Knowledge-Based Systems 211 (2021) 106563

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Evolutionary game theoretical on-line event detection over tweet streams

Rocco Di Girolamo [a], Christian Esposito [b], Vincenzo Moscato [a], Giancarlo Sperlí [a,*]

[a] University of Naples "Federico II", Department of Electrical Engineering and Information Technology (DIETI), Via Claudio 21, 80125, Italy
[b] University of Salerno, Department of Computer Science, Via Giovanni Paolo II 132, 84084 Fisciano, Italy

ARTICLE INFO

Article history:
Received 27 May 2020
Received in revised form 10 August 2020
Accepted 24 October 2020
Available online 2 November 2020

Keywords:
Event detection
Tweet categorization
Online social networks
Online clustering
Game theory
Dempster–Shafer theory

ABSTRACT

Current Online Social Networks represent a means for the continuous generation and distribution of information, which is slightly changed when moving from a user to another during the traversing of the network. Such an amount of information can overcome the capacity of a single user to manage it, so it would be useful to reduce it so that the user is able to have a summary of the information flowing the network. To this aim, it is of crucial importance to detect events within such an information stream, composing of the most representative words containing in each information instance, representing the event described by the set of tweet categorized together. There is a vast literature on off-line event detection on data-sets acquired from online social networks, but a similar solid set of approaches is missing if the detection has to be done on-line, which is demanding by the current applications. The driving idea described in this paper is to realize on-line clustering of tweets by leveraging on evolutionary game theory and the replicator dynamics, which have been used with success in many classification problems and/or multiobjective optimizations. We have adapted and enhanced a evolutionary clustering from the literature to meet the needs of on-line tweet clustering. Such a solution has been implemented according to the Kappa architectural model and assessed against state-of-the art approaches showing higher values of topic and keyword recall on two realistic data-sets.

1. Introduction

Over the last decade, *On-line Social Networks* (OSNs) reached a more and more increasing popularity, becoming a part of everyday life. Just observing what happens every day, millions of people use OSNs to inform friends about important events or to share information within social communities. In particular, microblogging platforms (i.e., Twitter and Snapchat) constitute nowadays the most natural environments where people can continuously report real-life events and share information, personal feelings and sentiments about public or private facts. In addition, the combined use of multimedia (still images, audio, videos) has given a new way of such information spreading, due to the old idiom that an "image is worth a thousands words".

OSNs and microblogs represent an important origin of information concerning events happening in a given location/country during a certain time period. Reporting those events could provide different perspectives to news items than traditional media due to re-forwarding what read in other posts by adding some personal opinions [1], and also valuable user sentiment about companies/products or a mean for alerts diffusion and emergency situations' management in a geo-referenced area.

We are witnessing a vast amount of data being exchanges on OSNs, most of them containing replicated information with a light difference among themselves, and single users have the difficulty to follow all of them, with the risk of missing valuable data. It is of pivotal importance to reduce such a redundancy by grouping similar tweets/posts by reporting to the users their salient features (*i.e.*, their most recurrent or important words), expressing the events that have originated them. Such an operation can be done off-line, by collecting a vast of data being exchanged within OSNs and processing them so as to obtain a report to be presented at the user, or on-line, so as to continuously show events to users as the stream of tweets/posts is happening. From the user perspective, the latter approach is more useful as it lowers the latency to get aware of something happening. As an example, OSNs are recently being proposed to detect natural disasters before any public announcements from the authorities, so as to quickly put in place escape plans for the involved population [2].

The basic assumption is that some related words would show an increase in the usage when an event is happening. An event

* Corresponding author.
E-mail addresses: digirolamorocco25@gmail.com (R. Di Girolamo), esposito@unisa.it (C. Esposito), vincenzo.moscato@unina.it (V. Moscato), giancarlo.sperli@unina.it (G. Sperlí).

https://doi.org/10.1016/j.knosys.2020.106563
0950-7051/

can be thus conventionally represented by a number of keywords showing burst in their appearance count over a tweet/post stream [3]. This is due to the fact that events can cause an information spread over a social community, when users start to share a given tweet or post containing the events itself, by adding some personal opinion or rephrasing the original event content. Thus, it is easy to infer when an event may happen, at it is possible to see a sudden and unpredictable happening of a massive spreading of semantically related tweets/posts over the network within a limited life time, where mostly all of them contains the same number of similar words, or their synonymous. Therefore, event detection is realized by conducting a clustering or linguistic pattern recognition of collected/streamed posts/tweets based on the shown equal/similar words and/or syntactical expressions within them, and returning to the user the clusters with a considerable number of elements (*i.e.*, over a given threshold). There is a relationship between event detection and tweet categorization [4], as tweets need to be clustered and classified, and the content shared among the cluster member can be extracted as representative of the event described by all the clustered tweets. It is therefore possible to conduct the categorization and classification and then realize the event detection, but in this work we preferred to perform both operations at the same time, by driving the classification based on the content similarity of the tweets. Despite in other works, such as [4,5], tweet classification is done by combining content and structural knowledge, we have preferred to only focus on the tweet textual representation. This is due to the fact that for event detection the underlying topological information of tweets does not represent key knowledge we can exploit; on the contrary to other applications of tweet classification, such as opinion propagation, where such structural knowledge is of upmost importance to study the phenomenon of interest. However, our approach is generic and it can be integrated with a structural content pipeline as done in [4].

Despite limiting the focus of the approach to only nouns and verbs and stemming them to obtain their root, it is hard to return clusters that are meaningful for the users, or the overhead to perform such a task may result unsuitable for an online execution. The current literature has leveraged on the Artificial Intelligence, Statistics, Natural Language Processing, and Big data Analytics for event detection, proposing approaches based on the detection of pre-defined events, or arbitrary events (alternatively defined as supervised and unsupervised detection). Despite pre-defined event detection is simpler to realize, its concrete applications are limited. The arbitrary event detection is more appealing as the approach learns autonomously the events happening within the OSNs, without any pre-configuration. An other classification is among off-line and on-line approaches. On the one hand, the first kind of approaches is based on the storing of tweets/posts within the cloud, and its consequent processing at the end of the day, or month (in the first case). On the other hand, the second class of approaches typically defines a small collection period, which is only some seconds and based on the spreading rate of the streams, and a fast processing for event detection.

The off-line approaches are easy and straightforward to implement by considering the widely known solutions of big data analytics, but implies a considerable delay of the detection time with respect to the event happening. The on-line approaches provide timely results, but may exhibits a lower accuracy than the previous ones, and are tougher to be implemented. Typically off-line approaches are unsupervised, while on-line ones are supervised. This is because the off-line approaches hold the overall view of the exchanged tweets/posts, are equipped with suitable computing resource and do not have any time restrictions, so to be able to look for arbitrary events. On the contrary, on-line approaches needs to fast return a result and have a limited view of the streams (only within the batch), so that they are typically pre-configured with the events of interest to be detected. Such a state-of-the-art literature poses limitations on the applicability and potential impact of such a technology, so it is important to achieve on-line approaches able to detect arbitrary events.

In this paper, we propose a novel solution for real-time detection of events in Twitter, based on the application of information filtering and clustering techniques by adapting the evolutionary clustering in [6] to face the requirements of on-line tweet clustering. Basically, a series of rules to filter the tweets are proposed so as to pre-process them and remove possible noise factors that can compromise the event detection, such as hashtags, retweets and mere syntactic sugar. Afterwards, the tweets are used to fill in a micro-batch and execute clustering. The literature on clustering is extremely vast, and its application to tweets and/or posts has been extensive in the recent decade. However, those related to arbitrary event detection to be run in real-time are limited and exhibit inadequate performance. Our driving idea is to model the overall clustering task as a evolutionary non-cooperative game [6], where players pick an item from the micro-batch or release one based on the reward they can achieve as a measure of item similarity, and the equilibrium is obtained when all the players have the same item selection, expressing a cluster. The evolutionary operators are integrated so as to study the strategy spreading within the population of players and reaching a stable state equilibrium within it (no external strategy can affect the population by bringing one of its individuals to change its strategy with the external one). Such an approach has been favored as the application of evolution brings in the game an indirect form of cooperative behavior, resulting into higher quality levels and lower price of anarchy and distance to the Pareto front of the optimization problem underlying the clustering [7]. Such a proposed work has been implemented within the context of a typical Big Data analytics platform as Apache Cassandra and the Kappa architectural model and assessed with two realistic data-sets obtained from Tweeter, exhibiting higher accuracy and latency than its competing related works.

The paper is organized as in the following. Section 2 describes and analyzes the existing literature on the topic of event detection within the context of OSNs, by indicating their drawbacks. Section 3 presents in details the proposed pre-filtering and clustering approach, while Section 4 illustrates the realized proof-of-concept and its consequent assessment to highlight the achievable performance. We conclude the paper in Section 5 with some final remarks and a plan for future work.

2. Related works

In the last decade, a plethora of event detection approaches have been proposed in the literature to discover real-world events from social data stream. Recent surveys, such as [8,9], have already provided a detailed review of the most diffused techniques. In particular, it is possible to characterize event detection approaches using the following classification [10] in two distinct classes: *Arbitrary Event Detection* and *Pre-defined Event Detection*. The first group encompasses Clustering, Lexical, Graph-based, and Statistical techniques (for the last one Latent Variable Models and Miscellaneous Models are being applied). The second includes Learning and Lexical techniques.

