this purpose we use the percentage of new elements over the total number of messages in the temporal windows $|M|$ and the *0.5 double normalization* of the percentage of new elements being added to the cluster c_j over the cluster size $|c_j|$.

The formula of the cluster score $score(C_j)$ is reported in Equation 2. We select the cluster with the highest score as the cluster that represents the event.

$$score(C_j) = \frac{\sum_{w_k \in c_j} tfidf(w_k, c_j, c)}{|c_j|} * \frac{\Delta_{c_j}}{|M|} * \left(0.5 + 0.5 * \frac{\Delta_{c_j}}{|c_j|}\right) \tag{2}$$

Matching

Because all previous steps are agnostic with respect to the event topic to be detected, a further one is required in order to understand if the identified event is within the user-defined topics. We assume that the system lets users to define a set of words for each topic of interest, and a set of corresponding rules that are tested in the matching phase. For example, if at least one word in the user-defined set is found in the cluster centroid (not null intersection), or else, if all words of a user-defined set are included in the centroid (inclusion), then the cluster is selected. Any rule can be defined using simple algebra of sets. Note that all user-defined words must be either stemmed or lemmatized (depending on the technique used) in order to be comparable with the centroids. This approach keeps the methodology general to any kind of topic.

IMPLEMENTATION AND RESULTS

We implement the methodology using many technologies and frameworks, as shown in Fig. 2. We use a streamer to process real-time Twitter messages, implementing it in Python using Twython[1]. We temporally store Tweets in a MongoDB database[2]. The text processing phase is implemented using the Natural Language Toolkit (NLTK)(Xue 2010) while the clustering algorithm is implemented using Cython[3], which increase by 5 times the performance with respect to pure Python implementations. For the classification phase we manually label some cluster sets with the help of a Python script, then we analyze the data (to study feature correlation and to compare the performance of several machine learning classifiers) using Weka[4]. We implement the final classification algorithm, the Scoring and the Filter blocks in Python.

The Python distribution used in our work is Anaconda[5] 4.1.1 which includes, scipy[6], numpy[7], scikit-learn[8] and pandas[9].

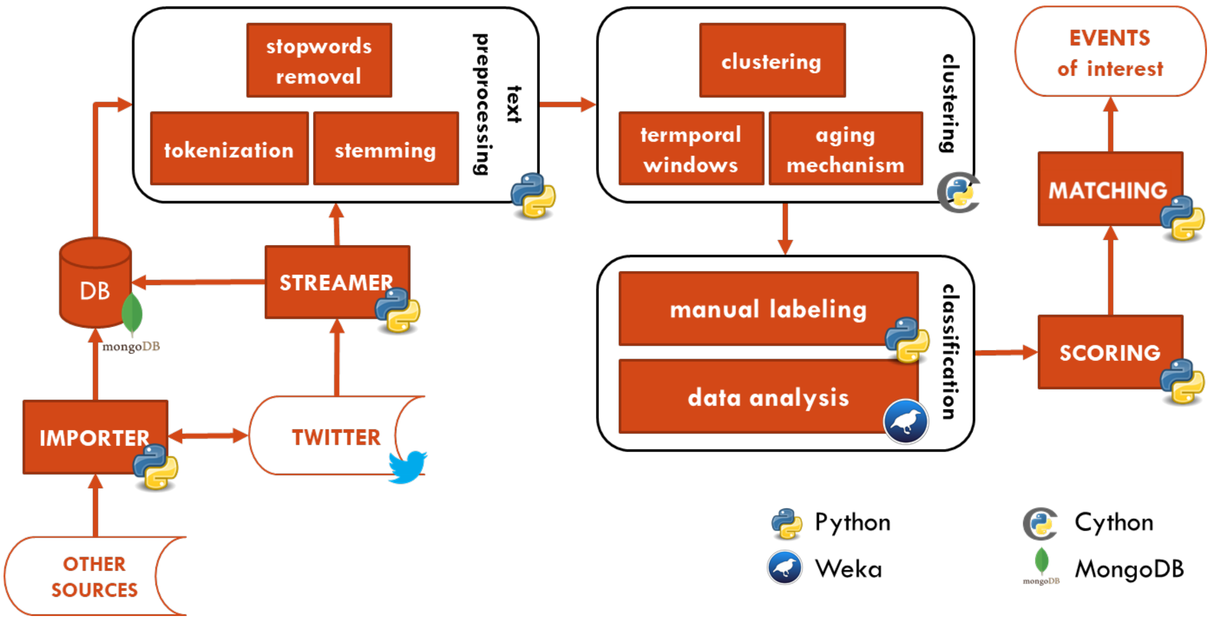


Figure 2. Block diagram of the implementation.

