Event identification in web social media through named entity recognition and topic modeling

Konstantinos N. Vavliakis *, Andreas L. Symeonidis *, Pericles A. Mitkas

Aristotle University of Thessaloniki, Dept. of Electrical and Computer Engineering, GR54124 Thessaloniki, Greece
Information Technologies Institute, Centre for Research and Technology — Hellas, GR57001 Thessaloniki, Greece

Article history:
Received 14 December 2011
Received in revised form 26 August 2013
Accepted 27 August 2013
Available online 4 September 2013

Abstract

The problem of identifying important online or real life events from large textual document streams that are freely available on the World Wide Web is increasingly gaining popularity, given the flourishing of the social web. An event triggers discussion and comments on the WWW, especially in the blogosphere and in microblogging services. Consequently, one should be able to identify the involved entities, topics, time, and location of events through the analysis of information publicly available on the web, create semantically rich representations of events, and then use this information to provide interesting results, or summarize news to users.

In this paper, we define the concept of important event and propose an efficient methodology for performing event detection from large time-stamped web document streams. The methodology successfully integrates named entity recognition, dynamic topic map discovery, topic clustering, and peak detection techniques. In addition, we propose an efficient algorithm for detecting all important events from a document stream. We perform extensive evaluation of the proposed methodology and algorithm on a dataset of 7 million blogposts, as well as through an international social event detection challenge. The results provide evidence that our approach: a) accurately detects important events, b) creates semantically rich representations of the detected events, c) can be adequately parameterized to correspond to different social perceptions of the event concept, and d) is suitable for online event detection on very large datasets. The expected complexity of the online facet of the proposed algorithm is linear with respect to the number of documents in the data stream.

Keywords:
Event identification
Social media analysis
Topic maps
Peak detection
Topic clustering

1. Introduction

The World Wide Web has transcended from a read-only to a read-write web. Nowadays, anyone can actively participate in content creation by providing personal opinions, reporting an event, or commenting on a posted article. Information is uploaded in real-time; the minute an event that attracts people’s attention occurs, its description becomes available on some (micro) blogging service. Thus, we should be able to identify online and real-life events (at least the important ones), just by looking at the web.

The definition of event depends on context and granularity. Dictionary.com defines an event as “something that occurs in a certain place during a particular interval of time”. The term may also refer to a significant occurrence or happening, or a social gathering or activity [1]. WordNet defines an event as “something that happens at a given place and time” [2]. In knowledge representation, the event concept is an activity that involves an outcome or an arbitrary classification of a space/time region by a
cognitive agent. In general, one could argue that an event is “a notable activity that happens”. This definition is intentionally vague, since the event concept is socially defined, meaning that an event may be important to a group of people and unimportant to others.

In our work, events cover real life as well as web “happenings”, and may comprise only one (e.g. a natural disaster), or several topics (e.g. the Olympic Games and all the games that take place during them). Thus, our methodology can detect any major event that triggers enough discussion in the web. In the case of real life events, we detect their virtual representations, which we consider to provide an accurate description of the real life event. Although the web representation will appear after the real life event, the two event times are expected to be very close, as users tend to immediately report events in the social media, especially in Twitter [3]. Thus, in our work we use the real life event time and the time of the web event representation interchangeably, and we consider as future work the quantitative analysis of the difference between them.

To our understanding, an event is a “significant” action that has a duration (beginning–peak–end), involves a set of entities (legal or physical), and is associated with one or more locations. An event can be described by a topic, which we define as the matter (subject) dealt in the text that is used to identify the event. We consider that an event takes place when a sudden peak of the mentioned topic/entity occurs in the web.

Having defined the notion of event within the context of our work, we discuss the notion of event identification (or event detection\(^1\)). According to [4], event identification is the problem of identifying stories in several continuous news streams that amount to a new or previously unidentified event. Identification may imply discovering previously unidentified events in an accumulated collection (“retrospective identification”), or flagging new events from live news-feeds in an on-line fashion (“on-line identification”). Event identification comprises numerous challenges: one has to integrate information from multiple document streams, extract the spatial and temporal information associated with each document, identify and distinguish possible unknown entities, and classify the event to multiple event types.

The definition of importance is also subjective. Events that may be deemed interesting by many people are often available in various web sources. Although they are authored by different “reporters” that may use different vocabulary and express diverse opinions, they all share common features. Documents (articles, news stories, blogposts, comments, tweets, etc.) referring to the same event are reported at time periods close to the actual event. They also contain similar information (topics and named entities) that define the reporting event. We argue that these information snippets are the principal components that indicate events. We define an event as important, if the event has affected enough people to be reported or commented on in the Web. The minimum number of reports (per time unit) can be a tunable threshold and depends on the specific application/domain. In this work, we propose a methodology to detect previously unknown important events, as reported through social media interactions. We take advantage of publicly available information in the blogosphere and identify the time and space of events, as well as their semantics through a collective intelligence process. By time, we mean the time the event is reported on the web, which is after the event took place, but very close to the real event [3]. By space, we refer to the physical location(s) the event takes place and we try to identify, wherever available, the specific location entities involved. The event significance is calculated by the number of entity/topic occurrences over some time period. In addition, we propose an efficient unsupervised algorithm that detects the important events described in a dataset. The overall framework integrates a variety of unsupervised learning techniques, such as named entity recognition, dynamic topic map discovery, topic clustering, and peak detection, in order to identify events. Our methodology is suitable for fast detection of socially defined topics, both in an online, as well as in a retrospective manner. The contribution of our work can be summarized in the following: a) We propose a methodology that accurately detects interesting events and augments each event with semantic information pointing out the topic, the entities involved, the place, and the time period the event was observed on the web. b) We also propose an algorithm that can be adequately parameterized to accommodate different perceptions of the event concept and has expected complexity that grows linearly with the number of documents in the stream. This makes the algorithm suitable for online event detection on very large datasets. Our implementations of the proposed methodology and algorithm were evaluated on a dataset of 7 million blogposts and have outperformed other approaches in an international social event detection challenge.

The rest of the paper is organized as follows: related work on event and peak detection, as well as on topic extraction, is discussed in Section 2. Section 3 describes in detail all the facets of the proposed methodology, while Section 4 introduces the tunable algorithm for event detection. Section 5 evaluates the methodology through extensive experimental action and compares it against other approaches. Section 6 summarizes our work, discusses future directions, and concludes the paper.

2. Background and related work

The main objective of the event identification problem is to identify events from temporally-ordered streams of documents and organize these documents according to the events they describe. Towards this direction, numerous algorithms and techniques have been proposed, with most of them in the unsupervised learning category.

In the majority of cases, dynamic clustering techniques are used. One common approach is to model event identification as an online incremental clustering task [5]. For each document, its similarity to existing events (clusters of documents) is computed and the document is assigned to either an existing event, or to a new event based on predefined criteria. Following a different approach, Zhang et al. [6] propose a news event detection model that speeds up the detection task by using a dynamic news
indexing-tree. The tree uses term re-weighting based on previous story clusters, or statistics on training data, to learn a model for each class of stories. Allan et al. [7] propose single pass clustering and online adaptive filtering to handle evolving events within a stream of broadcast news stories, while Yang et al. [8] investigate text retrieval with hierarchical and non-hierarchical clustering to automatically detect novel events from a stream of news stories. Evolutionary theme patterns from text through exploitation of temporal text mining have also been investigated [9], in order to create an evolution graph of themes and analyze their life cycles.

Chen et al. [10] adopt a spatial approach, which transforms document streams into feature streams of tens of thousands of features. They propose a method that clusters bursty features to form bursty events and associate each event with a power value which reflects its burst level. As a result, the timely detection of bursty events that have occurred recently and the discovery of their evolutionary patterns along the timeline are possible. In [11], bursty events based on the feature distributions are detected by a parameter-free probabilistic approach, called feature-pivot clustering. This approach utilizes the time information to determine a set of bursty features which may occur in different time windows. Additionally, He et al. [12] present a topic independent method that dynamically represents documents over time and amplifies their features in proportion to their burstiness at any point in time. Recently, Petkos et al. [13] proposed the use of known clusters in the examined domain, in order to supervise the multimodal fusion and clustering procedure. By taking advantage of the explicit supervisory signal, they achieved superior clustering accuracy that required the specification of a small number of parameters. Furthermore, Zeppelzauer et al. [14] proposed the combination of spatio-temporal clustering with various filtering and refinement steps for event detection in Flickr datasets. First, they clustered the data based on the most reliable information (timestamps and geotags) to obtain robust event candidates and then employed additional contextual information for filtering and refinement of the event candidates.

Event identification has also been attempted through statistical methods. Ha-Thuc et al. [15] propose a scalable system that employs Latent Dirichlet Allocation in order to track events and sub-events, while excluding non-relevant (to the event of interest) text portions. On the other hand, Tang et al. [57] used latent temporal structures for complex event detection through spatio-temporal features of videos, appropriately quantized and aggregated into Bag-of-Words descriptors. Nikovshi and Jain [16] propose two fast algorithms with a computational complexity of $O(N^2)$, $N$ being the total number of documents, for the detection of abrupt changes in streaming data that can operate on arbitrary unknown data distributions before and after a hypothesized change. The first algorithm is based on computing the average Euclidean distance between all pairs of data points, while the second algorithm is based on computing the log-likelihood ratio statistic for the data, similar to the classic CUSUM algorithm. In [17], a TF-IDF model was presented for news event detection that includes generation of source-specific models, similarity score normalization based on document-specific averages, similarity score normalization based on source-pair specific averages, term re-weighting based on inverse event frequencies, and segmentation of the documents.