For the arbitrary event detection techniques, the main objective is detecting any kinds of events without any prior information about the event type or context. Therefore, the basic idea behind this type is to group similar data in order to generate a set of candidate events for further processing by means of clustering algorithms but still lexical, statistical, and graph-based techniques. For the second kind of available techniques,

there is a set of prior details about the events to be detected (e.g., earthquakes, crimes, or traffic jams) that can be properly exploited to label social stream w.r.t. some given classes. Thus, the core of such approaches is a classification (learning-based or lexical-based) technique.

Another interesting classification divides the approaches into the following categories:

- *Document-pivot*: This category comprises methods that represent documents as items to be grouped/clustered using some similarity measure.
- *Feature-pivot*: These are based on detecting abnormal patterns in the appearance of features, such as words. Once an event happens, the expected frequency of a feature will be abnormal comparing to its historical behavior, thus indicating a potential new event.
- *Topic modeling*: This includes methods that utilize statistical/probabilistic models to identify events as latent variables in the documents.

In the following, we describe some recent event detection techniques form which our approach – that falls in the clustering based techniques for arbitrary event detection category – takes its roots. In addition, Table 2 reports a summary of the analyzed techniques.

Concerning arbitrary event detection techniques, Arin et al. [11] presented the I-TWEC system: a document-pivot tweets' clustering tool, which allows users to interactively select and merge clusters of documents on the basis of their semantic similarity. An online document clustering method has been also proposed by Aggarwal et al. [12] using a similarity measure based on tweets' content and users' proximity. In a similar way, Comito et al. [13] defined an online cluster methodology (BEaTS) that combines textual and temporal features for identifying clusters of tweets as event summarizing the examined tweets into the cluster centroid. Form the other hand, Petrovic et al. [14] introduced a document pivoting strategy (Doc-p) with Locality-Sensitive Hashing (LSH) clustering. In addition, a general framework for supporting event online clustering, using temporal,spatial and textual features, has been proposed in [15]. In [16], after a typical pre-processing of the text, the pipeline includes a singleton set populated with an SVM with a linear kernel. This is a supervised approach, which needs a training phase and data. In on-line event detection, such a supervised classification approach is impractical unless we want to detect a specific known event happening, such as an earthquake in disaster management or fake news.

Among lexical/statistical feature-pivot techniques, Choi and Park [17] described a streaming method based on High Utility Pattern Mining (HUPM) aiming to identify emerging events from documents represented as a set of words with high frequency and utility in sliding temporal windows of tweets, while Aiello et al. [18] presented a pattern mining technique (namely, SFPM) based on n-grams cooccurrence and tdf–idf. Also in [19], a sentiment classification model is proposed by using generating domain-specific sentiment lexicons.

A statistical technique based on a latent topic modeling has been proposed by [20] to handle tweets' data sparsity, using a particular LDA algorithm that considers super tweet (aggregation of similar tweets) as a document instead of single tweet. Eventually within the graph-based approaches, Enhanced Heartbeat Graph (EHG) has been proposed by Saeed et al. [21] leveraging KL-divergence to generate temporal graphs of terms from documents and centrality measures to determine the event-related topics.

Regarding feature-pivot techniques for pre-defined events, Dabiri et al. [22] used particular social stream features to realize a multi-class approach based on CNN and word embedding techniques for classifying tweets into three different categories (i.e., non-traffic, traffic incident and traffic information and condition). In a similar way, D'Andrea et al. [23] introduced a binary classification engine for categorizing tweets into two classes (i.e., traffic and not-traffic) using an NLP pipeline and a SVM.

Zhang et al. [24] proposed GeoBurst relying on an authority measure to analyze geo-topic correlations among tweets; in particular, representative tweets (*pivots*) are identified within sliding windows to generate candidate events, composed by tweets similar to the pivots, which are compared with historical activities to unveil spatio-temporally features used to identify relevant events. An advanced version, namely GeoBurst+, has been then proposed by [25] for identifying local events from Twitter stream using a cross-modal authority measure and geo-topical features. Furthermore, Zhang et al. [26] developed a multimodal embedding method, called TrioVecEvent, for online local event detection through a two-step schema. Firstly, tweets are divided in sliding windows according to geo-topic clusters produced through a Bayesian mixture model capturing text semantic by learning multimodal embedding of the location, time and text. Successively, different features have been extracted for classifying the geo-topic clusters with respect to local event.

Khodabakhsh et al. [27] developed a semantic approach that models life events using word embedded techniques. In addition, the authors propose a multi-class classifiers to assign a tweet to a specific life event using as input features its mover's similarity with respect to life events and tweet sequence before or after the examined tweet. Amato et al. [28] combined tweets' multimedia features together with the related diffusion properties within social networks to detect and better manage extreme events. A shared multi-view data representation model has been described in [29] that learns intrinsic structures among heterogeneous data from online news and social media. Also, the work in [4] structural information are combined with textual processing. [19]

Finally, an online method, leveraging a one-off semi-supervised initialization technique, has been described in [30] for detecting unusual bursts in discrete-time signals extracted from Twitter social stream.

Indeed, the event detection approaches can be also classified into supervised and unsupervised methods. The former relies on machine learning classification schemata (e.g. [28–30]) or embedding models (e.g. [22,27]) to learn main features for identifying global or local events. The latter have the goal of identifying events on the basis of frequent pattern mining (e.g. [17,21]) or clustering techniques (e.g. [11–13,24]). The cited approaches suffer from different drawbacks. On the one hand, learning based methods require a training phase that is a very difficult task due to the variability of messages' lexicon. In a similar manner, pattern mining approaches assume a reduced variability of messages' lexicon and often are domain-dependent and present scalability problems. On the other hand, the majority of clustering techniques are not able to capture and follow in a correct way event temporal dynamics.

More recently, social sensing arose as new paradigm to collect observations from humans. Here, one of the main issue concerns the problem of identifying the truth events from unlabeled raw tweets. To this aim, Zhang et al. [31] developed a framework, called *Scalable Streaming Truth Discovery* (SSTD), that relies on dynamic truth discovery scheme based on Hidden Markov Models (HMM) to infer the evolving truth of reported claims. In [32], the authors described an approach to deal with a multidimensional maximum likelihood estimation problem with the aim to estimate jointly the truthfulness and topic relevance of claims. In [33], global and local context of a textual corpus has been

Table 1
State-of-art techniques.

Technique	Event	Type	Summary	Cons
Singh et al. [20]	Arbitrary	Statistical	LDA for tweets' data sparsity	No clusters' evolution
I-TWEC[11]	Arbitrary	Clustering	Online clustering using semantic and lexical similarity	No clusters' evolution
Aggarwal et al.[12]	Arbitrary	Clustering	Online clustering based on tweets' similarities and user proximity	No clusters' evolution
Doc-p)[14]	Arbitrary	Clustering	Clustering based on Locality-Sensitive Hashing	No clusters' evolution
Alsaedi et al.[15]	Arbitrary	Clustering	Online clustering using temporal, spatial and textual features.	No clusters' evolution
BEaTS[13]	Arbitrary	Clustering	Online clustering combining textual and temporal features	No clusters' evolution
HUPM[17]	Arbitrary	Lexical	Pattern mining analyzing words' frequency and utility measures	Lexicon variability
SFPM[18]	Arbitrary	Lexical	Pattern mining based on n-grams cooccurrence and tdf–idf	Lexicon variability
EHG[21]	Arbitrary	Graph-based	Pattern mining using temporal graphs	Graph size
Dabiri et al.[22]	Pre-defined	Learning based	Multi classification based on CNN and word embedding	Dataset Availability
D'Andrea et al.[23]	Pre-defined	Learning based	Binary classification engine using NLP pipeline and SVM	Dataset Availability
GeoBurst[24]	Pre-defined	Learning based	Spatio-temporal features to identify geo-located events	Dataset Availability
GeoBurst+[25]	Pre-defined	Learning based	Cross-modal authority and geo-topical features to unveil events	Dataset Availability
TrioVecEvent[26]	Pre-defined	Learning based	Multimodal embedding method	Dataset Availability
Poblete et al.[30]	Pre-defined	Learning based	Semi-supervised initialization technique	Dataset Availability
Yang et al.[29]	Pre-defined	Learning based	Pattern mining	Dataset Availability
Khodabakhsh et al.[27]	Pre-defined	Learning/Lexical	Semantic approach based on word embedding	Lexicon variability
Amato et al. [28]	Pre-defined	Learning/Lexical	Multimedia and diffusion properties for extreme events	Lexicon variability

analyzed through a unified language model based on matrix factorization techniques with the goal of discovering topics, learning also word embedding collaboratively.

Furthermore, Li et al. [34] investigated truth discovery problem by introducing a privacy-preserving truth discovery mechanism considering jointly utility and privacy. In particular, the authors conducted a weighted aggregation, whose weights are based on information quality, among users' data, that is independently perturbed. Another perturbation approach, relying on two layers mechanism to independently sample users' private probabilities from a hyper distribution, has been proposed by Li et al. [35] with the aim to capture the user quality and aggregate user-contributed answers according to a weighted combination. Table 1 summarizes the state-of-art techniques, also detailing type of events and proposed methodologies.