We use the Twitter Streaming API to gather Tweets in real-time (Andypiper and Niall 2015; Morstatter et al. 2013), knowing that this API gives up to 1% of the entire Twitter stream and that it can be filtered using several criteria, including keywords, language, position, etc. A common approach is to specify a set of words in order to limit the volume of messages but, as explained before, it is not viable for our goal that is to implement a generalized event

[1]https://twython.readthedocs.io/
[2]https://www.mongodb.com/
[3]http://cython.org/
[4]https://www.cs.waikato.ac.nz/ml/weka/
[5]https://www.continuum.io/
[6]https://www.scipy.org/
[7]http://www.numpy.org/
[8]http://scikit-learn.org
[9]http://pandas.pydata.org/

detection. Instead, we request all geolocalized messages (i.e., Tweets having latitude and longitude coordinates) in a specific language. The language is required because we use either a *stemmer* or a *lemmatizer*, both of which are language dependent, while the geolocation can help to identify where detected events take place and, at the same time, to reduce the number of Tweets to be processed.

For evaluating the implementation we capture and analyze in real-time an Italian stream from the 4th October 2016 to 15th March 2017[10].In order to train the binary classifier, we manually label 2000 randomly selected temporal-windows of 15 minutes (about 20 days). Using this training set, we apply a subset evaluation(Witten et al. 2016) which uses a greedy forward search to select the best feature set.

As a result, we find the following features as the most significant ones:

- $\mu(\{|c_j| \, \forall c_j \in C\})$: mean clusters size
- $\mu(\Delta_C)$: mean number of new tweets in the clusters
- $\max(\Delta_C)$: max number of new tweets in the clusters
- $\mu(TFIDF)$: mean TFIDF over the cluster set
- $\max(TFIDF)$: max TFIDF over the cluster set
- $Var(TFIDF)$: variance of TFIDF over the cluster set
- $\mu(norm(TFIDF))$: mean $norm(TDFIDF)$ over the cluster set
- $Var(norm(TFIDF))$: variance of $norm(TDFIDF)$ over the cluster set
- $\max(norm(TFIDF))$: max $norm(TDFIDF)$ over the cluster set
- $\min(\Delta_{\%new})$: minimum percentage of new elements in a cluster over the total number of new elements
- $Var(\Delta_{\%new})$: variance of the percentage of new elements in a cluster over the total number of new elements
- $\mu(\Delta_{\%new_C})$: mean percentage of new elements in a cluster over the total elements added to the same cluster

We try several binary classifiers and we select Random Forest (Biau 2012) using a ten fold cross-validation on the manually labeled training set, because it achieves the best results in term of Accuracy and F-Score, which are 94.6% and 86.7%, respectively.

The positive class represents the presence of an event in the analyzed temporal window and the negative class the absence of it.

The chosen parameters for the algorithms are:

- Slot time $T = 15min$;
- Clustering attach threshold $Th_s = 0.6$;
- Clustering aging threshold Th_{a1} (max time to live for a cluster) = 20 T (5 hours);
- Clustering aging threshold Th_{a2} (max time slots without receiving messages) = 2 T (30 min);

These parameters allow to keep under control the number of cluster (below 500), thus enabling a fast execution also on commodity hardware (we use a desktop computer[11]).