Chen and Chundi [18] extract hot spots of basic and complex topics, using temporal scan statistics to assign a discrepancy score to each of the intervals in the time period spanning the entire document set. Scan statistics are also used in [19], for taking the maximum of functions and comparing observed and expected numbers of events in a window of width $w$, in order to identify disease outbreaks. Similarly, Shi and Janeja [20] present an anomalous window discovery method through scan statistics for linear intersecting paths.

Probabilistic methods have also been widely used in the event detection task. Sakaki et al. [3] built a probabilistic spatiotemporal model based on Kalman and particle filtering in order to find the center and trajectory of targeted events residing in tweets. On the other hand, Zhang et al. [21] used probabilistic models and demonstrated that the cosine similarity metric and a redundancy measure based on a mixture of language models are both effective for identifying redundant documents. In [22], Li et al. proposed a probabilistic model that incorporates both content and time information for retrospective news event detection. Another interesting work is that of Traag et al. [59] who apply social event detection in massive mobile phone data using probabilistic location inference.

Classification algorithms are also extensively used in the event detection field. Yang et al. [23] use a supervised learning algorithm to classify the online document stream into predefined broad topic categories, and then perform topic-conditioned detection for documents in each topic. Following a different approach, Kumaran and Allan [24] use text classification and named entities, while exploring modifications to the document representation in a vector space-based system, in order to improve the performance of news event detection schemes.

Event detection has also been explored through graph analysis. Zhao and Mitra [25] define an event as a set of relations between social actors on a specific topic over a certain time period and represent the social text streams as multi-graphs, where each node represents a social actor and each edge represents a piece of text communication that connects two actors. Events are detected by combining text-based clustering, temporal segmentation and graph cuts of social networks. Sayyadi et al. [26] hypothesize that documents describing the same event contain similar sets of keywords and the graph of keywords for a document collection contains clusters of individual events. In this context, they built a network of keywords based on their co-occurrence in documents and proposed a graph-based event detection algorithm, that uses community detection methods to discover and describe events. More recently, Schinas et al. [27] used the Structural Clustering Algorithm for Networks (SCAN) [28] for detecting "communities" of documents. These candidate social events were further processed by splitting the events exceeding a predefined time range into shorter events.

Other advanced computer science techniques have also been employed. An infinite-state automaton, where bursts appear naturally as state transitions is used in [29] for identifying and analyzing the underlying content of bursty activities. Fuzzy theory is proposed in [18], where an efficient hot spot extraction algorithm that uses several efficient strategies to improve performance is proposed. A different approach is followed by Adaikkalavan and Chakravarthy [30], who propose SnoopIB, an event specification language and detect events based on interval-based semantics. On the other hand, Lee et al. [58] focus on Twitter for mining
microblogging text stream to obtain real-time and geospatial event information. A watershed-based method with support from external data sources was proposed by Dao et al. [31] to detect social events where metadata is transformed to an image so that each row contains all records of a user. These records are then sorted by time, transforming the social event detection task into a watershed-based image segmentation task. He et al. [32] consider the problem of analyzing word trajectories in both time and frequency domains, with the specific goal of identifying important and less-reported periodic and aperiodic words. In another interesting work, Weng and Lee [56], built signals for individual words by applying wavelet analysis on the frequency-based raw signals of the words for event detection in Twitter. Finally, Brenner and Izquierdo [33] propose the combination of various modalities from annotated photos as well as from external data sources within a framework that has a classification model at its core.

It is obvious that a significant amount of work has been done in this field. Nevertheless, none of the aforementioned efforts efficiently integrate named entity recognition and topic modeling with peak detection for discovering “important” events in large document streams. As discussed, the majority of approaches either cluster similar documents using text similarity metrics, thus they lack entity relevant information, or are based on spatio-temporal data and entity identification to extract events, thus they lack semantic information on the topic of interest. In any case, existing approaches do not consider the differences in the definition of event due to different social perceptions. Our approach is not only adequately parameterizable and takes into account the different social defined perceptions of “important” events, but can also define semantic representations of the detected events, meaning that they are described by relevant named entities and topics. This functionality is provided through time efficient algorithms, thus is suitable for retrospective, as well as online event detection in very large datasets.

3. Proposed methodology and requirements

Based on the above discussion, we define an event \( e_i \in E \) as a triple, where \( E \subseteq \mathcal{P}(NE \times \Theta \times T) \) a powerset of named entities \( NE \), topics \( \Theta \), and timestamps \( T \). Named entities comprise a powerset of persons \( P \), organizations \( O \), locations \( L \), and other entities \( OE \), thus \( NE \subseteq \mathcal{P}(P \times O \times L \times OE) \). A topic \( \theta_i \in \Theta \) (\( \theta \) from the Greek word “}\thetaμα\( “ for topic) is defined as a set of \( N_W \) words \( \theta_i = \{w_1, w_2, \ldots, w_{N_W}\} \) (\(|\theta_i| = N_W\), \( \forall i \)) while \( T = \{t_s, t_d, t_e\} \) is a sequence of the starting time \( t_s \), detection time \( t_d \), and end time \( t_e \) of the event. Events may have more than one active period (e.g. iterative events); in this case \( NE \) and \( \theta_i \) remain the same and \( T \) is updated with new timestamps. According to the above definition, each event has a semantically rich representation that provides information on the entities involved, the place and the time period that it took place, and the topic of the event.

Problem Statement: Given a document stream \( D = \{d_1, d_2, \ldots, d_N\} \), the problem is to detect a set of previously unknown, underlying events \( E = \{e_1, e_2, \ldots\} \). To do so, we propose a five step methodology, that is suitable for on-the-fly as well retrospective event detection and is depicted in Fig. 1. The five steps are:

1. Document stream preprocessing (Spam removal, html tag removal, stemming, etc.).
2. Entity and topic analysis, which includes named entity recognition (persons, organizations, locations, and other entities), topic discovery and topic clustering.
3. Evaluation of topic clusters using semantically-aware similarity metrics.
4. Event detection.
5. Event evaluation.

In the following subsections we elaborate on each step of the proposed methodology.

3.1. Preprocessing

In order to avoid the notorious “garbage in, garbage out” predicament, preprocessing is necessary to reduce noise, clean erroneous instances, and appropriately transform data. During preprocessing, HTML tags and stop-words are removed, in order to
create meaningful topic maps later. Moreover, stemming is required to reduce the inflected and derived words to their stem/root form and map related words to the same stem.

Further preprocessing actions may be necessary, depending on the format and nature of the original document stream. Spam documents have to be detected and removed. In any case, it is unlikely that spam posts refer only to certain entities. Even if this is the case, spam should be equally distributed over time. Since we are looking for entity–topic spikes in the web that denote events, spam documents have little effect on our methodology. Finally, in case of multilingual documents, one has also to take into account translation, mapping, and several other issues.

### 3.2. Entity and topic discovery

The second step of our methodology involves named entity recognition, topic discovery, and topic clustering. First, documents in the stream have to be scanned for named entities and then interesting topics have to be extracted for the top entities. As top entities we consider entities with enough occurrences, which may participate in various events with different contextual content, taking place at the same or different time periods. The minimum number of occurrences for an entity to be annotated as top entity is manually defined by the user and depends on the application. The final task of this step includes topic clustering, which has to take place in order to merge similar events. This step results into a set \( \text{NE} \) \( \{P \times O \times L \times \text{OE}\} \) of named entities and a set of \( K \) topic clusters, where each topic cluster is comprised of a set of \( N_\theta \) topics: \( \Theta = \{\theta_1, \theta_2, \ldots, \theta_{N_\theta}\} \). The number of topics \( N_\theta \) is a user defined variable, while \( K \leq N_\theta \) is the number of topic clusters. The selection of \( N_\theta \) and \( K \) is further discussed in Sections 3.2.2 and 3.2.3.

#### 3.2.1. Named entity recognition

Since entity names form the main contextual content of a document, it is important to identify them. Named entity recognition (NER) is the process of finding mentions of specified things in running text. It can be regarded as a classification task, where the goal is to detect and classify strings of text to different classes. Two are the dominant approaches followed in NER tasks; the first is a knowledge-based approach that uses explicit resources like hand-crafted rules and gazetteers, while the second is a dynamic approach, where a tagged corpus is used to train a supervised learning algorithm. We propose employing both methods, since their combination has shown to exhibit the best results in NER tasks [34].

#### 3.2.2. Topic discovery

The objective of this step is to discover sets of topics, as expressed by a stream of documents that identify their semantic content of those documents and express the semantic similarity among them. We employ topic modeling by the use of Latent Dirichlet Allocation (LDA) for extracting semantic information from document streams [35,36]. Topic modeling is based on the assumption that each document \( d_i \) can be described as a random mixture of topics and each topic as a focused multinomial distribution over terms. LDA builds a set of thematic topics from terms that tend to co-occur in a given set of documents. The result of the process is a set of \( N_\theta \) topics, each expressed with a set of \( K \) terms per topic \( N_w \) have to be defined in advance. The two parameters can be used to adjust the degree of specialization of the latent topics.