In order to overcome the discussed issues, we propose arbitrary event detection technique based on a dynamic online clustering strategy using three types of actions: create, merge and association. In the following, we summarize the main novelties of the proposed approach:

- The proposed approach is a methodology that avoids to use a training model (differently from machine learning methods), which could be difficult to define due to the highly variable lexicon of a social stream.
- Our general-purpose technique can easily support different domains, unlike the methods that are focused on traffic information.
- Our approach is not based on geographic information that is not always available and often misleading, also defining a dynamic methodology for creating and modifying clusters based on the *Game Theory* in order to correctly capture event dynamics.
- The proposed method can be used for very large datasets and Big Data applications by means proper computing infrastructures, differently from others that require very costly similarity metrics.
- In our approach different methodologies for identifying truth events from unlabeled raw tweets can be easily embedded, by customizing and extending the information filtering rules.

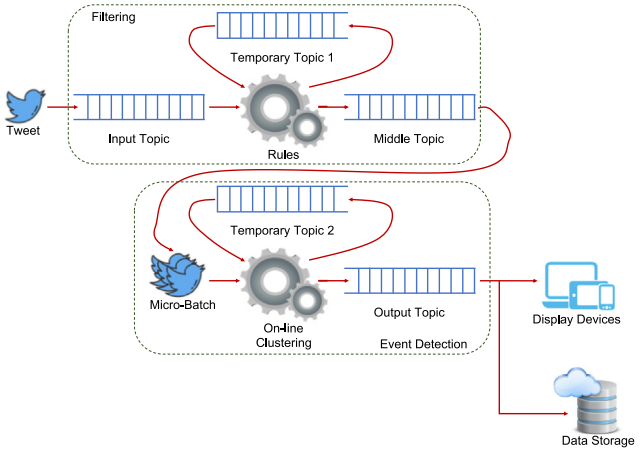


Fig. 1. Event detection workflow.

3. Methodology

The proposed approach, whose process overview is depicted in Fig. 1, aims at unveiling events from a social stream, using an on-line clustering technique based on the Game Theory.

To this aim, the first subsection introduces a series of definitions supporting the description of our on-line clustering solution. The second subsection describes a pre-processing phase to be done before performing the clustering, which is described in the two subsections. Specifically, the third subsection is then devoted to present the overall clustering approach, while the last one presents in details the game formulation of the clustering and its consequent resolution for capturing event dynamics.

3.1. Background

A social data stream may contain heterogeneous multimedia information (*i.e.*, text, images and videos), so that we can exploit the following definition to describe such a kind of stream.

Definition 3.1 (*Social Data Stream*)**.** Let m_t be a given multimedia object arriving at the time t, a *Social Data Stream* (SDS) can be defined as the following set of elements:

$$SDS = \{m_{t_1}^1, \ldots, m_{t_1}^{k_1}, \ldots, m_{t_2}^1, \ldots, m_{t_2}^{k_2}, \ldots, m_{t_n}^1, \ldots, m_{t_n}^{k_n}\} \quad (1)$$

k_i being the number of objects arrived at the time t_i and $t_1 < t_2 < \cdots < t_n$

Each multimedia object or *message* can be composed by one or more assets (e.g., a post in Facebook can be constituted by a simple text or by a text with images and videos) and has a set of attributes: the identifier of the user who has published it, the identifier of the post to which it belongs, in the case of Twitter, the number of retweets and citations, the set of hashtags/keywords, etc. With an abuse of notation, we indicate with $m[att]$ the value of the attribute att for the message m.

In our vision, an event is a phenomenon that can happen or has already happened and that other people are already talking about it or will start immediately discussing, and can be associated to a particular set of messages within a social data stream.

Definition 3.2 (*Event*)**.** Given a Social Data Stream (SDS) an *event* e is a set of objects belonging to the SDS sequence:

$$e = \{m_{t_j}^i\}, i \in [1, \ldots, K], j \in [1, \ldots, N] \quad (2)$$

K and N being the total number of messages and of temporal instants related to the social stream, respectively.

An event is then defined as *interesting* if it has an increasing number of people talking about it, but it does not share similarities to any of the previously detected events (in terms of keywords describing past and current events). The peculiarity of the event is in our vision an evidence of its relevance. Here, we are interested in detecting interesting events from the social streams: specifically in our work we are interested to Twitter data streams, but the approach is generic.

In addition, each event is characterized by a *topic*, *i.e.* a set of *words* inferred by the messages' multimedia content that semantically described the event itself.

Event detection consists in clustering messages within a social stream based on a certain set of criteria measuring the similarity among the various grouped messages. By restricting our investigation to only the text within the social streams, clustering is done by looking at keywords in common, according to the following definition.

Definition 3.3 (*Cluster*)**.** Given a Social Data Stream (SDS) a *cluster* C_x is a set of objects belonging to the SDS sequence showing some text similarity. In particular, a *Cluster* can be expressed as a set of words and their relative occurrence rate:

$$C_x = \{w_1 : n_1, w_2 : n_2, \ldots, w_m : n_m\}, \quad (3)$$

w_i represents the ith word and n_i is the number of occurrences for the ith word, respectively.

Given a threshold y, it is possible to obtain a subset of the most significant and representative words within the cluster. In the preliminary phase of this work, various representations of this threshold have been investigated: the average or the median of the various occurrence rates, 0 and 1.

The median proved to be the one exposing the best results, as it is the statistical measure being able to tolerate outliers, so that a member of a given cluster is the objects from a social stream whose list of words representative of the cluster is greater than the given threshold. As an example, let us consider the case represented in Fig. 2 where a cluster is described by 4 words, each with a given number of occurrences: the first one has 25 occurrences, the second 8, the third 2 and the last one 1. The lines in the figure represents the various possible thresholds to be applied.

By using the mean, the cluster is represented by only the first word, with the median, by the first two words, with a threshold equal to 1 by the first three words, and last by a threshold of 0 by all the words. It derives that with an average, only the word with the highest frequency is used to represent the cluster, implying a considerable reduction in the expressively of the cluster description. On the contrary a threshold of 1 or 0 bring too many words to describe the cluster, causing a too broad clustering.

The median represents a good trade-off in this sense.

3.2. Pre-processing

The tweets or any other object taken from a social stream cannot be directly clustered for the event detection, but they are needed to pre-processed in order to filter out noisy information that do not correspond to any interesting event. Such a pre-processing phase is done according to a rule-based approach, where tweets are processed according to the following three rules.

Definition 3.4 (*Rule 1*)**.** Let SDS and $H = \{h_1, \ldots, h_m\}$ be respectively a social data stream and a set of hashtags extracted from the messages within SDS. A message $m \in$ SDS will be considered if and only it contains an hashtag $h \in H$ such that $f(h) \geq \tau_1$, where $f(h)$ is the occurrences frequency of the examined hashtag within the stream and τ_1 is a given threshold.

We provide the following example to show how this particular rule works.

Example 3.1. If a concert by a famous singer is presented, a hashtag will be created to properly identify this event. In the days prior to the concert, users will talk about it in an accentuated way and this hashtag will be obviously repeated much more than usual. After the concert, users will stop talking about it. This allows us to identify a bell-like pattern of the frequency of occurrences of the concert hashtag. When this trend exceeds a certain threshold, for us it can be related to an event of interest.

We want to note that it is better to avoid to filter out immediately tweets that do not match the first rule, because some of them could be refer to an event for which an hashtag has not been still defined. For this reason the second criterion concerns the re-consideration of a given message.

Definition 3.5 (*Rule 2*)**.** Let SDS and W_t be respectively a social data stream and a given time-window centered in t. A message $m \in$ SDS will be considered if and only if $(m[retweet] + m[citation]) \geq \tau_2$ where $m[retweet]$ and $m[citation]$ are respectively the number of retweets and citations of message m within the temporal interval W.

To better understand this criteria we provide the following example.

Example 3.2. If an earthquake occurs in a city, users will probably post a message and it will be spread by other users in order to maximize the diffusion of the news without creating any hashtag. In this case, rule 1 does not help us identify the event bat the second one may serve our purposes. Thus in this case, the event is represented by the sudden diffusion of the news.

Finally, a last rule concerns importance of the user who has published a given message.

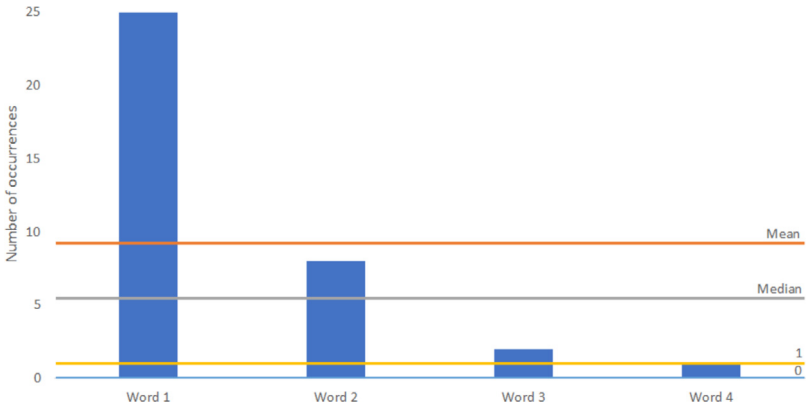


Fig. 2. Different threshold to describe a cluster of 4 words.

Definition 3.6 (*Rule 3*)**.** Let SDS and U be respectively a social data stream and a set of users publishing the messages within SDS. A message m will be considered if and only if the user u owner of m exceeds a threshold τ_3 in terms of number of followers.