We report here the results achieved with three different event types. In detail we analyze:

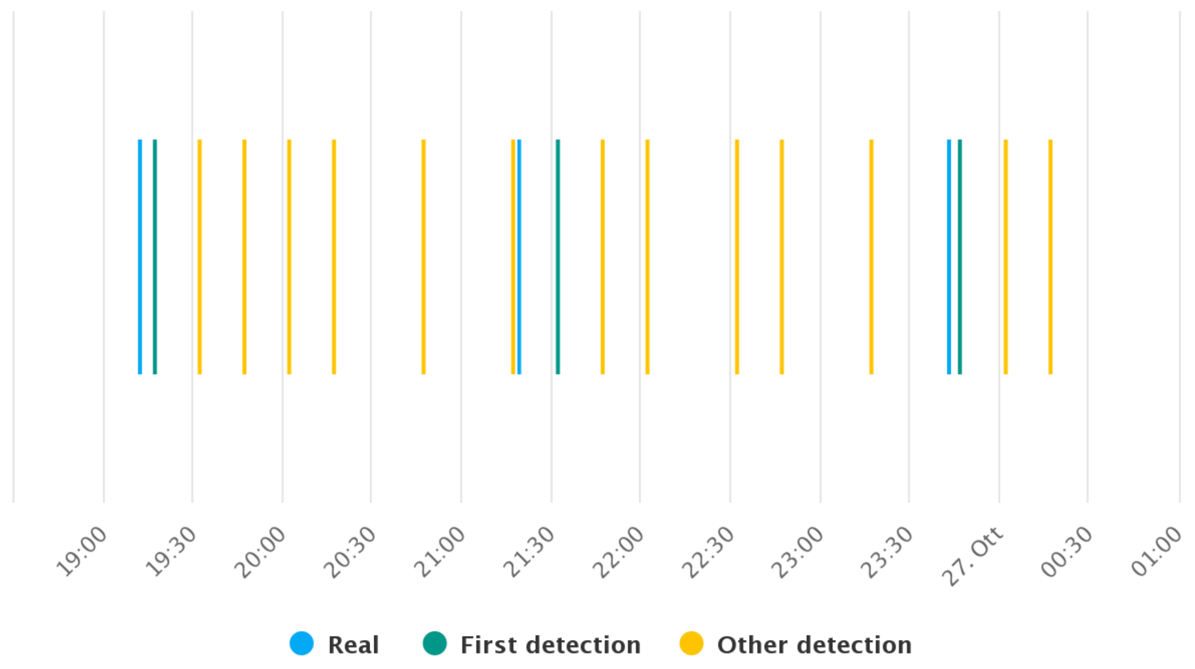
- One unplanned events - 26th October 2016: Earthquake in Central Italy;
- One recurrent events - 7th March 2017: nothing happens in the morning except the awakening of people that write "good morning to their friends";

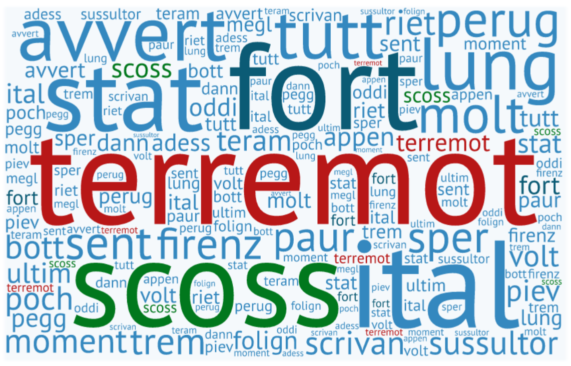
[10]On average 38000 tweets per day
[11]Intel® Core™ i5-7400T (3 GHz) - 16 GB RAM - Windows 10 - Python 3.5.2 64 bit

- One planned events - 7th March 2017: a Champions League match between Napoli and Real Madrid was played in the evening.

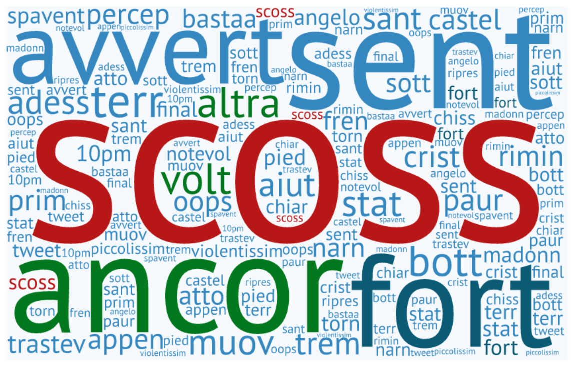
In the first case, we use the keyword *earthquake* in our final Matching module. This earthquake had 3 strong shakes in the same day[12], which are represented in Fig. 3a with blue lines. In the same Fig., we represent with green lines the first detections after each real shake occurred, while all subsequent ones are shown in yellow. In Fig. 3b and Fig. 3c we report the word cloud of the first and second shake, respectively.