LDA discovers a mixture of topics \( P(\theta|d) \) for each document \( d_i \), where each topic is described as a mixture of terms \( P(w|\theta) \) following another probability distribution as given in Eq. (1). The probability of the \( w \) term describing a given document \( d \) is \( P(w_i|d) \), where \( \theta_i \) is the latent topic, and \( P(w_i|\theta_i = j) \) is the probability of \( w_i \) within topic \( j \). The probability of picking a term from topic \( j \) in a document is \( P(\theta_i = j|d) \).

\[
P(w_i \mid d) = \sum_{j=1}^{N_\theta} P(w_i \mid \theta_i = j)P(\theta_i = j \mid d)
\]

(1)

LDA estimates the topic–term distribution \( P(w|\theta) \) and the document–topic distribution \( P(\theta|d) \) from an unlabeled corpus of documents using Dirichlet priors for the distributions and a fixed number of topics. The Gibbs sampler [37] iterates multiple times over each term \( w_i \) in document \( d_i \) and samples a new topic \( j \) according to Eq. (2), until the LDA model parameters converge. In Eq. (2), \( C^w\theta \) maintains a count of all topic–term assignments, \( C^{\theta\theta} \) counts the document–topic assignments, \( W \) is the set of all available terms, \( \theta_\ldots \) are all topic–term and document–topic assignments except the current assignment \( \theta_i \) for term \( w_i \), and \( \alpha \) and \( \beta \) are the hyperparameters for the Dirichlet priors. Based on these counts, the posterior probabilities of Eq. (2) are estimated in Eqs. (3) and (4).

\[
P(\theta_i = j \mid w_i, d_i, \theta_{-i}) \propto \frac{C^w\theta_{w_i,j} + \beta}{\sum_{w} C^w\theta_{w,j} + W\beta} \times \frac{C^{\theta\theta}_{d,\theta_i,j} + \alpha}{\sum_{\theta} C^{\theta\theta}_{d,\theta,j} + \theta\alpha}
\]

(2)

\[
P(w_i \mid \theta_i = j) = \frac{C^w\theta_{w_i,j} + \beta}{\sum_w C^w\theta_{w,j} + W\beta}
\]

(3)
\[ P(\theta_i = j \mid d_i) = \frac{C_d^{\theta_j} + \alpha}{\sum_j C_d^{\theta_j} + \Theta \alpha} \]

The Gibbs sampler sets the complexity of topic modeling to \(O(N_o N I)\), where \(I\) is the number of the sampler’s iterations. In the context of our method, we build a different topic model for each of the \(N_o\) top entities, thus resulting in \(N_o\) topic models, each computed at maximum from \(N/N_o\) documents. LDA is generally considered a time and resource consuming process. Therefore, we propose offline building of new topic models at regular time periods and subsequent online assignment of topics to the document stream. Since we build different models for each top entity, this process can easily be distributed.

### 3.2.3. Topic clustering

Even though LDA is a powerful method for extracting topics from document streams, it requires a priori specification of the number of total topics \(N_o\). This constraint may result in reduced accuracy in the final detected events. Depending on the selection of \(N_o\) for entities with too few associated topics, LDA may be forced to produce exceedingly detailed topics, or even semantically overlapping topics (topics bearing the same meaning, expressed with different tags). On the other hand, limiting the number of topics may result in loss of vital information, as common entities may be associated with a higher number of topics than the one selected. Even if one chooses \(N_o\) based on the type and frequency of each entity, it is impossible to know a priori the number of topics to use. In addition, the same entity may participate in different number of events in different time periods.

We propose to start with a high number of topics. The exact starting value for \(N_o\) will depend on the application and/or on the quantity of available data (in our evaluation presented in Section 5 we set it to 50 topics per top entity) and then dynamically determine the optimum value by merging semantically similar topics, with the help of some clustering technique. This approach will minimize information loss for common entities associated with many topics, but it will allow extensive topic merging for low-profile entities. Next, we present two popular clustering techniques that can efficiently merge semantically relevant topics: a) graph and b) hierarchical clustering.

#### 3.2.3.1. Graph clustering

As the name suggests, topic clusters are represented as multi-graphs, where each node corresponds to a topic and each edge represents the similarity between two nodes. It has been shown that community-based semantics emerge from this graph representation and transformation process in similar applications [38]. A topic model \(TM\) can be regarded as a hypergraph (a bipartite graph, also known as a two-mode graph) with hyperedges. The set of vertices is partitioned into two disjoint sets:

1. **Topics**: \(\Theta = \{\theta_1, \theta_2, ..., \theta_{N_o}\}\), and
2. **Concepts (words-tags)**: \(C = \{w_1, w_2, ..., w_{WN}\}\)

Thus, the topic model \(TM\) is defined as \(TM \subseteq \Theta \times C\). The bipartite network is defined as \(H(TM) = (V, E)\), where \(V = \Theta \cup C\) is the set of vertices, and \(E = \{(\theta, w) | (\theta, w) \in TM\}\) is the set of edges. We can reduce the hypergraph into a one-mode network by folding it. This can be achieved if we denote the matrix of the hypergraph as \(H = [h_{ij}]\), where \(h_{ij} = 1\) if \(\theta_j\) is associated with concept \(w_i\). We define a new matrix \(S = [s_{ij}]\), where \(s_{ij} = \sum_k h_{ik}h_{kj}\) (or in matrix notation \(S = HH^t\)), showing the associations of topics weighted by the number of common concepts. The betweenness centrality score of an edge is defined as the extent to which that edge lies along shortest paths between all pairs of nodes, and is a good measure to find the edges between two communities. Inter-community edges will always obtain a high score, since the shortest paths between nodes from different communities will pass through them. Thus, by computing the edge betweenness for all edges in the graph and then removing edges with high betweenness scores, we can create clusters of topics that contain similar concepts. The selection of the number of edges \(k\) we remove, which determines the number of topics and the level of detail for each cluster, is determined in the topic cluster evaluation step discussed in Section 3.3.

#### 3.2.3.2. Hierarchical clustering

The generation of a graph comprising many topics is a resource-consuming procedure, that is not suitable for real-time event detection. As an alternative to edge-betweenness clustering, we can use single-link hierarchical clustering [39], which is much faster than graph clustering, and in many cases yields similar results. Single-link is based on the nearest neighbor concept, where the similarity of two clusters is the similarity of their most similar members.

To perform single-link text clustering it is necessary to define a representation model of textual data, a similarity measure among the documents, and a strategy for the cluster formation. The widely used space-vector model represents each document \(d_i\) as a vector of words, where each term is accompanied by its frequency of occurrence. Clusters are successively merged (in each step the two clusters whose two closest members have the smallest distance) until their similarity reaches a predefined partition distance \(p\). The selection for the value of \(p\) is discussed next.

### 3.3. Evaluation of topic clusters

The results of the topic clustering process can be assessed by the use of a semantic similarity evaluation metric which can progressively lead to the optimal number of clusters for each entity. Thus, the semantic metric should recommend an appropriate value for the number of edges \(k\) to remove (in case of graph clustering), or for the optimal value of partition distance \(p\) (in case of
hierarchical clustering). Semantic similarity is preferred over string-based matching because it will identify and, thus, cluster together semantically relevant concepts that consist of different strings. We propose using two similarity metrics: 1) the Normalized Google Distance — $\text{NGD}$ [40], or 2) the Wikipedia-based Semantic Similarity Metric.

Keywords with similar meanings tend to be close in “units” of Google distance, while words with dissimilar meanings tend to be farther apart. The $\text{NGD}$ between two search terms (topics in our case), $\theta_i$ and $\theta_j$ is presented in Eq. (5), where $\text{WP}$ is the total number of web pages searched by Google, $f(\theta_i)$ and $f(\theta_j)$ are the number of hits for topics $\theta_i$ and $\theta_j$ respectively, and $f(\theta_i,\theta_j)$ is the number of web pages that contain both $\theta_i$ and $\theta_j$. In practice, $\text{NGD}$ values belong to the range $[0-1]$, with 0 referring to complete semantic match.

$$\text{NGD}(\theta_i, \theta_j) = \frac{G(\theta_i, \theta_j) - \min \{G(\theta_i), G(\theta_j)\}}{\max \{G(\theta_i), G(\theta_j)\}} = \frac{\max \{\log f(\theta_i), \log f(\theta_j)\} - \log f(\theta_i, \theta_j)}{\log \text{WP} - \min \{\log f(\theta_i), \log f(\theta_j)\}}$$

Although $\text{NGD}$ is a reliable semantic metric, calculating the similarity of thousands of terms may turn out to be a time consuming process. As web search engines have limited throughput for computer generated queries, calculating the $\text{NGD}$ value for a large number of documents within an acceptable time period is not feasible. Another approach, followed by many researchers, computes the semantic similarity between literals using WordNet. Nevertheless, measuring semantic similarity with hand-crafted lexical resources like WordNet, which are not available for many languages and have limited coverage may be problematic, particularly for specialized domains. Thus, we propose using Wikipedia documents to compute the semantic similarity of terms and sets of terms. As with the $\text{NGD}$ concept, this approach also uses the distribution similarity as a proxy for semantic similarity, but without any limitations in the throughput of computer generated queries. Indeed, one can download a copy of Wikipedia and perform any queries without worrying about limitations and web-politeness. The calculation of Wikipedia based similarity is based on Eq. (5), except that in this case $\text{WP}$ refers to the total number of Wikipedia pages.

### 3.4. Detection of interesting events

Given that the notion of interestingness is subjective, the event detection algorithm should be tunable enough to identify a variety of different events and, in case of online event detection, time-efficient. We propose an algorithm with these characteristics, which is discussed in detail in Section 4.

### 3.5. Evaluation

The final step of the proposed methodology involves the evaluation of the whole process. Since a large number of events may be detected, they have to be ranked and evaluated against a ground truth event set. Alternatively, detected events can be evaluated according to user-based criteria. Evaluation results can be used to optimize the event detection algorithm. This optimization can be generic or personalized depending on whether evaluation is based on either a generally accepted ground truth or user preferences.