In the following we provide an example that describes how such a criterion works.

Example 3.3. If Donald Trump (about 60M of followers) publishes a message, even if it is not discussed on social network, for us it however represents an event of interest.

This third criterion allows to identify news published by specific users, also known as "influencers".

3.3. On-line clustering

The stream of tweets filtered by using the above presented rules represents the input of an online clustering approach, partially inspired by [6] by means of four distinct operations.

Our approach aims at realizing online clustering, so that the tweets are grouped together as soon as they are published. However, a purely online clustering is not possible, as a minimum set of tweets are needed so as to run the clustering. For this reason, we have defined a sampling period during which the tweets are collected in a so-called *microbatch*.

When a microbatch is available, a cosine similarity of the tweets among themselves and with the already found clusters is evaluated (to this aim we considered keywords extracted from tweets' text and representative words of clusters by means of a word2vector approach), and the online clustering is executed.[1]

Specifically, the overall clustering process has been modeled as a multi-player non-cooperative game, in which the number of cluster is not pre-defined by they are sequentially extracted from the tweets provided in input. In such a game theoretic formulation, each player aims to maximize their own profit (defined throughout a given *payoff matrix*) based on an estimation of the goodness of the applied strategies with respect to those adopted by the other players.

The algorithm terminates its execution when it finds a Nash equilibrium, in which each player finds not profitable to change its strategy (*i.e.*, such a change is not able to bring an higher payoff).

In our formulation, each incoming tweet is seen as a player, with a set of strategies and actions to be done during playing the game. The approach is concluded by a refactoring/improvement of the solution obtained throughout the game theoretic formulation. Such a sequence of operations are infinitely repeated as tweets arrive. The clustering process has to return a maximally coherent sets of tweets from the input stream, satisfying the criteria of internal coherency (*i.e.*, all the tweets within a cluster have a high mutual similarity) and external coherency (*i.e.*, all the tweets not belonging to a cluster have a low mutual similarity with those in the cluster). Therefore these criteria can be assumed for the formulation of the payoff function.

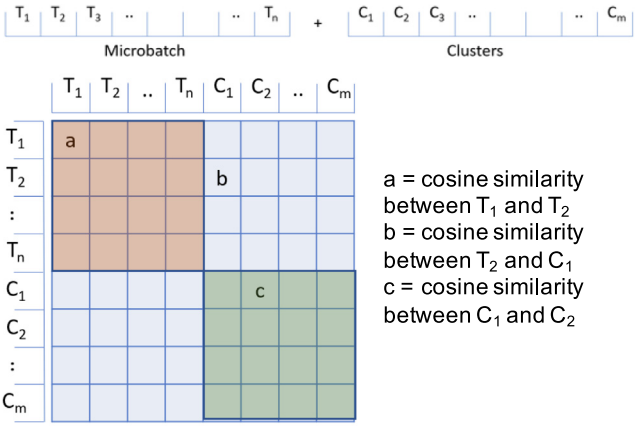


Fig. 3. Matrix construction from the list of microbatch and representative words for the found clusters.

At the beginning a microbatch is needed to be built. Given the stream of incoming tweets from the rule-based pre-processing, each tweet will be stored in a given progressive position of a list (initially empty) of a length equal to n, which is a parameter of the algorithm and determines its duration (*i.e.*, which lasts until the list is not filled up). When the list is full, it represents the microbatch to be processed. It is possible that we have a continuous arrival of tweets, but it also happens that there may not be tweets provided for a certain time period. In order to avoid to have tweets waiting too much for a microbatch to be built, there is a timeout on time of filling up the list (equal to 30 s), and at its expiration the list (even if it not completed filled) will be considered a microbatch with a length lower than n.

Once a microbatch has been defined, it is appended with a list of length m, which is the number of clusters found at that time, containing all the representative words for each cluster. The obtained list is multiplied against itself according to the Cartesian product, so as to obtain a matrix with all ordered pairs (a, b) where a and b are elements from the list. Afterwards, at each

[1] To reduce representative words of tweets and clusters, and make more efficient clustering, we can optionally remove synonyms considering the terms' synsets of WordNet (https://wordnet.princeton.edu/) lexical database before the word2vector mapping.

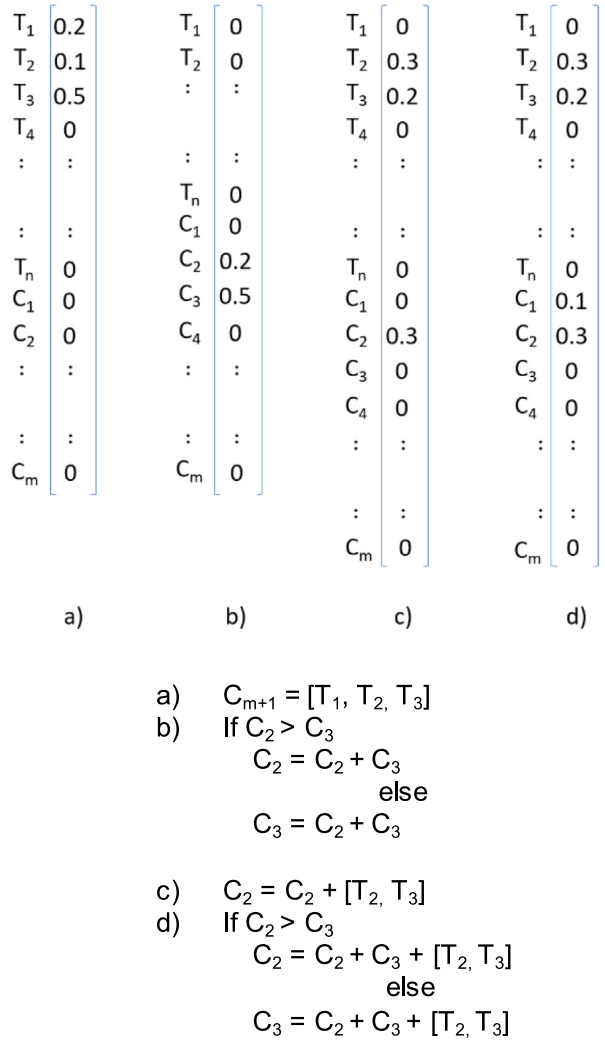


Fig. 4. Possible final population states and their meanings.

pair, a cosine similarity between the words in the two elements is constituted, as depicted in Fig. 3. Such a matrix is divided in four portions, in the one in the upper left side (with a size of *nxn* and in red in the figure), there is a measure of the similarities among the tweets; while the one in the lower right side (with size *mxm*), there is a measure of the similarity among the clusters.

The two remaining sub-matrices describes the similarities of the tweets with respect to the various clusters. There two sub-matrix represents one the transpose of the second: the first one has tweets arranged over the columns while the second one has the tweets over the rows. The sub-matrix with the tweet mutual similarity has the primary diagonal (whose elements represent the similarity of a tweet with itself) with all values set to zero (or general fairly low values) in order to avoid clusters sharing some elements.

The input of our clustering game is constituted of the similarity matrix and a random vector with length equal to $m + n$, which indicates the initial population state. At the arrival to an equilibrium, or at the expiration of a timeout, the game will return a novel vector with length equal to $m + n$ representing the final population state.

This vector indicates the winning strategy, whose examples are provided in Fig. 4:

- Case (a) *Creation of a novel cluster*: If the vector contains one or more non-null values within the first *n* positions, while all the remaining ones are null, a novel cluster with the items associated with the non null values has to be created.
- Case (b) *Merge of two or more clusters*: if all the *n* positions exhibits null values, while some of the remaining ones at the *m* positions at the end of the vector are not null, the relative clusters have to be merged.
- Case (c) *Association of items to existing clusters*: if there are some non-null values in the first *n* positions and a non-null value in one, namely ν, of the remaining *m* positions, then the relative items have to be associated to the ν-th cluster.
- Case (d) *Merge of two or more clusters and association of some items to the merged cluster*: if there are non null values in the first *n* and the last *m* positions of the vector, a new cluster has to be created by encompassing the items and merging the clusters associated to the non null values.

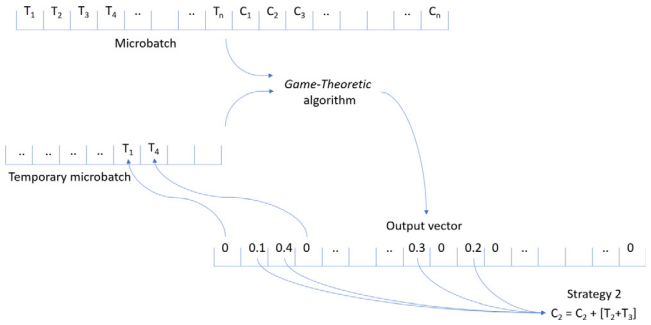


Fig. 5. Improvement in case of missing equilibrium convergence.

If the final population state presents all the elements as null, none of the previous strategies can be applied and another run of the game with new collected items will be performed. In fact, both in the case of an equilibrium not being achieved or a given strategy to emerge as the dominant one, a refactoring and improvement of the game is elaborated as in Fig. 5.

Specifically, items not being clustered (as in the final population state they presented a null value) are not lost but re-proposed to the next run of the game by building a novel microbatch including also some of the tweets received during the game run, and an updated list of clusters after having considered the dominant strategy represented by the population state.