(a) Earthquake - 26 October 2016 (Central Italy)





(b) Word cloud after first shake.



(c) Word cloud after second shake.

Figure 3. Earthquake Detections and Word Clouds

The word clouds can be effectively used to understand the topic of the event. In Fig. 3b (first earthquake shake) the biggest words are "terremot" and "scoss" that are the stemmed version of the Italian words "terremoto" and "scossa", which mean "earthquake" and "shake", respectively. Another relevant word is "fort", which is the stemmed version of "forte" that means "strong". In Fig. 3c we can see "ancor", which is the stemmed version of "ancora" that is "again", and this helps to understand that we are speaking about a new shake. The first shake is detected after 5 minutes, while the second shake is detected after 13 minutes. The third, and last, shake is detected after 4 minutes but its word cloud is very similar to the second shake and it is omitted for brevity.

[12]https://en.wikipedia.org/wiki/October_2016_Central_Italy_earthquakes

We report in Fig. 4 the other two cases (recurrent and planned event), for which we use for the matching the word "buongiorno" ("goodmoring") and "Napoli" (name of one of the two teams playing the match), respectively.

In Fig. 4a we show the volume of Tweets and the corresponding number of clusters over time. We represent with a yellow bar the timeframe in which the detections of the "buongiorno" event occurred. We count 11 detections in a timeframe of 2 hours. A clear peak is correspondence to the start of the football match is clearly evident from the graph. We note also that the total number of clusters stays well under 500 also in correspondence of the volume peak, confirming the robustness of our aging procedures. In Fig. 4b we plot the word cloud of the "buongiorno" event, which is dominated by the keywords "buon" and "giorn": the stemmed version of "buon giorno" that is "good-morning".

In Fig. 4d and Fig. 4c we show the detections of the football match and the word cloud of the first detection. The detections happened in proximity of the match kick-off (20:45) and end (22:15), while the word cloud are dominated by the stemmed names of teams that played the match, i.e., "Napol" and "Real Madrid".

CONCLUSIONS AND FUTURE WORKS

We have proposed a methodology for detecting any kind of event in real-time using Twitter streams. We allow the user to select the event topics to be detected by inserting a set of keywords and a matching rule for each topic of interest. We realized an implementation of such methodology and tested it with real data in three different cases. We verified that our approach is able to quickly detect all tested events and that the implementation was able to smoothly run on common hardware (desktop pc).

Future work will include a wider evaluation, an additional labeling for the event classifier, an optimization of the thresholds in order to improve the performances in terms of detection speed and reduce the clustering complexity. Additionally, we will test our solution with different similarity metrics and with word embedding techniques to evaluate eventual performance improvements. Finally, we will try to detect the location of the recognized event based on the tweets and to produce a panic map in near real-time.

ACKNOWLEDGEMENT

This work was partially funded by the European Commission through the I-REACT project (H2020-DRS-1-2015), grant agreement n.700256.

REFERENCES

Abel, F., Hauff, C., Houben, G.-J., Stronkman, R., and Tao, K. (2012). "Semantics + filtering + search = twitcident. Exploring information in social web streams". In: *Proceedings of the 23rd ACM conference on Hypertext and social media - HT '12*.

Allan, J. (2002). "Introduction to Topic Detection and Tracking". In: *Topic Detection and Tracking The Information Retrieval Series*, pp. 1–16.

Allan, J., Papka, R., and Lavrenko, V. (1998). "On-line new event detection and tracking". In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '98*.

Andypiper and Niall (2015). *Potential adjustments to Streaming API sample volumes*.

Biau, G. (2012). "Analysis of a Random Forests Model". In: *J. Mach. Learn. Res.* 13.1, pp. 1063–1095.

Chen, L. and Roy, A. (2009). "Event detection from flickr data through wavelet-based spatial analysis". In: *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09*.