### 4. An efficient unsupervised event detection algorithm

In this Section, we propose an efficient unsupervised event detection algorithm that identifies all important events (as defined in Section 1) available in a document stream. Our algorithm searches for peaks of entity/topic combinations in streams and generates two sets of data. The first is the set of “potential events”, i.e. every combination of entity/topic that occurs at least once in a document. For each potential event, the algorithm counts the number of occurrences, calculates average occurrence and records the first and last occurrence timestamps within a specific temporal window of length $hV$. If the average occurrence exceeds an upper threshold, we regard the potential event as an identified event. To avoid continuously increasing the number of potential events, the algorithm removes the ones that do not occur frequently. As a result, the required memory space and running time of the algorithm are significantly improved.

The second dataset contains only the average number of encounters for every combination of entity/topic identified as a potential event. This is a relatively small set, which does not affect the time complexity of the algorithm and can be exploited to further reduce the number of potential events. Moreover, a global metric of the average number of occurrences for each entity/topic combination can be derived from this dataset and used as the baseline for the peak detection task.

Fig. 2 depicts a high level flowchart of the proposed event detection algorithm. First, we define the time interval to examine potential events. We define the concepts of frequent and infrequent potential events using a threshold value as discussed later in Section 4.1. The algorithm examines one by one all the documents in the stream and extracts the named entities and topics residing in them. Subsequently, for each document it calculates all the unique entity/topic combinations and for every combination, it keeps the number of occurrences and creates a new potential event, or updates an existing one. At regular time intervals, defined by $\text{examTimePeriod}$, our algorithm examines all potential events in order to decide which ones are too rare, so that they are no longer considered, and which ones are too frequent, so that they are identified as new events.
4.1. Analysis of the algorithm

The pseudocode of the proposed algorithm is listed in Algorithm 1. Our approach connects to the stream and analyzes one document at a time (line 5). First, it tracks the number of documents analyzed during the current exam time period (line 6) in order to calculate average values. It extracts the timestamp of the document at line 7 and identifies the named entities and topic of the document (line 8), as discussed in Section 3.2.2. It should be noted that both tasks can be performed offline and distributed in various machines.

For topic detection, LDA can be used as discussed in Section 3.2.2. LDA calculates for each document the probability to belong to each topic but here we only consider the topic with the highest probability. LDA is a time-consuming process, thus this approach is only practical in the case of retrospective event detection.

For online event detection, a faster approach is necessary. To increase efficiency, one can easily extract the topic of the document by examining its relevance to a set of pre-discovered topics, using one of the semantic similarity metrics discussed in Section 3.3. These pre-discovered topics are topic models that are computed at regular time intervals from scratch using the documents available on that particular time interval. We consider as typical interval a value between one hour and one day, depending on the size of the dataset, the available processing power, and the level of detail we want to achieve in the identified events.

Next, all available combinations of the $|NE(d_i)| + |\Theta(d_i)|$ entities/topics of document $d_i$ are calculated (line 9), where the order of the elements is not important and repetitions are discarded. The combinations are stored in a list named $PET$. The number of combinations is given by:

$$
\sum_{x=1}^{\binom{|NE(d_i)| + |\Theta(d_i)|}{x}} \frac{(|NE(d_i)| + |\Theta(d_i)|)!}{x!(|NE(d_i)| + |\Theta(d_i)|−x)!}
$$

Fig. 2. Flowchart of the event detection algorithm.
To improve the time efficiency of the method for documents that mention too many entities (e.g., historic articles) and after enough experimentation we have empirically set the maximum number of entities allowed in a single blogpost to $|NE(d)|_{\text{max}} = 5$, as we consider that documents with $|NE(d)| > 6$ do not focus on any particular entity. We also set $|\Theta(d)| = 1$, as we consider that each document is associated with one topic. Thus, the maximum number of combinations according to Eq. (6) is limited to 63. Then, the algorithm goes through every entity/topic combination (line 10) and updates the respective potential event (lines 11 and 12) or creates a new one (line 14), in case one does not already exist, and adds it in the PEV list of potential events. Since a single document contains multiple combinations of entities/topics, a document can participate in multiple events. Likewise, the algorithm updates an object (line 18) or creates a new one (line 20), that holds the global average occurrences of the examined permutation.

At regular time intervals, defined by the $\text{timeUnit}$ parameter (line 23), the list of potential events is traversed (line 24) and potential events that exhibit increased activity are retained, while potential events with minimal activity are removed from the list. To decide which events to identify and which to discard, the following process is implemented: the occurrence frequency of a potential event during the previous examination time period is divided by the number of documents examined in this time period (line 25) to define the relative average number of occurrences, $r_{cj,t}$. Calculating $r_{cj,t}$ is necessary in order to a) normalize the potential event counter, b) avoid missing events, or c) create false-positive recognitions due to variations in the total number of documents in each investigated time unit. Such variations may be due to differences in the collection rates.

Two critical steps in the algorithm are the removal of “weak” potential events (lines 26–27) and the detection of “important” potential events as new events (lines 28–29). For these operations, two upper and two lower thresholds are defined. A new event is detected when the average times encountered, $\text{avgRelEnc}(hV)$, within the $hV$ interval is higher than a minimum detection threshold $\text{detectionAvg}$ (to avoid detecting very weak events) and sufficiently higher than the average times $\text{AVGInfo}(PET_j)$ these entities/events occur in general. The sufficiently higher concept is expressed as the product of a factor $\text{upLimit} > 1$ with the $\text{AVGInfo}(PET_j)$. Similarly, a potential event is removed from the list of potential events, when $\text{avgRelEnc}(hV)$ is lower than a “destruction” threshold $\text{destructionAvg}$, or sufficiently lower than the $\text{AVGInfo}(PET_j)$. Again, the sufficiently lower concept is expressed with a factor $\text{lowLimit} < 1$ that is multiplied with $\text{AVGInfo}(PET_j)$. Finally, the algorithm increases the time unit counter (line 31) and nullifies the document per hour counter (line 32), in order to begin processing for a new time slot.

**Algorithm 1.**

**Algorithm 1** Pseudocode of the event detection algorithm.

```plaintext
Require: stream of $N$ documents: $D = (d_1, d_2, ..., d_N)$
        low threshold under which potential events are destroyed: $\text{detectionAvg}$
        upper threshold for detecting a new event: $\text{DetectionAvg}$
        factors that affect the detection/destruction of a potential event: $\text{upLimit}/\text{lowerLimit}$
        number of time periods to take into account when calculating the average number of encounters with $\text{avgRelEnc}()$: $hV$

1: i = 0
2: $\text{examTimePeriod} = \text{setExamTimePeriod}()$ //User defined value, expected between 1-12 hours
3: $\text{docCounterPerExamTimePeriod} = 0$; //Counter of total documents encountered per examTimePeriod (for normalization purposes)
4: $\text{AVGInfo} = \text{PEV} = \text{DEV} = 0$; //Sets of global average information, potential events (PEV), and detected events (DEV)
5: while $d_i \neq \text{null}$ do
6:   $\text{docCounterPerExamTimePeriod} + 1$;
7:   $t_i = d_i.\text{timestamp}$;
8:   $\text{entitiesTopics} = \text{extractNamedEntitiesAndTopics}(d_i)$; //Find all named entities and the topic of document $d_i$
9:   $\text{PET} = \text{getCombinationsOfEntitiesTopics(entitiesTopics)}$; //Get all combinations of entities/topics
10: for $j = 1$ to length($\text{PET}$) do
11:   if $\text{PET}_j \notin \text{PEV}$ then
12:     $\text{PET}_j.\text{lastEncountered} = t_i$; //Update the last encounter date of event $j$
13:     $\text{PET}_j.\text{timesEncountered} + 1$; //Increase number encountered for event $j$
14: else
15:     $\text{PEV} = \text{PEV} \cup \text{PET}_j$; //Add entity/topic combination $j$ as a new potential event
16: end if
17: if $\text{AVGInfo}.\text{containsInfoFor}($PET$)_j$ then
18:   $\text{AVGInfo}.\text{PET}_j.\text{timesEncountered} + 1$; //AVGInfo contains global information of all entity/topic combinations
19: else
20:   $\text{AVGInfo}.\text{addInfoFor}($PET$)_j$; //Add new information for $\text{PET}_j$
21: end if
22: end for
23: if $t_i > \text{examTimePeriod}$ then
24:   for $j = 1$ to length($\text{PET}$) do
25:     $r_{cj,t} = \text{PET}_j.\text{timesEncountered}/\text{docCounterPerExamTimePeriod}$; //Calculate the average number of encounters for potential event $\text{PET}_j$
26:   if $\text{PET}_j.\text{avgRelEnc}(hV) < \text{detectionAvg}$ or $\text{PET}_j.\text{avgRelEnc}(hV) < \text{lowerLimit} \times \text{AVGInfo}($PET$)_j$ then
27:     $\text{PEV} = \text{PEV} \cup \text{PET}_j$; //Destroy potential event
28:   else if $\text{PET}_j.\text{avgRelEnc}(hV) > \text{DetectionAvg}$ and $\text{PET}_j.\text{avgRelEnc}(hV) > \text{upLimit} \times \text{AVGInfo}($PET$)_j$ then
29:     $\text{DEV} = \text{DEV} \cup \text{PET}_j$; //Detect new event
30: end if
31: $\text{PET}_j.\text{timeUnits} + 1$; //Add one time unit to the potential event
32: $\text{PET}_j.\text{timesEncountered} = 0$;
33: end for
34: $\text{examTimePeriod} = \text{examTimePeriod} + t_i$; //New exam time period begins where the last one ended
35: $\text{docCounterPerExamTimePeriod} = 0$; //Set the counter of total documents encountered per exam time period equal to zero
36: $\text{AVGInfo} = \text{removeRareArgInfo}(\text{AVGInfo})$; //Remove information for too rare entity/topic combinations
37: end if
38: $t + 1$;
39: end while
```
4.2. Time complexity of the proposed method

The complexity of the proposed algorithm is an aggregation of the complexities of its five main tasks: A) named entity recognition, B) construction of the topic models, C) topic clustering, D) topic clusters evaluation, and E) event detection. Tasks B, C and D can be performed in offline mode (and easily distributed to different machines), while Tasks A and E can either be executed in offline mode for retrospective event detection, or in online mode for real time event detection.