3.4. Game theoretic clustering

In the current literature of game theory, the concept of non-cooperative game has been applied to clustering in the last decade [36]. Basically it consists into a two-player game with complete knowledge over the game (*i.e.*, the similarity among the objects to be clustered), where each player has to select an object within the set of object to be clustered. The reward for this selection is given by the similarity between the objects selected by the two players (if they pick up the same object their reward is 0). This forces the players to select objects from the same cluster, increasing the internal coherence in the cluster definition, so that at the equilibrium both players may have the same cluster definition. Specifically, the game approach the clustering game according to a peeling-off strategy: by defining a cluster at a time, and re-running the game on the remaining un-clustered objects.

The cluster definition requires that both players reaches a same selection of objects at the equilibrium, leading to the so-called symmetric equilibrium; however it is possible to have some equilibria that are not symmetrical, leading to players to fail to learn in an unsupervised manner a common definition of an objects' cluster, and compromising the external coherence

of the clusters. In order to limit the arrival to only symmetrical Nash equilibria, evolutionary game can be exploited as in [6], which proved to bring clusters with high internal and external equilibria, which we have modified in order to have a continuous cluster definition. Specifically, instead of a couple of rational players (which select the strategies to maximize their payoff), we have a population of non-rational players, each instructed to select the same object, not the most convenient one. Strong strategies with high payoff will spread within the population through a learning, copying, inheriting or infection process. In fact, from the population a pair of individuals are extracted, and they receive a reward based on the similarity of the two selected objects, and from such a reward a proper function is used to compute the fitness of selecting such an object over other objects. The fitness of selecting the ith object is given by the different of the similarity among the cluster members without the object and the similarity with it. The population state represents the percentage of individuals in the population programmed to select a given object (*i.e.*, the ith value of the population state represents the ratio of individuals that pick the relative ith object to be clustered).

With respect to the approach in [6], in our solution not only the object to be clustered can be selected but also one of the clusters identified in the previous run of the game. A proper selection mechanism promotes the individuals with a high reward or fitness value in place of those with a lower one. Without going too much into the details, the frequencies of the object selection choices within the population change according to the payoffs, which in turn depend on the actions of the other individuals and hence, on the selection frequencies, yielding a feedback loop and leading to the most fitted selection choices to dominate over the others (which are destined to be erased from the population as they will have a null frequency in the vector state). The solution of the game is represented by a Evolutionary Stable State (ESS), in which the population cannot be invaded by any mutant strategy, and the population state is not affected by any changes. The output of the game is the population state at the ESS. Computing the ESS can be hard [37], and we have preferred to fix a timeout on the evolution of the game in order to do not have to pay the efforts for such a computation.

The input of the evolutionary clustering game is the similarity matrix A and the initial random population vector state x. Population evolution is performed according to given dynamics. The Replicator Dynamics (RD) is an example, falling within the class of the so-called imitation dynamics and being inspired by the Darwinian selection, and is expressed as follows:

$$x_i^{(t+1)} = x_i^{(t)}[f_i(x) - \phi(x)], \quad \text{where}$$
$$\phi(x) = \frac{1}{n}\sum_{j=1}^{n}(x_i f_i), \sum_i x_i^{(t)} = \sum_i x_i^{(t+1)} = 1 \tag{4}$$

where x_i is a real number indicating the ratio of individuals in the population selecting the ith item or cluster, $f_i(x)$ is the fitness value associated to the strategy of selecting the ith element (coming from the reward as expressed by the game rules) defined as $f_i(x) = (Ax)_i$ with A is the similarity matrix, and $\phi(x)$ is the average fitness in the population. According to the considerations in [6], other different classes of selection dynamics can be applied, such as the Infection and Immunization Dynamics (InfImmDyn), exhibiting a better asymptotic behavior compared to the RD [7]. However, we preferred to keep our approach as simple as possible as we are interesting in fast convergence for our online clustering. Therefore, we have applied RD until a ESS is reached, *i.e.*, when the Nash error of the current population state is below ϵ^2. Given the $g = Ax$ and $r = g - (x^T g)e$, the Nash error can be formulated as

$$\sum_{i \in S} min(x_i; -r_i) \tag{5}$$

Sometimes the convergence to ESS takes time, so our approach is stopped after 10 s even if the Nash error do not meet the previous condition. The output of such an interrupt game is used in the next run with the tweets collected so far, hoping that such additional input allows the game to converge. We have selected such a threshold of 10 s as in multiple runs of our approach, we noticed that in average the game converges within 10 s, and only in 5% of the cases the convergence takes more time.

4. Experimental results

In this section, we present the proof-of-concepts of our proposed approach and its consequent assessment, where some experimental results related to the efficiency and efficacy of the proposed approach are presented by considering the Twitter social data stream.

We have also empirically measured the latency of our approach, which in theory has a complexity estimated by the work in [6] to be $O(kn)$, if pure strategies are considered, or equal to $O(n^2 + kn)$ in case of mixed strategies, where k is the number of needed iterations.

The following subsections will respectively describe the realization of our prototype, the experimental campaign, and the obtained results.

4.1. Prototype implementation details

Our prototype has been designed according to the well-known *Kappa* architectural pattern [38], which is mainly focused on real-time analysis, for reducing latency time necessary to process the social data stream. The prototype is composed by three different layers: data ingestion, flow processing and data management (see Fig. 6).

The data ingestion layer relies on Apache Kafka,[2] a distributed streaming platform based on publish–subscribe pattern, to crawl in a real-time way tweets' flow gathered by the API Tweepy.[3] The crawled information is analyzed using the flow processing layer that firstly clean the social data stream on the basis of defined criteria and successively performs an online clustering. The obtained events are then stored into the NoSQL columnar database *Apache Cassandra*,[4] chosen for data management purposes due to its high performance in the writing phase. Finally, the obtained events can be shown to final users through any visualization facilities by querying the knowledge base.

4.2. Assessment approach

The evaluation process consists of three main activities:

- Firstly, we have compared performances of our event detection system implemented on the top of two different stream processor: Apache Kafka[5] and Apache Spark Streaming[6] in terms of tweets' latency and throughput;
- Then, we describe a running example for the proposed methodology, reporting some considerations about the adopted online clustering strategy.

2 https://kafka.apache.org/.
3 https://www.tweepy.org/.
4 http://cassandra.apache.org/.
5 https://kafka.apache.org/.
6 https://spark.apache.org/streaming/.

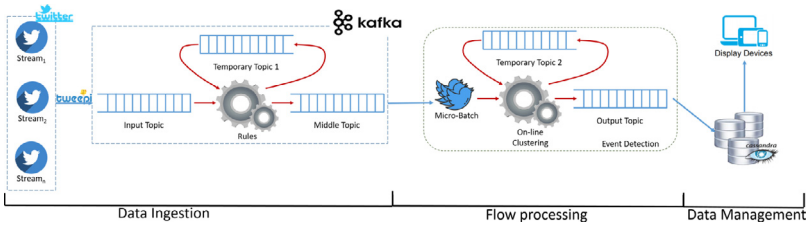


Fig. 6. Prototype overview.

Table 2
Parameters' configuration for stream processors.

N	Spark			Kafka				
	Spark setting	Input rate (Tw/s)	Windows time (min)	N	Input rate (Tw/s)	Number of consumer	Number of topics	Replication factor
1	Default	200	3	1	200	1	1	1
2	Default	200	1	2	650	1	1	1
3	Default	90	3	3	950	1	1	1
4	Spark executor memory = 4 Gb	200	3	4	1200	2	1	2
5	Spark driver memory = 8 Gb	200	3	5	1200	3	1	3
6	Spark driver memory = 6 Gb Master = 2 core	200	3	6	1200	2	2	1
7	Spark driver memory = 6 Gb Master = 2 core Spark Backpressure enabled	200	3					
8	Spark driver memory = 6 Gb Master = 2 core Spark memory fraction = 0.4	200	3					

- Finally, we compare the proposed approach with respect to the state-of-art methods with respect to some effectiveness measures, analyzing also the scalability of these approaches varying tweets' number.

For the first evaluation an artificial dataset has been produced varying the input rate (Tweet/second); whilst the second one has been performed by using a dataset of 4.200, 000 tweets collected for three days: every time it has been used, all the tweet dates have been replaced with that of the system at the time of use.

For the last kind of experiments we used two benchmark datasets [18] (i.e., *FA Cup* and *US election*). In particular, these datasets concern two main events happened in the 2012: *FA Cup* represents a time-ordered sequence of tweets – 124,524 tweets collected in six hour with 13 topics representing as keywords' set – published during the final match of the Football Association Challenge Cup; while *US Election* is composed by 2,335,105 tweets (with 64 topics) published during the United States presidential election of 2012.

Finally, concerning the experimental environment, the evaluation has been performed by using two Virtual Machine (VM) with 4 core and 16 GB RAM deployed on Microsoft Azure.[7]

For a fair comparison among the selected methods, we have run multiple experiments by varying the parameters of each approach, and we have picked the parameter configuration able to return the best performance. So, we have compared the algorithms by considering their best performance.

[7] https://azure.microsoft.com/.