Choromanska, A. and Monteleoni, C. (2011). "Online Clustering with Experts". In: *Proceedings of the 2011 International Conference on On-line Trading of Exploration and Exploitation 2 - Volume 26*. OTEAE'11. Bellevue, Washington, USA: JMLR.org, pp. 1–18.

Hila Becker Mor Naaman, L. G. (2011). "Beyond Trending Topics: Real-World Event Identification on Twitter". In: *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*.

Huang, A. (2008). *Similarity Measures for Text Document Clustering*.

Li, K., Yao, F., and Liu, R. (2011). "An online clustering algorithm". In: *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. Vol. 2, pp. 1104–1108.

Li, R., Lei, K. H., Khadiwala, R., and Chang, K. C.-C. (2012). "TEDAS: A Twitter-based Event Detection and Analysis System". In: *2012 IEEE 28th International Conference on Data Engineering*.

Liberty, E., Sriharsha, R., and Sviridenko, M. (2014). "An Algorithm for Online K-Means Clustering". In: *CoRR* abs/1412.5721. arXiv: 1412.5721.

Maceachren, A. M., Jaiswal, A., Robinson, A. C., Pezanowski, S., Savelyev, A., Mitra, P., Zhang, X., and Blanford, J. (2011). "SensePlace2: GeoTwitter analytics support for situational awareness". In: *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*.

Marcus, A., Bernstein, M. S., Badar, O., Karger, D. R., Madden, S., and Miller, R. C. (2011). "Twitinfo". In: *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*.

Morstatter, F., Pfeffer, U., Liu, H., and Carley, K. (2013). *Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose*.

Na, S., Xumin, L., and Yong, G. (2010). "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm". In: *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 63–67.

Sanou, B. (2013). *ICT Facts and Figures*. URL: `http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2013-e.pdf`.

Simon, T., Goldberg, A., and Adini, B. (2015). "Socializing in emergencies. A review of the use of social media in emergency situations". In: *International Journal of Information Management* 35.5, pp. 609–619.

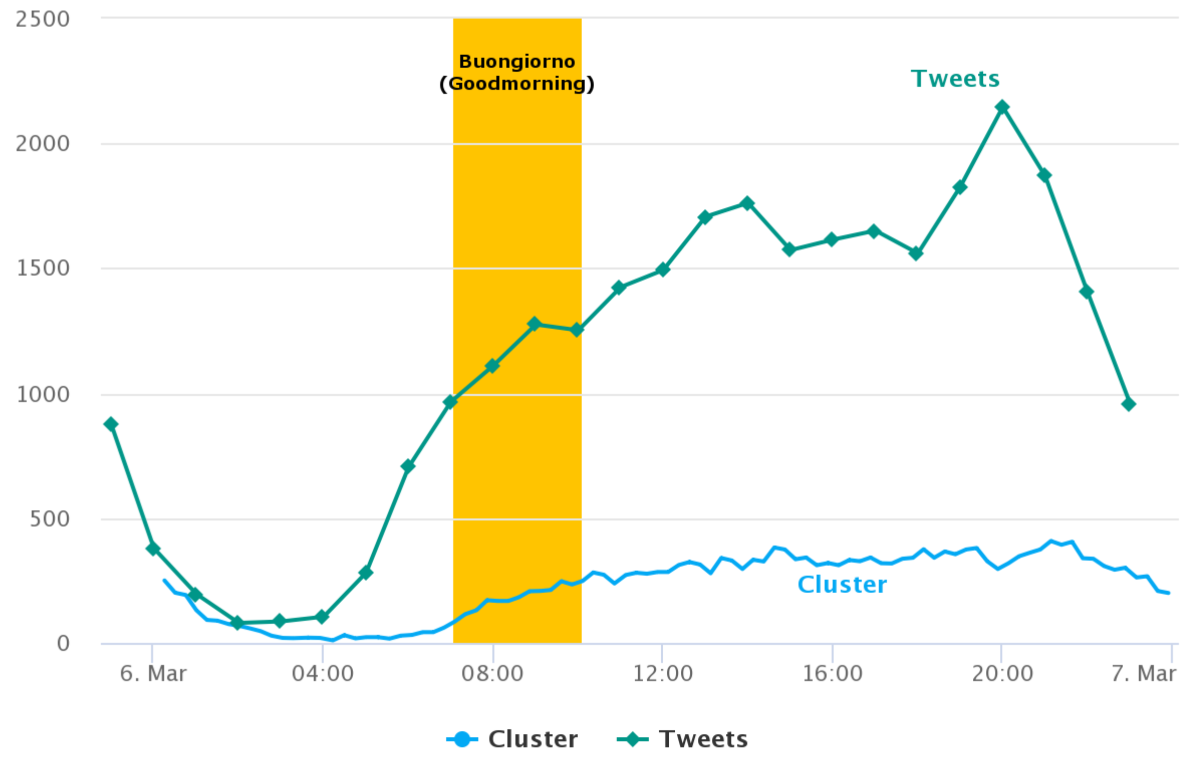
Twitter REST APIs (2018). URL: `https://dev.twitter.com/rest/public`.