The named entity recognition task is performed on a per document basis. Since the execution time for a single document depends on its length, which can be regarded as constant equal to the average document length, Task A’s complexity for N documents is \( O(N) \). The expected complexity of Task B depends on the complexity of Gibbs sampler, which is \( O\left(\frac{N}{N_T} N_p I\right) \), where \( N \) is the number of documents, \( N_T \) the number of top entities for which we perform LDA analysis, \( N_p \) the number of topics, and \( I \) the number of iterations. As discussed earlier, we perform topic modeling analysis on a per entity basis, only for the top \( N_T \) entities. Thus, we can assume that it is sufficient to limit the number of topics: \( N_T \in (0,100) \) for each top entity, while \( I \in (500,1000) \) usually applies [36,42]. Since \( N_T, N_p \) and \( I \) are constants, the complexity of this Task for \( N \) documents is \( O(N) \).

The complexity of Task C naturally depends on the clustering method employed. The complexity for graph clustering based on betweenness centrality is \( O(k m N_\theta) \), where \( k \) is the number of edges to remove, \( m \) is the total number of edges, and \( N_\theta \) is the total number of vertices (topics). Since the number of topics \( N_\theta \) is the maximum number of total edges, and consequently the maximum number of edges to remove, are constant \( (N_{\theta\text{MAX}}^2 = 10,000 \text{ in our case}) \), the complexity of the betweenness clusterers for the \( N_T \) top entities depends only on the number of topic models to be built and is independent of \( N \). Likewise, the complexity for the single-link hierarchical clustering is \( O(N_\theta^2) \) [43] and independent of \( N \). As far as Task D is concerned, the complexity for evaluating the topic clusters is \( O(N_\theta^2) \), since in the worst case every topic has to be compared against every other topic, and is independent of \( N \).

The operation that defines the actual online time complexity of the proposed method is the event detection algorithm. Task E cannot be easily distributed and must be executed in real time for online event detection. The main block that defines the complexity of Algorithms 1 is the while loop in lines 5–39. Since all available documents are parsed one by one (line 5), \( N \) iterations are needed. The sub-block in lines 23–37 is executed for all potential events at regular time periods defined by examTimePeriod. In the worst case, the number of potential events is in the order of \( N \) times the available combinations of entities/topics (limited to 63 according to Eq. (6)), and examTimePeriod is in the order of the average time interval between two documents. Thus, in the worst case, the sub-block between lines 5–39 will be executed for all \( N \) documents and the length of PEV would be \( N \), so the overall time complexity of the algorithm would be:

\[
O\left(\frac{N^2}{\text{number of documents within an examTimePeriod}}\right) = O(N^2).
\] (7)

In practice though, the expected running time of the algorithm can be reduced to \( O(N) \), as we explain next. Let us assume that we set examTimePeriod to an appropriate value in order for the number of documents within it to be in the order of \( \sqrt{N} \). This means that we check and update the list of potential events every 3162, 10,000 or 31,620 document when the stream consists of 10, 100 or 1000 millions of documents respectively, which is more than acceptable for the needs of our algorithm. We also assume that the number of potential events is significantly lower than the number of total documents by selecting a proper value for upLimit that limits the total number of detected events in the order of \( \sqrt{N} \). This is also acceptable, since in all our experiments in Section 5 the maximum number of potential events has been identified to be much lower than \( \sqrt{N} \) for all the different parameter values tested (plus we can always manually limit them). All things considered, it is obvious that the expected complexity of the online part of the detection algorithm depends on the number of \( N \) iterations (line 5) times the number of iterations of sub-block between lines 23–37, which for \( N \) documents is:

\[
O\left(\frac{N}{\text{number of potential events}}\right) = O\left(\frac{N^{\sqrt{N}}}{\sqrt{N}}\right) = O(N)
\] (8)

5. Experimental evaluation

In this section, we provide some implementation details and evaluate the topic modeling and topic clustering steps, as well as we discuss the process of selecting the appropriate topic clusters using a large dataset of blogposts. Analysis and evaluation are performed with benchmark data provided by the ICWSM 2009 Data Challenge. In addition, we perform multivariate test analysis and analyze the accuracy, total number of detected events, average event duration, and execution time of detected events. Finally, we compare our methodology with other approaches in the context of an international event detection challenge (MediaEval2012).
5.1. Implementation details

In order to implement the various steps of the proposed methodology (Fig. 1), a number of tools were employed. In the preprocessing phase, the Porter Stemmer [44] was applied, followed by a simple spam detection method, which was developed to remove very short messages and messages with many outgoing links. The LingPipe [45] linguistic framework was used for building the named entity recognition model, while we also employed supervised training for building statistical models that use dictionary matching and regular expression matching for improving the entity recognition process. In addition, we built a complete lexicon of organizations and locations, from publicly available open linked data, taken from DBPedia [46] and Geonames.org (http://www.geonames.org). The lexicon is used for reducing the false positive results of the entity recognition model.

For grouping similar topic-maps into clusters, we employed the graph clustering method presented in [47] that is based on calculating the betweenness values of edges. The running time of this algorithm is $O(k m N^{\theta})$, where $k$ is the number of edges to remove, $m$ is the total number of edges, and $N$ is the total number of vertices. For very sparse graphs, the running time is closer to $O(k N^{2 \theta})$, while for graphs with strong community structure the complexity is even lower.

To calculate the Wikipedia-based semantic similarity, we followed the approach proposed by Kolb [48]. First, we used a simple context window of three words for counting co-occurrences. By moving the window over a Wikipedia corpus consisting of 220,000 words (resulting into 267 million tokens, obtained from http://www.linguatools.de/disco/disco-download_en.html), we generate a co-occurrence matrix. In order to find a word’s distributionally similar words, one should compare every word vector with all other word vectors. For vector comparison, Lin’s information theoretic measure [49] was employed. In order to compute the overall matching score between two topics, we used the matching average method (Eq. (9)), which calculates $WSim$, the similarity between two topics $\theta_i$ and $\theta_j$ by dividing the sum of similarity values of all match candidates of both sets by the total number of set tokens:

$$WSim = \frac{2 \cdot \text{Match}(\theta_i, \theta_j)}{|\theta_i| + |\theta_j|}.$$  

5.2. Evaluation of proposed methodology and algorithm

The 3rd International AAAI Conference on Weblog and Social Media hosted the ICWSM 2009 Data Challenge. The Challenge was based on a dataset [50] consisting of 62 millions of blogposts (compressed corpus size was 45GB), as collected, processed and delivered in XML format by the Spinn3r (http://www.spinn3r.com) company. The corpus consisted of weblogs published from August 1, 2008 to October 1, 2008. This dataset spans over a number of big news events, like the 2008 Athens Olympics, both US presidential nominating conventions, the initial stages of the financial crisis, as well as everything else one might expect to find posted to blogs. Thus, this dataset is an excellent test-bed for testing and evaluating the proposed methodology.

5.2.1. Data preprocessing

In the preprocessing phase we selected the English posts created in August 2008, in order to have a better overview of the events that took place during that period. This action resulted in a dataset of 12 million blogposts. The number of documents in per day- and per hour-basis are depicted in Fig. 3, where all dates use Greenwich Mean Time. Different collection rates, driven by differences in network throughput and availability, processing resources, or external variables may lead to differences in the available data. This is the main reason for using the normalization variable $docCounterPerExamTimePeriod$, as described in Algorithm 7. In addition, we compiled two gazetteers, one comprising a list of organizations and one comprising a list of locations. For compiling these gazetteers, we retrieved the appropriate entities available in DBPedia.org and Geonames.org, yielding a list of 257,373 organizations and 6,929,290 location names, respectively.

![Fig. 3. Total number of documents on a a) per day and b) per hour basis.](image)
Having created the topic models, our next task is to merge similar topics through clustering based on the edge-betweenness degree and decide on the number of clusters. Fig. 4 illustrates an example of the process, where four graphs of different clustering iterations [42].

Having created the topic models, our next task is to merge similar topics through clustering based on the edge-betweenness degree and decide on the number of clusters. Fig. 4 illustrates an example of the process, where four graphs of different clustering iterations [42].

5.3. Entity and topic discovery — cluster identification and evaluation

5.3.1. Entity and topic discovery

Before applying any stemming or stop-words removal, we identified the entities on the original blogpost context, since NER techniques use prepositions and other grammatical types to discover entities. Table 1 shows the distribution of entities into the three main classes of Persons, Organizations and Locations. Despite the fact that the number of distinct organizations and locations is significantly lower than the number of distinct persons, the use of the gazetteers resulted to comparable numbers of occurrences (i.e. limited number of invalid/faulty entities discovered for these categories).

5.3.1.1. Preparation of the ground truth event set. In order to evaluate the accuracy of our methodology, we manually generated a set of ground truth events that took place during the period we examined. Trying to keep the ground truth set as objective as possible, we used the list of August 2008 events reported by Wikipedia (http://en.wikipedia.org/wiki/August_2008) and compiled a list of 199 events. A small sample of that list is depicted in Table 2.