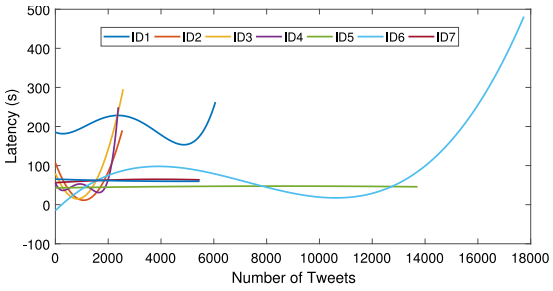
4.3. Stream processing performances

As already described, as first step we compared the efficiency of our system implemented on the top of two different stream processor (Apache Spark and Apache Kafka), varying the throughput rate from 200 to 1200 tweets for second. In particular, the two stream processor have been configured according to the parameters described in Table 2.

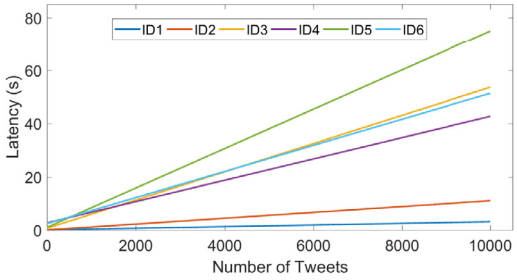
As easy to note in Fig. 8, the implementation based on Spark does not achieve a high throughput due to its high memory requirements (requested by the microbatch approach). In more details, the Spark execution using configurations from 6 to 8 was interrupted because the running time of the different batch processes exponentially increase (going from about 10/15 s to 6/7 min). In turn, Apache Kafka reaches 1.200 Tweet/s as maximum speed considering both producer and consumer on a single machine. In the case 1 the system guarantees a delay that corresponds to a maximum values of 1 s, whilst in the other cases the computational latency increases more or less linearly with the number of tweets.

In Fig. 7, we explicitly note that:

- In Spark, when the system cannot guarantee a low latency for small temporal windows, the tweets will be captured in the subsequent windows.
- In Kafka, the latency increases with the number of employed consumers, although the throughput of each consumer is reduced.



(a) Apache Spark Structured Streaming



(b) Apache Kafka

Fig. 7. Analysis about latency for tweets' computation varying input rate and framework configurations.

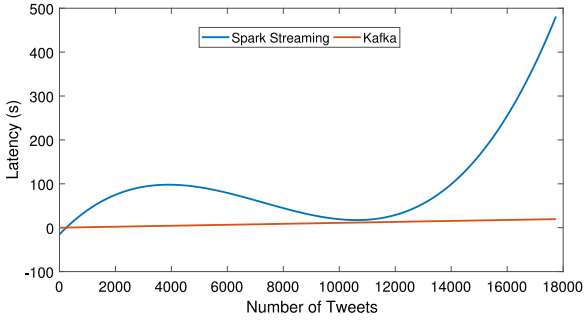
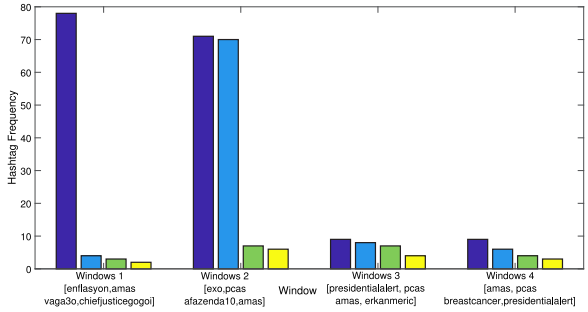
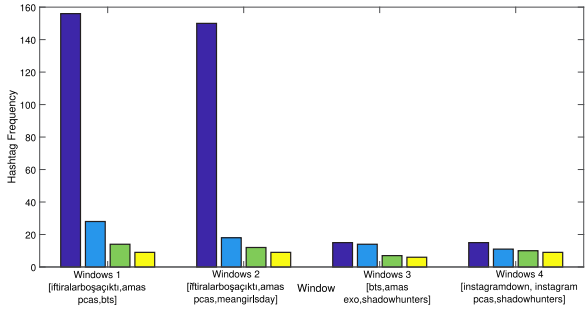


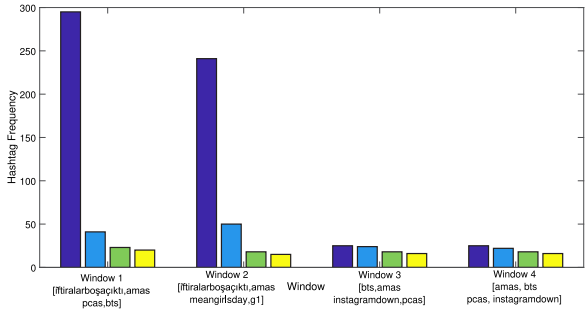
Fig. 8. Comparison between Apache Spark and Kafka about tweets' latency.



(a) Rule 1 with a 10-second time window.



(b) Rule 1 with a 1-minute time window.



(c) Rule 1 with a 3-minute time window.

Fig. 9. Fine tuning for Rule 1.

- In Kafka, the introduction of additional topic slows down the ingestion process.

From these observations it is clear that, in the cases where many data sources can be connected, it is advisable to increase the number of consumers, each listening to a different partition. If this solution fails to provide sufficient performances in order to meet the low latency requirement, it is better to increase hardware resources or add a further Kafka broker to the system.

In Fig. 8, the tweets' latency between Apache Kafka in the standard configuration and Apache Spark in the best configuration is shown. In addition, we note how the dataset has been used with a tweets' input rate of 200 Tweet/s, thus data reception from about 4 Tweepy Bees has been simulated.

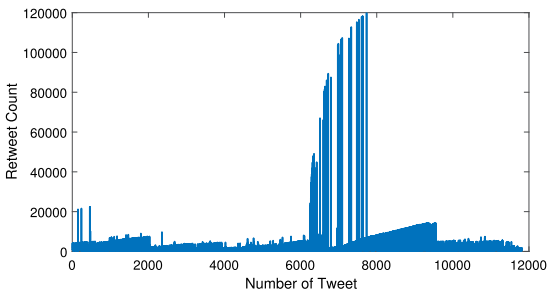
Thus, we can state that the two solutions require very similar times and we selected Apache Kafka as stream processor for the next experimentation.

4.4. Running example

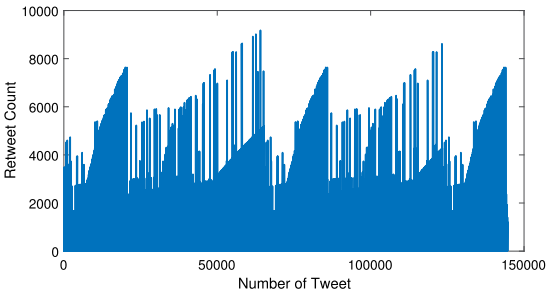
In this section we describe a case study for the proposed approach working for 3 days (7–9 October 2018) in which we collected 4, 2000, 000 tweets, where some considerations about system parameters setting can be done.

Firstly, we perform a fine tuning of parameters concerning the above described filtering rules. Fig. 9 shows occurrences number of 5 most frequent hashtags computed according to the Rule 1 varying windows size among 10 s, 1 and 3 min. It is easy to note that the first two time-windows represent the best case in which the maximum number of occurrences was obtained during the experiment. The windows 3 and 4 represent an average case, while the last two the worst case. Therefore, observing the graphs, it can be noticed that in windows 1 and 2 one or more outliers were always captured. This means that the rule is independent on the window size. A good window is that 1 min long because, in this way, there is a trade-off between the amount of information that is stored and the frequency with which the process which reset the values of the dictionaries is awakened. The threshold value in this case has been set to 50.

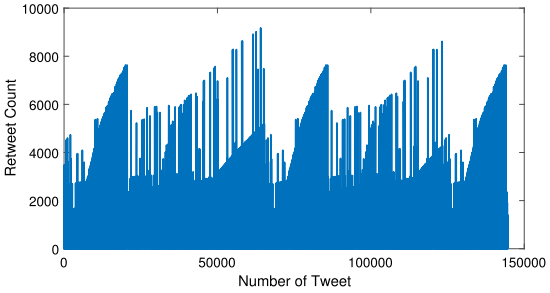
Concerning the Rule 2, it is possible to set the seniority of the retweet or cited tweet and the threshold values of the field that



(a) Rule 2 Retweet trend - 1 hour maximum retweet age.



(b) Rule 2 Retweet trend - 8 hours maximum retweet age.



(c) Rule 2 Retweet trend - 8 hours maximum quoted tweet age.

Fig. 10. Fine tuning for Rule 2.

concerns the quoted count and the retweet count. Two experiments were carried out (see Fig. 10): the first with a threshold of seniority of the tweet at 1 h and one at 8 h. Regarding the threshold values, they have been set voluntarily to low values in order to obtain significant trends.

Because the second rule allows us to capture tweets of interest, in order to guarantee to each tweet the necessary time for becoming an event of interest on the social network and then be captured, it was preferred to use an 8-hour window. In fact, among the results that will be successively presented, we can see how through this rule, tweets of public events were captured 2 h after publication.

Note that in this case, it is not of interest how many tweets are captured, but for the purposes of the rule it is important to know the trend that the retweet and quoted counts have. Then the thresholds have been set respectively to 3000 and 20,000 for quoted and retweet count.