Vieweg, S., Hughes, A. L., Starbird, K., and Palen, L. (2010). "Microblogging During Two Natural Hazards Events: What Twitter May Contribute to Situational Awareness". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, pp. 1079–1088.

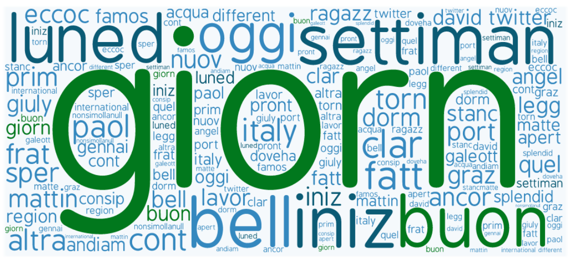
Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann Publisher.

Xue, N. (2010). "Steven Bird, Evan Klein and Edward Loper. Natural Language Processing with Python. O'Reilly Media, Inc. 2009. ISBN: 978-0-596-51649-9." In: *Natural Language Engineering* 17.03, pp. 419–424.

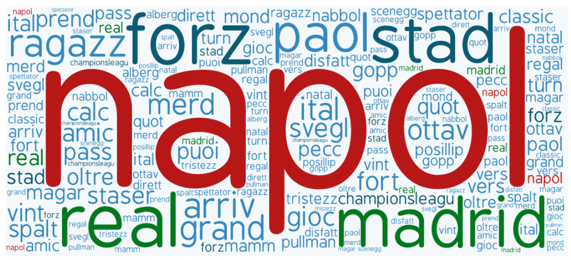
Yang, Y., Pierce, T., and Carbonell, J. (1998). "A study of retrospective and on-line event detection". In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '98*.



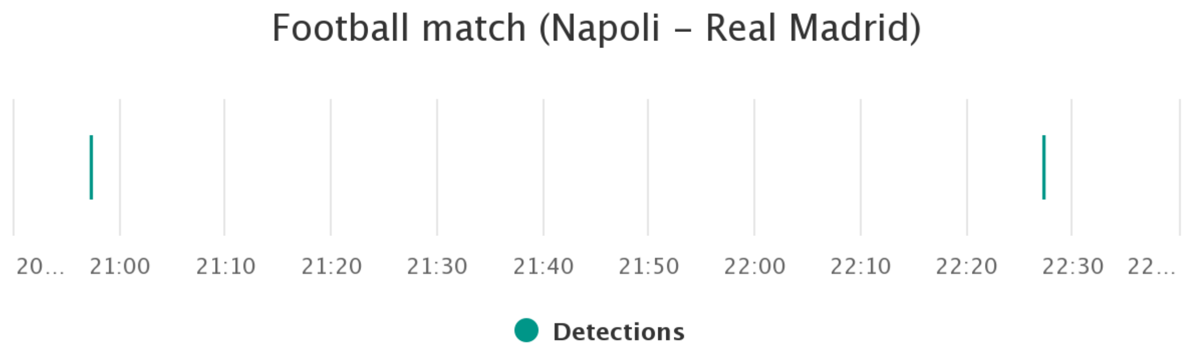
(a) Tweet volume and cluster number over time. The yellow bar represents the temporal window in which the the Good-morning event was detected (11 detections within 3 hours).



(b) Good-morning detection - Word Cloud



(c) Football match detection - Word Cloud



(d) Football match detections

Figure 4. Detection without filter module