Next, we created the topic models for a set of “top entities”, using the Latent Dirichlet Analysis with the Gibbs Sampler. Taking into account the size of our dataset, we regarded as top entities the ones with more than 1000 occurrences. We used this threshold for two reasons: a) entities with fewer occurrences are regarded as unlikely to participate in multiple topics and b) it is inefficient to create millions of different topic models. This process resulted into 240 entities, with the last one having 1000 occurrences. The five most common entities along with one example topic are available in Table 3.

Based on these entities, we built the respective topic models. Our intention was to create a relatively high number of topics and then merge the ones that were similar. For this reason, we set the number of topics to $N_0 = 50$ for each model. For building the topic models we used typical values found in literature, such as: $\alpha_0 = 50/N_0 = 1$, $\beta_0 = 0.1$, and $I = 1000$ Gibbs Sampler’s iterations [42].

5.3.2. Clustering and cluster evaluation

Having created the topic models, our next task is to merge similar topics through clustering based on the edge-betweenness degree and decide on the number of clusters. Fig. 4 illustrates an example of the process, where four graphs of different clustering levels are depicted, concerning President Obama. Clusters are shown as tagclouds, which provide a versatile textual and visual representation. In the upper left quadrant of Fig. 4, all topics form one cluster with no edges being removed by the clustering algorithm. Thus, there is only one general topic describing the nomination campaign of President Obama. As more and more edges are removed in subsequent steps of the algorithm, more clusters are created, thus more specific topics emerge. The general topic describing the campaign of President Obama remains in all four subfigures, but, as we move into higher levels of detail, topics such as the web campaign that took place during the nomination period, an interview with Pastor Rick Warren, an arrest that took place at that period, etc. begin to emerge.

A key point in our methodology is the selection of the optimal number of topic clusters using two semantic similarity metrics, the NCD and the Wikipedia based similarity. Thus, we created various sets consisting of different numbers of clusters and then evaluated each cluster set. After enough experimentation, we concluded that cluster sets having average Wikipedia-based similarity WSim near to 0.1 presented the most meaningful topics. This is a subjective criterion used by the authors; others may

<table>
<thead>
<tr>
<th>Date</th>
<th>Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8/08</td>
<td>Balcilar, Konya, Turkey</td>
<td>11 people die due to explosion in Balcilar in Konya province in Turkey</td>
</tr>
<tr>
<td>2/8/08</td>
<td>Antonio Pettigrew</td>
<td>U.S. 4 × 400 team is stripped of gold medal after Antonio Pettigrew admits to doping</td>
</tr>
<tr>
<td>3/8/08</td>
<td>Himachal Pradesh, India</td>
<td>162 people die in a stampede in Naina Devi in the Indian state of Himachal Pradesh</td>
</tr>
<tr>
<td>4/8/08</td>
<td>Robert Novak</td>
<td>U.S. political commentator Robert Novak, involved in the CIA leak scandal, retires</td>
</tr>
<tr>
<td>5/8/08</td>
<td>Sichuan, China</td>
<td>6.0 Richter earthquake strikes Sichuan, China, according to the US Geological Survey</td>
</tr>
<tr>
<td>6/8/08</td>
<td>Salim Hamdan</td>
<td>Salim Hamdan, former driver of Osama Bin Laden, convicted of supporting terrorism</td>
</tr>
<tr>
<td>7/8/08</td>
<td>Andrew Cuomo, Citigroup</td>
<td>Andrew Cuomo reaches $78 settlement with Citigroup to buy auction rate securities</td>
</tr>
</tbody>
</table>
select some higher value of $WSim$ to create more general topics, while others may prefer more specialized topics by selecting a lower value of $WSim$.

Fig. 5 shows the results of clustering based on edge-betweenness as was applied to the topic model built on President Obama. Fig. 5a depicts the number of removed links with respect to the different number of clusters created. Fig. 5b depicts the average semantic distance for each clustering level, using the NGD and the Wikipedia based metric, while Fig. 5c plots the average cluster cohesion and cluster separation for different clustering levels. Cluster cohesion is defined as the number of links between all topics of the same cluster, while cluster separation is defined as the number of links between all topics of different clusters [51]. Similarly, Fig. 6 plots the partition distance $p$, the average semantic similarity, the cluster cohesion, and the cluster separation for different cluster numbers. The depicted example results from applying the agglomerative hierarchical clusterer, with different values of the partition distance.

In the case of hierarchical clustering, changing the value of partition distance causes sharp changes in the number of clusters, while in graph clustering, changing the number of edges to remove causes smooth changes in the number of topic clusters. This

<table>
<thead>
<tr>
<th>Entity</th>
<th>Occurrences</th>
<th>One example topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>224,953</td>
<td>Price market dollar trade economy stock growth inflation oil global high currency</td>
</tr>
<tr>
<td>John McCain</td>
<td>61,950</td>
<td>Tax cut plan pay raise spend economy year propose income secure money rate budget</td>
</tr>
<tr>
<td>Barack Obama</td>
<td>58,199</td>
<td>Convention speech Hillary party john republican nation Denver speak campaign former</td>
</tr>
<tr>
<td>George Bush</td>
<td>25,847</td>
<td>Kill Iraq Baghdad bomb force attack suspect soldier office suicide explosion terrorist</td>
</tr>
</tbody>
</table>

select some higher value of $WSim$ to create more general topics, while others may prefer more specialized topics by selecting a lower value of $WSim$.

Fig. 5 shows the results of clustering based on edge-betweenness as was applied to the topic model built on President Obama. Fig. 5a depicts the number of removed links with respect to the different number of clusters created. Fig. 5b depicts the average semantic distance for each clustering level, using the NGD and the Wikipedia based metric, while Fig. 5c plots the average cluster cohesion and cluster separation for different clustering levels. Cluster cohesion is defined as the number of links between all topics of the same cluster, while cluster separation is defined as the number of links between all topics of different clusters [51]. Similarly, Fig. 6 plots the partition distance $p$, the average semantic similarity, the cluster cohesion, and the cluster separation for different cluster numbers. The depicted example results from applying the agglomerative hierarchical clusterer, with different values of the partition distance.

In the case of hierarchical clustering, changing the value of partition distance causes sharp changes in the number of clusters, while, in graph clustering, changing the number of edges to remove causes smooth changes in the number of topic clusters. This

<table>
<thead>
<tr>
<th>Entity</th>
<th>Occurrences</th>
<th>One example topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>224,953</td>
<td>Price market dollar trade economy stock growth inflation oil global high currency</td>
</tr>
<tr>
<td>John McCain</td>
<td>61,950</td>
<td>Tax cut plan pay raise spend economy year propose income secure money rate budget</td>
</tr>
<tr>
<td>Barack Obama</td>
<td>58,199</td>
<td>Convention speech Hillary party john republican nation Denver speak campaign former</td>
</tr>
<tr>
<td>George Bush</td>
<td>25,847</td>
<td>Kill Iraq Baghdad bomb force attack suspect soldier office suicide explosion terrorist</td>
</tr>
</tbody>
</table>

select some higher value of $WSim$ to create more general topics, while others may prefer more specialized topics by selecting a lower value of $WSim$.

Fig. 5 shows the results of clustering based on edge-betweenness as was applied to the topic model built on President Obama. Fig. 5a depicts the number of removed links with respect to the different number of clusters created. Fig. 5b depicts the average semantic distance for each clustering level, using the NGD and the Wikipedia based metric, while Fig. 5c plots the average cluster cohesion and cluster separation for different clustering levels. Cluster cohesion is defined as the number of links between all topics of the same cluster, while cluster separation is defined as the number of links between all topics of different clusters [51]. Similarly, Fig. 6 plots the partition distance $p$, the average semantic similarity, the cluster cohesion, and the cluster separation for different cluster numbers. The depicted example results from applying the agglomerative hierarchical clusterer, with different values of the partition distance.

In the case of hierarchical clustering, changing the value of partition distance causes sharp changes in the number of clusters, while, in graph clustering, changing the number of edges to remove causes smooth changes in the number of topic clusters. This

<table>
<thead>
<tr>
<th>Entity</th>
<th>Occurrences</th>
<th>One example topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>224,953</td>
<td>Price market dollar trade economy stock growth inflation oil global high currency</td>
</tr>
<tr>
<td>John McCain</td>
<td>61,950</td>
<td>Tax cut plan pay raise spend economy year propose income secure money rate budget</td>
</tr>
<tr>
<td>Barack Obama</td>
<td>58,199</td>
<td>Convention speech Hillary party john republican nation Denver speak campaign former</td>
</tr>
<tr>
<td>George Bush</td>
<td>25,847</td>
<td>Kill Iraq Baghdad bomb force attack suspect soldier office suicide explosion terrorist</td>
</tr>
</tbody>
</table>

select some higher value of $WSim$ to create more general topics, while others may prefer more specialized topics by selecting a lower value of $WSim$.

Fig. 5 shows the results of clustering based on edge-betweenness as was applied to the topic model built on President Obama. Fig. 5a depicts the number of removed links with respect to the different number of clusters created. Fig. 5b depicts the average semantic distance for each clustering level, using the NGD and the Wikipedia based metric, while Fig. 5c plots the average cluster cohesion and cluster separation for different clustering levels. Cluster cohesion is defined as the number of links between all topics of the same cluster, while cluster separation is defined as the number of links between all topics of different clusters [51]. Similarly, Fig. 6 plots the partition distance $p$, the average semantic similarity, the cluster cohesion, and the cluster separation for different cluster numbers. The depicted example results from applying the agglomerative hierarchical clusterer, with different values of the partition distance.