Furthermore, concerning Rule 3, being this rule strongly dependent on the input list provided by the user, the only settings that can be made concern the difference between the number of followers and the number of friends of each user that published the tweets. The only possible choice is to ensure that the rule does not filter out influencers. By observing some influencers on Twitter, the threshold value has been experimentally set to 10,000.

Eventually, we have analyzed how the obtained parameters settings can affect the clustering effectiveness. Some experimental data are shown in Table 3, whose results were obtained adopting a round robin strategy for stream queue processing and we can note that:

- the increasing number of incoming tweet improves the performance of real time clustering;
- reducing threshold of algorithm improves real time and final clustering performances;
- the increasing number of incoming tweets, the use of small batch size and reducing algorithm threshold improves the performance of the final clustering;

Finally, after setting all the working parameters, the system has been left running for about 3 days. As regards tweepy API, two instances have been used with generic (vocal) keywords in order to capture generic events for any language. In addition, it has been set as a list of words to look for in the users' username like "news, event, etc"..

Analyzing the results, the following events were highlighted:

- Publication of an album by a Chinese singer;
- Publication of a new piece by the Stray Kids music group;
- The cries of a singer (Jimin) during a concert at the Citifield stadium;
- Donald Trump's tweets;
- Ronaldinho tweets;
- Tweets from a candidate in the Brazilian elections;
- "UFC229" martial arts event in America.

Among the extracted events, we have focused our attention on: the UFC229 event (produced by Rule 1) and the Ronaldinho tweet (produced by Rule 2).

About the UFC229 event, there are 10,412 tweets in Cassandra. The event began at 8 pm on 6 October, and considering the time zone, the event was discovered more than 30 min before the effective starting.

Regarding the Ronaldinho event, it is given by a set of 836 tweets captured by the system. However, the event of interest is not represented by the number of retweets, but only by Ronaldinho tweet. Considering the time zone, the tweet became

Table 3
Online Clustering Tuning.

	Batch Size	Threshold Rule 1	Word/ Tweet	Threshold algorithm	Cluster Representative Word	Cluster Number	Clusterized Tweets in Real Time	Real Time Silhouette	Final Silhouette
200 Tweet/s	10	5	3	0	Average	8	13k	0.43	0.32
	10	5	5	0	Average	3	18k	0.25	0.21
	20	5	3	0	Average	9	41k	0.34	0.31
	20	5	5	0	Average	7	33k	0.09	0.05
	50	5	5	0	Average	3	48k	0.31	0.03
	20	5	5	0.01	Average	3	50k	0.30	0.28
	20	5	5	0.05	Average	9	33k	0.14	0.11
60 Tweet/s	50	5	5	0	Average	6	40k	0.17	0.15
	20	10	3	0	All	9	44k	0.28	0.27
	20	10	3	0	Median	7	45k	0.37	0.37
	20	10	3	0.02	Median	9	45k	0.28	0.28
	20	10	5	0	Median	7	45k	0.31	0.31
	50	10	5	0.05	Median	9	29k	0.25	0.24

Table 4
Topic recall values comparing 5 state-of-art techniques with respect to the proposed one on FA Cup dataset varying number of retrieved keywords.

Method	Top-K									
	2	4	6	8	10	12	14	16	18	20
SFPM	.615	.840	.840	1	1	1	1	1	1	1
BNGRAM	**.769**	.920	.920	.920	.920	.920	.920	.920	.920	.920
Doc-P	**.769**	.850	.920	.920	1	1	1	1	1	1
EHG	.379	.591	.727	.727	.864	.864	1	1	1	1
Beats	.611	.611	.716	.716	.798	.798	.798	.798	.798	.798
Proposed	.712	**.944**	**.944**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Table 5
Topic recall values comparing 5 state-of-art techniques with respect to the proposed one on US election dataset varying number of retrieved keywords.

Method	Top-K										
	2	10	20	30	40	50	60	70	80	90	100
SFPM	.359	.359	.465	.525	.540	.540	.540	.540	.540	.540	.540
BNGRAM	**.480**	.480	.495	.495	.495	.495	.495	.495	.495	.495	.495
Doc-P	.234	.234	.415	.505	.560	.615	.615	.690	.690	.720	.740
EHG	.279	.608	.670	.688	.733	.746	.762	.772	.780	.796	.805
Beats	.321	**.670**	.714	.714	.714	.714	.714	.714	.714	.714	.714
Proposed	.299	.591	**.745**	**.745**	**.745**	**.745**	**.783**	**.783**	**.814**	**.814**	**.820**

of interest in the social and was discovered by the system about 2 h later.

4.5. Effectiveness evaluation

In this section, we evaluated the proposed method with respect to some of the state-of-art approaches. In particular, we selected: (i) SFPM [18], (ii) Doc-p [14], (iii) BEaTS [13], (iv) EHG [21].

Two measures are used for performing this evaluation: Topic-Recall@K (T-Rec), representing the percentage of ground truth topics detected correctly from top-k retrieved topics, and Keyword-Precision@K (K-Pre), measuring as the percentage of keywords detected correctly out of the top-K number of words. In particular, each ground-truth topic is composed by different keywords that are classified in mandatory, optional and forbidden. For computing T-Rec only mandatory keywords are considered whilst also optional ones are used for K-Prec.

Tables 4 and 5 show the obtained results in terms of T-Rec, where it is possible to see that increasing the number of retrieved topics our approach outperforms the state-of-the art approach. Indeed, our approach determines dynamically topics on the basis of most frequent top-k words in each cluster. Our approach shows better performances with respect to the state-of-the-art ones because clusters are dynamically updated, divided and/or merged during the batch analysis. More in details, clusters identified in the previous run of the game are added to the Game Theoretic Clustering for performing different strategies as well as clustering update, split and merge. The dynamic operations on clusters allows to better identify and manage burst in the tweets' sequence, as evident in Table 6, where we report the results for K-Prec.

Finally, we evaluated the scalability of the proposed approach with respect to *BEATS*[13] and *EHG,saeed2019enhanced*. As easy to note in Fig. 11, our approach outperforms the state of the art ones combining the online strategy with big data technologies.

5. Conclusions and future work

In order to cope with the massive amount of tweets/posts a given event can produce, users demands proper means to detect events and to get informed about them without having to go

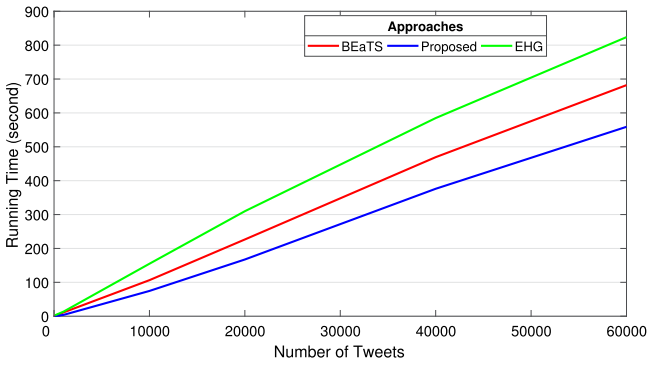


Fig. 11. Scalability evaluation of the proposed approach with respect to *BEATS* and *EHG* varying tweets' number.

Table 6
Keyword recall values comparing 5 state-of-art techniques with respect to the proposed one on FA Cup and US election datasets.

Method	Dataset	
	FA Cup	US Election
SFPM	.233	.241
BNGRAM	.299	.405
Doc-P	.337	.401
EHG	.442	.591
Beats	.650	.492
Proposed	**.676**	**.611**

through all the exchanged data and clean them from syntactical sugar. To this aim, this study has analyzed the problems and research challenges underlying on-line detection of arbitrary events within OSNs, and proposed a rule-based approach for filtering tweets and a game theoretical clustering to group tweets and extract key terms to be used for specifying events to be presented to the users. Such a solution has been implemented and assessed against the main competing related works showing an improvement in terms of achievable precision and accuracy. The presented solution has been designed and implemented by considering the features of the data within the tweets. However, its application is not limited to Tweeter only. It is easily extensible to any other possible OSN by implementing proper pre-processing rules so as to clean up data and remove noise able to compromise the consequent processing.

As a future work we plan to integrate a linguistic analysis of tweets in order to detect possible compromission of the tweets for fake news detection or truth analysis.

Furthermore, we will study how to assign different weights to the various terms that describe a cluster. To this aim, we can leverage proper linguistic resources (domain lexicon, thesauri, vocabularies, taxonomies, ontologies, etc.) that can be exploited to give more importance to terms characterizing in a better way a given domain (politics, sport, emergency, etc.).

CRediT authorship contribution statement

Rocco Di Girolamo: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Christian Esposito:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Vincenzo Moscato:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Giancarlo Sperlí:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

All authors approved the version of the manuscript to be published.

References

[1] R. Ureña, G. Kou, Y. Dong, F. Chiclana, E. Herrera-Viedma, A review on trust propagation and opinion dynamics in social networks and group decision making frameworks, Inform. Sci. 478 (2019) 461–475.
[2] N. Said, K. Ahmad, M. Riegler, K. Pogorelov, L. Hassan, N. Ahmad, N. Conci, Natural disasters detection in social media and satellite imagery: a survey, Multimedia Tools Appl. 78 (22) (2019) 31267–31302.
[3] J. Kleinberg, Bursty and hierarchical structure in streams, Data Min. Knowl. Discov. 7 (4) (2003) 373–397.
[4] J. Cotelo, F. Cruz, F. Enrquez, J. Troyano, Tweet categorization by combining content and structural knowledge, Inf. Fusion 31 (2016) 54–64.
[5] A. Tommasel, D. Godoy, A social-aware online short-text feature selection technique for social media, Inf. Fusion 40 (2018) 1–17.
[6] S.R. Bulò, M. Pelillo, A game-theoretic approach to hypergraph clustering, IEEE Trans. Pattern Anal. Mach. Intell. 35 (6) (2013) 1312–1327.
[7] S.R. Bulò, I.M. Bomze, Infection and immunization: A new class of evolutionary game dynamics, Games Econom. Behav. 71 (1) (2011) 193–211.
[8] F. Atefeh, W. Khreich, A survey of techniques for event detection in twitter, Comput. Intell. 31 (1) (2015) 132–164.
[9] A. Weiler, M. Grossniklaus, M.H. Scholl, Survey and experimental analysis of event detection techniques for twitter, Comput. J. 60 (3) (2017) 329–346.
[10] R.A. Alghamdi, A. Magdy, M.F. Mokbel, Towards a unified framework for event detection applications, in: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, 2019, pp. 210–213.
[11] İ. Arın, M.K. Erpam, Y. Saygın, I-TWEC: Interactive clustering tool for Twitter, Expert Syst. Appl. 96 (2018) 1–13.
[12] C.C. Aggarwal, K. Subbian, Event detection in social streams, in: Proceedings of the 2012 SIAM International Conference on Data Mining, SIAM, 2012, pp. 624–635.
[13] C. Comito, A. Forestiero, C. Pizzuti, Bursty event detection in Twitter streams, ACM Trans. Knowl. Discov. Data (TKDD) 13 (4) (2019) 1–28.
[14] S. Petrović, M. Osborne, V. Lavrenko, 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, Association for Computational Linguistics, 2010, pp. 181–189.
[15] N. Alsaedi, P. Burnap, O. Rana, Can we predict a riot? Disruptive event detection using Twitter, ACM Trans. Internet Technol. (TOIT) 17 (2) (2017) 1–26.
[16] E.S. Tellez, D. Moctezuma, S. Miranda-Jiménez, M. Graff, An automated text categorization framework based on hyperparameter optimization, Knowl.-Based Syst. 149 (2018) 110–123.
[17] H.-J. Choi, C.H. Park, Emerging topic detection in twitter stream based on high utility pattern mining, Expert Syst. Appl. 115 (2019) 27–36.
[18] L.M. Aiello, G. Petkos, C. Martin, D. Corney, S. Papadopoulos, R. Skraba, A. Göker, I. Kompatsiaris, A. Jaimes, Sensing trending topics in Twitter, IEEE Trans. Multimed. 15 (6) (2013) 1268–1282.
[19] Y. Li, H. Guo, Q. Zhang, M. Gu, J. Yang, Imbalanced text sentiment classification using universal and domain-specific knowledge, Knowl.-Based Syst. 160 (2018) 1–15.
[20] J. Singh, I. Kaur, A.K. Singh, Event detection from Twitter data, in: Proceedings of the 4th International Conference on Information Systems and Computer Networks, ISCON, IEEE, 2019, pp. 793–798.
[21] Z. Saeed, R.A. Abbasi, I. Razzak, O. Maqbool, A. Sadaf, G. Xu, Enhanced Heartbeat Graph for emerging event detection on Twitter using time series networks, Expert Syst. Appl. 136 (2019) 115–132.
[22] S. Dabiri, K. Heaslip, Developing a twitter-based traffic event detection model using deep learning architectures, Expert Syst. Appl. 118 (2019) 425–439.

[23] E. D'Andrea, P. Ducange, B. Lazzerini, F. Marcelloni, Real-time detection of traffic from twitter stream analysis, IEEE Trans. Intell. Transp. Syst. 16 (4) (2015) 2269–2283.
[24] C. Zhang, G. Zhou, Q. Yuan, H. Zhuang, Y. Zheng, L. Kaplan, S. Wang, J. Han, Geoburst: Real-time local event detection in geo-tagged tweet streams, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 513–522.
[25] C. Zhang, D. Lei, Q. Yuan, H. Zhuang, L. Kaplan, S. Wang, J. Han, Geoburst+: effective and real-time local event detection in geo-tagged tweet streams, ACM Trans. Intell. Syst. Technol. (TIST) 9 (3) (2018) 34.
[26] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, J. Han, Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 595–604.
[27] M. Khodabakhsh, M. Kahani, E. Bagheri, Z. Noorian, Detecting life events from twitter based on temporal semantic features, Knowl.-Based Syst. 148 (2018) 1–16.
[28] F. Amato, V. Moscato, A. Picariello, G. Sperlì, Extreme events management using multimedia social networks, Future Gener. Comput. Syst. 94 (2019) 444–452.
[29] Z. Yang, Q. Li, L. Wenyin, J. Lv, Shared multi-view data representation for multi-domain event detection, IEEE Trans. Pattern Anal. Mach. Intell. (2019).
[30] B. Poblete, J. Guzmán, J. Maldonado, F. Tobar, Robust detection of extreme events using twitter: worldwide earthquake monitoring, IEEE Trans. Multimed. 20 (10) (2018) 2551–2561.
[31] D.Y. Zhang, C. Zheng, D. Wang, D. Thain, X. Mu, G. Madey, C. Huang, Towards scalable and dynamic social sensing using A distributed computing framework, in: 2017 IEEE 37th International Conference on Distributed Computing Systems, ICDCS, 2017, pp. 966–976.
[32] C. Huang, D. Wang, Topic-aware social sensing with arbitrary source dependency graphs, in: 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN, 2016, pp. 1–12.
[33] G. Xun, Y. Li, J. Gao, A. Zhang, Collaboratively improving topic discovery and word embeddings by coordinating global and local contexts, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 535–543.
[34] Y. Li, H. Xiao, Z. Qin, C. Miao, L. Su, J. Gao, K. Ren, B. Ding, Towards differentially private truth discovery for crowd sensing systems, 2018, arXiv preprint arXiv:1810.04760.
[35] Y. Li, C. Miao, L. Su, J. Gao, Q. Li, B. Ding, Z. Qin, K. Ren, An efficient two-layer mechanism for privacy-preserving truth discovery, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1705–1714.
[36] M. Pelillo, S.R. Bulò, Clustering games, in: Registration and Recognition in Images and Videos, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 157–186.
[37] J. Li, G. Kendall, R. John, Computing Nash equilibria and evolutionarily stable states of evolutionary games, IEEE Trans. Evol. Comput. 20 (3) (2016) 460–469.
[38] N. Marz, J. Warren, Big Data: Principles And Best Practices of Scalable Realtime Data Systems, Manning Publications Co., 2015.

Rocco Di Girolamo is a Big Data Engineer at the Microlise company. He received his master's degree in computer engineering from the University of Naples "Federico II" in 2018. His research interests include real-time architectures for big data and the application of data mining and artificial intelligence techniques.

Christian Esposito is currently a Tenured Assistant Professor at the University of Salerno, and received the National Qualification in Italy as Associate Professor in Computer Engineering and Computer Science, respectively in May 2017, and July 2018. He was an Assistant Professor at the University of Napoli "Federico II", and a two-year Research Fellow and short-term Researcher at the Institute of High Performance Computing and Networking (ICAR) of the Italian National Research Council (CNR) from 2011 to 2015. He graduated in Computer Engineering in 2006 and got his PhD in 2009, both at the University of Naples "Federico II", in Italy. He has published about 100 papers at international journals and conferences, and has been a PC member or involved in the organization of about 60 international conferences/workshops. He regularly serves as a reviewer in journals and conferences in the field of distributed and dependable systems and is member of the editorial board of the International Journal of Computational Science and Engineering and the International Journal of High Performance Computing and Networking, both by Inderscience. He is Associate Editor of the IEEE Access, and has served as guest editor for various special issues at international journals. His interests include positioning systems, reliable and secure communications, game theory and multi-objective optimization.

Vincenzo Moscato is an Associate Professor at the Electrical Engineering and Information Technology Department of University of Naples "Federico II". He received the Ph.D. degree in Computer Science from the same University by defending the thesis: "Indexing Techniques for Image and Video Databases: an approach based on Animate Vision Paradigm". He is one of the leaders of PICUS (Pattern and Intelligence Computation for mUltimedia Systems) departmental research groups and a member of the Big Data and Artificial Intelligence national laboratories within the Consorzio Interuniversitario Nazionale per l'1Informatica (CINI). His research activities lay in the area of Multimedia, Big Data, Artificial Intelligence and Social Network Analysis. He was involved in many national and international research projects and coordinated as principal investigator. He was in the program committees of numerous international conferences and in the editorial boards of several important journals. Finally, he was an author of about 160 publications on international journals, conferences proceedings and book chapters.

Giancarlo Sperlí is a Research Fellow at the Department of Electrical and Computer Engineering of the University of Naples "Federico II". He obtained his PhD in Information Technology and Electrical Engineering at the same University defending his thesis: "Multimedia Social Networks". He is a member of the PICUS (Pattern and Intelligence Computation for mUltimedia Systems) departmental research group. His main research interests are in the area of Cybersecurity, Semantic Analysis of Multimedia Data and Social Networks Analysis. Finally, he has authored about 50 publications in international journals, conference proceedings and book chapters.