In the case of hierarchical clustering, changing the value of partition distance causes sharp changes in the number of clusters, while, in graph clustering, changing the number of edges to remove causes smooth changes in the number of topic clusters. This

<table>
<thead>
<tr>
<th>Table 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>The 5 most common entities with one example topic identified for each of</td>
<td></td>
</tr>
<tr>
<td>them.</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Occurrences</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>United States</td>
<td>224,953</td>
</tr>
<tr>
<td>John McCain</td>
<td>61,950</td>
</tr>
<tr>
<td>Barack Obama</td>
<td>58,199</td>
</tr>
<tr>
<td>George Bush</td>
<td>25,847</td>
</tr>
</tbody>
</table>

select some higher value of $WSim$ to create more general topics, while others may prefer more specialized topics by selecting a lower value of $WSim$.

Fig. 5 shows the results of clustering based on edge-betweenness as was applied to the topic model built on President Obama. Fig. 5a depicts the number of removed links with respect to the different number of clusters created. Fig. 5b depicts the average semantic distance for each clustering level, using the NGD and the Wikipedia based metric, while Fig. 5c plots the average cluster cohesion and cluster separation for different clustering levels. Cluster cohesion is defined as the number of links between all topics of the same cluster, while cluster separation is defined as the number of links between all topics of different clusters [51]. Similarly, Fig. 6 plots the partition distance $p$, the average semantic similarity, the cluster cohesion, and the cluster separation for different cluster numbers. The depicted example results from applying the agglomerative hierarchical clusterer, with different values of the partition distance.

In the case of hierarchical clustering, changing the value of partition distance causes sharp changes in the number of clusters, while, in graph clustering, changing the number of edges to remove causes smooth changes in the number of topic clusters. This
different behavior can be easily explained as follows. In the case of graph clustering, the initial graph comprises a large number of edges, so a gradual increase in the number of edges to remove results in a slight increase in the number of derived clusters. On the other hand, in the case of hierarchical clustering, the calculation of partition distance is based only on the number of common terms between two clusters, which results into a smaller number of available values, which in turn, results into a smaller number of possible clusters.

5.4. Evaluation of the event detection algorithm

Next, we present a multivariate testing of the peak detection algorithm and we evaluate the events detected, the algorithm accuracy, and its execution time.

5.4.1. Multivariate testing

The required inputs for the event detection algorithm presented in Section 4 include the document stream, the minimum normalized average occurrences for detecting a new event ($detectionAvg$), the minimum normalized average occurrences under

![Graph showing removed links, average semantic similarity, and average cluster cohesion and separation as functions of the number of clusters.](image)

**Fig. 5.** a) Number of removed links, b) average semantic similarity, and c) average cluster cohesion and separation as functions of the number of clusters, calculated by the edge-betweenness clusterer.
which potential events are destroyed (destructionAvg), the upper threshold for detecting a new event (upLimit), the lower threshold for destroying a potential event (lowLimit), and the number of time periods to take into account when calculating the relative average times an event was encountered (hV).

Our first set of experiments includes testing the behavior of the algorithm for different values of five input variables. We conducted experiments for detectionAvg = 1 to 5, destructionAvg = 0.1 to 1, upLimit = 1.1 to 2.1, lowLimit = 0.5 to 0.9, and hV = 1 to 48. For these values we computed the number and average duration of total detected events, as well as the number of detected ground truth events (out of 199). We consider a ground truth event to be correctly identified when our algorithm detects an event where the same entities participate and the two events overlap (± 24 h due to different local hours and the one day event resolution of Wikipedia). Going over the representative results of the experiments, as depicted in Table 4, one may draw various conclusions. As expected, low values of detectionAvg lead to the detection of a larger number of events and improve the true positive rate, while lower detectionAvg values make it easier for the algorithm to detect new events. Since we are dealing with an open world problem (there is no way to know for sure that a detected event is not an actual event), we are not able to calculate the rate of false positives. On the opposite, low values of destructionAvg lead to fewer detected events. Nevertheless, high values of upLimit lead only to the detection of events referred by a large number of documents, while low values of lowLimit make it easier for an event to be removed from the potential event list, thus leading to fewer discovered events. Finally, the number and the average duration of detected events are greatly influenced by the number of historic values taken into account, as with higher
values of $hV$, peaks must be compared with a larger number of historic values, a process that makes events harder to detect, and is more time consuming.

Fig. 7 depicts the number of identified events and the percentage of the ground truth events correctly identified. The experiments include tests with different values on $detectionAvg$, $destructionAvg$, $upLimit$, $lowLimit$, and $hV$. One may notice that the accuracy of the algorithm reaches 90.45% (180 out of 199 ground truth events). Some examples of automatically detected events are available in Table 5, where the detection date, end date, and the entities participating are depicted. The ranking factor ($RF$) mentioned in this Table is the relative increase of the number of event occurrences in the data stream.

Two interesting cases of the event detection process are depicted in Figs. 8 and 9. Fig. 8 depicts documents associated with the nomination campaign of President Obama. One may notice that the time-line (that is the product of NER, topic modeling and clustering) is close to the one expected. There is a small peak on August 16 2008, when the U.S. presidential candidates John McCain and Barack Obama were interviewed by pastor Rick Warren at Saddleback Church in Lake Forest, California. Another peak appears on August 23, when Barack Obama’s campaign announced that Senator Joe Biden would be the Vice Presidential nominee. The strongest peak appears on August 27, when Obama was officially declared nominee of the Democratic Party for the 2008 presidential election. Fig. 8 depicts the results of three runs of the event detection algorithm. In the first run ($upLimit = 1.1$, $lowLimit = 0.5$, $hV = 48$) the high value of $hV$, and the low value of $lowLimit$ result into discovering only two events of very long duration. On the second run ($upLimit = 1.1$, $lowLimit = 0.5$, $hV = 48$) the high value of $upLimit$ results into detecting only a few, short-termed significant events. In the final run ($upLimit = 1.1$, $lowLimit = 0.5$, $hV = 48$) 44 different detected events are identified, as the low values of $upLimit$ and $lowLimit$ cause the algorithm constantly to detect and destroy events. In all three runs, we set $detectionAvg = 3$ and $detectionAvg = 0.5$.

Fig. 9 depicts the number of documents associated with John McCain’s and Barack Obama’s interview by pastor Rick Warren at Saddleback Church. This topic could be considered as a subtopic of President Obama’s nomination campaign, but since we have set the Wikipedia-based similarity to $WSim <= 0.1$, we consider this topic as a separate one. Notice that the values in Y axis are significantly lower than the ones in Fig. 8, as the topic of the nomination campaign is associated with more documents than just Obama’s interview.

5.4.2. Tests on historic values and average detected event duration

Another parameter that defines the algorithm’s efficiency is the interval $hV$ of historic values to be taken into account when calculating the relative average times a potential event was processed. Fig. 10 depicts the number of total events identified and the average event duration in hours ($avgDuration$) are presented. The test includes different values of $detectionAvg$, $destructionAvg$, $upLimit$, $lowLimit$, and $hV$.

<table>
<thead>
<tr>
<th>Counter</th>
<th>DetectionAvg</th>
<th>DestructionAvg</th>
<th>upLimit</th>
<th>lowLimit</th>
<th>$hV$</th>
<th>Recognized E.</th>
<th>Ground truth E.</th>
<th>Avg duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>0.3</td>
<td>1.1</td>
<td>0.5</td>
<td>1</td>
<td>3459</td>
<td>153</td>
<td>32.6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.3</td>
<td>1.1</td>
<td>0.5</td>
<td>48</td>
<td>2855</td>
<td>131</td>
<td>72.4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0.3</td>
<td>1.1</td>
<td>0.9</td>
<td>1</td>
<td>3918</td>
<td>166</td>
<td>25.7</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0.3</td>
<td>1.1</td>
<td>0.9</td>
<td>48</td>
<td>3426</td>
<td>157</td>
<td>64.7</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>0.3</td>
<td>1.9</td>
<td>0.5</td>
<td>1</td>
<td>2305</td>
<td>103</td>
<td>48.3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0.3</td>
<td>1.9</td>
<td>0.5</td>
<td>48</td>
<td>1613</td>
<td>63</td>
<td>62.5</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>0.3</td>
<td>1.9</td>
<td>0.9</td>
<td>1</td>
<td>2808</td>
<td>113</td>
<td>33.9</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0.3</td>
<td>1.9</td>
<td>0.9</td>
<td>48</td>
<td>2216</td>
<td>96</td>
<td>64.7</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1.1</td>
<td>0.5</td>
<td>1</td>
<td>4593</td>
<td>170</td>
<td>28.3</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1.1</td>
<td>0.5</td>
<td>48</td>
<td>4565</td>
<td>167</td>
<td>51.9</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1.1</td>
<td>0.9</td>
<td>1</td>
<td>4651</td>
<td>178</td>
<td>24.0</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>1</td>
<td>1.1</td>
<td>0.9</td>
<td>48</td>
<td>4675</td>
<td>178</td>
<td>48.5</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>1</td>
<td>1.9</td>
<td>0.5</td>
<td>1</td>
<td>3332</td>
<td>128</td>
<td>37.2</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>1</td>
<td>1.9</td>
<td>0.5</td>
<td>48</td>
<td>3060</td>
<td>106</td>
<td>41.9</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>1</td>
<td>1.9</td>
<td>0.9</td>
<td>1</td>
<td>3480</td>
<td>131</td>
<td>29.9</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>1</td>
<td>1.9</td>
<td>0.9</td>
<td>48</td>
<td>3363</td>
<td>123</td>
<td>43.2</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>0.3</td>
<td>1.1</td>
<td>0.5</td>
<td>1</td>
<td>457</td>
<td>71</td>
<td>49.6</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>0.3</td>
<td>1.1</td>
<td>0.5</td>
<td>48</td>
<td>353</td>
<td>58</td>
<td>107.7</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>0.3</td>
<td>1.1</td>
<td>0.9</td>
<td>1</td>
<td>672</td>
<td>100</td>
<td>31.9</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>0.3</td>
<td>1.1</td>
<td>0.9</td>
<td>48</td>
<td>476</td>
<td>77</td>
<td>81.1</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>0.3</td>
<td>1.9</td>
<td>0.5</td>
<td>1</td>
<td>267</td>
<td>37</td>
<td>72.7</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
<td>0.3</td>
<td>1.9</td>
<td>0.5</td>
<td>48</td>
<td>156</td>
<td>26</td>
<td>97.7</td>
</tr>
<tr>
<td>23</td>
<td>5</td>
<td>0.3</td>
<td>1.9</td>
<td>0.9</td>
<td>1</td>
<td>430</td>
<td>58</td>
<td>47.4</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>0.3</td>
<td>1.9</td>
<td>0.9</td>
<td>48</td>
<td>264</td>
<td>39</td>
<td>81.0</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>1</td>
<td>1.1</td>
<td>0.5</td>
<td>1</td>
<td>709</td>
<td>82</td>
<td>44.4</td>
</tr>
<tr>
<td>26</td>
<td>5</td>
<td>1</td>
<td>1.1</td>
<td>0.5</td>
<td>48</td>
<td>594</td>
<td>73</td>
<td>93.6</td>
</tr>
<tr>
<td>27</td>
<td>5</td>
<td>1</td>
<td>1.1</td>
<td>0.9</td>
<td>1</td>
<td>915</td>
<td>106</td>
<td>30.8</td>
</tr>
<tr>
<td>28</td>
<td>5</td>
<td>1</td>
<td>1.1</td>
<td>0.9</td>
<td>48</td>
<td>710</td>
<td>87</td>
<td>77.1</td>
</tr>
<tr>
<td>29</td>
<td>5</td>
<td>1</td>
<td>1.9</td>
<td>0.5</td>
<td>1</td>
<td>405</td>
<td>49</td>
<td>60.7</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>1</td>
<td>1.9</td>
<td>0.5</td>
<td>48</td>
<td>320</td>
<td>38</td>
<td>84.4</td>
</tr>
<tr>
<td>31</td>
<td>5</td>
<td>1</td>
<td>1.9</td>
<td>0.9</td>
<td>1</td>
<td>553</td>
<td>67</td>
<td>44.6</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>1</td>
<td>1.9</td>
<td>0.9</td>
<td>48</td>
<td>408</td>
<td>48</td>
<td>78.5</td>
</tr>
</tbody>
</table>
the average event duration with regard to \( hV \). In order to compare these attributes, relative values were used. One may notice that increasing \( hV \) results into a higher average duration of the detected events and a lower number of detected events.

5.4.3. Tests on time efficiency

Next, we test the execution time of our method. As discussed in Section 4.2, despite the fact that the complexity of the event detection method in the worst case is \( O(N^2) \), the expected complexity is \( O(N) \). In this Section, we confirm this claim experimentally. Figs. 11 and 12 for numerous tests with different values of \( \text{detectionAvg}, \text{destructionAvg}, \text{upLimit}, \text{lowLimit}, \) and \( hV \) plot execution times with respect to \( \text{detectionAvg} \) and \( \text{destructionAvg} \), respectively. In all tests, the order of the running time remains constant. Lower values of \( \text{destructionAvg} \) and \( \text{lowLimit} \), as well as higher values of \( hV \), increase the execution time, while the \( \text{detectionAvg} \) and \( \text{upLimit} \) variables do not significantly affect the execution time.

Furthermore, we conducted a set of tests for various sizes of the input stream. The results are depicted in Fig. 13 and confirm our initial claim that the proposed algorithm’s expected complexity is \( O(N) \). All tests were executed in a typical PC (Quad Duo Core, 4 GB Ram).

5.5. Evaluation of our methodology against other approaches

In order to further validate our approach, we evaluated it against other approaches in the context of the MediaEval2012 Social Event Detection (SED) international competition [52]. The SED competition comprised three challenges on a common test dataset of images with their metadata (timestamps, tags, geotags).

The goal of the first challenge was to identify in the test collection public technical events, such as exhibitions and fairs that took place in Germany, (i.e. the annual CeBIT exhibition). The goal of the second challenge was to find all soccer events that took place in Hamburg (Germany) and Madrid (Spain). Soccer events, for the purpose of this task, were soccer games and social events centered around soccer such as the celebration of winning a cup. The goal of the third challenge was to find demonstration and protest events of the Indignados movement occurring in public places in Madrid. The Spanish Indignados organized a series of

<table>
<thead>
<tr>
<th>Detection date</th>
<th>End date</th>
<th>RF</th>
<th>Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug. 4, 4:28</td>
<td>Aug. 21, 3:54</td>
<td>22.79</td>
<td>Georgia</td>
<td>Russia, Georgia, and South Ossetia go to war (Aug. 7–26)</td>
</tr>
<tr>
<td>Aug. 4, 4:56</td>
<td>Aug. 6 11:52</td>
<td>9.18</td>
<td>A. Solzhenitsyn</td>
<td>Solzhenitsyn dies (Aug. 3), and buried (Aug. 6)</td>
</tr>
<tr>
<td>Aug. 6, 7:57</td>
<td>Aug. 22, 1:07</td>
<td>2.17</td>
<td>J. Cheek, China</td>
<td>Joey Cheek denied visa by the Chinese government (Aug. 6)</td>
</tr>
<tr>
<td>Aug. 6, 16:47</td>
<td>Aug. 9, 22:54</td>
<td>6.34</td>
<td>Paris Hilton</td>
<td>Hilton releases parody of the Presidential campaign (Aug. 6)</td>
</tr>
<tr>
<td>Aug. 29, 15:59</td>
<td>Aug. 29, 16:57</td>
<td>25.33</td>
<td>McCain, Palin</td>
<td>Sarah Palin revealed as McCain’s Vice-President (Aug. 29)</td>
</tr>
</tbody>
</table>
(a) The high value of $hV$ results into detecting events with long duration.

(b) The high value of $upLimit$ results into detecting only short-termed significant events.

(c) The very low values of $upLimit$, and $lowLimit$ cause the algorithm constantly to detect new events, and then destroy them.

Fig. 8. Time series of the number of documents regarding the nomination campaign of President Obama. The colored lines on the top represent the duration of detected events.
public demonstrations and other protests all over Spain, which were related to the financial crisis outbreak and national politics in general. In contrast to the events that challenges one and two were concerned with, the events that were of interest to this third challenge were not scheduled, well-organized events (e.g., a technical fair or a soccer game, which are typically scheduled several months or days in advance). SED provided 167,332 photos collected from Flickr.com that were captured from 2009 to 2011. All

**Fig. 9.** Time series of the number of documents regarding John McCain’s and Barack Obama’s interview by pastor Rick Warren. The line on the top represents the duration of the detected event.

**Fig. 10.** Number of events identified, and average event duration as function of hV.

**Fig. 11.** Execution time of the proposed algorithm as function of detectionAvg.
photos were originally geotagged. However, before providing the XML photo metadata archive (including any tags, geotags, time-stamps, etc.) to the task participants, the geotags were removed for 80% of the photos in the collection (randomly selected). This was done for simulating the frequent lack of geotags in photo collections on the Internet (including the Flickr collection).

Although the SED dataset included photos augmented with metadata, we focused on textual metadata, in order to treat all photos as documents. We consider the use of visual information from our algorithm as future work. Evaluation of the submissions to the SED task performed by the organizers using ground truth that partially came from the EventMedia dataset [53] (for Challenge 1), and in part as the result of a semi-automatic annotation process carried out with the CrEve tool [54] (for all three challenges). Two evaluation measures were used: a) the Harmonic mean (F-score) of Precision and Recall for the retrieved images, which cannot measure the number of retrieved events, nor how accurate the correspondence between retrieved images and events was, and b) the Normalized Mutual Information (NMI) that compared two sets of photo clusters (where each cluster comprised the images of a single event), jointly considering the quality of the retrieved photos and their assignment to different events [55]. Both evaluation measures received values in the range [0–1] with higher values indicating a better agreement with the ground truth results. These evaluation measures were calculated both per challenge and on aggregate (for those teams that submitted runs to all challenges).

The evaluation criteria for each submission took into account the number of detected events (out of all relevant events in the test set) and the number of correct/incorrect media detected for these events. What we were looking for was a set of photo clusters, each cluster comprising only photos associated with a single event (thus, each cluster defining a retrieved event). The photos associated with a single event that we were looking for were all photos of the test collection that were directly related (in content, and also in terms of place/time) with the event of interest.

Table 6 presents the results of our methodology against other event detection approaches in the MediaEval2012 Social Event Detection Task. We present our results for two runs, one using topics automatically created by the LDA process and one using topics manually created from a domain expert for each challenge. We then compare these results with the best runs of other approaches. Topics

![Fig. 12. Execution time of the proposed algorithm as function of destructionAvg.](image1)

![Fig. 13. Execution time of the detection algorithm, when tested with different number of documents.](image2)
Table 6
Results of the proposed approach against other event detection approaches in the context of MediaEval2012 Social Event Detection Task.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>NMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) 1st challenge: Technical events in Germany.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our approach (topics compiled by the domain expert)</td>
<td>76.29</td>
<td>94.90</td>
<td>84.58</td>
<td>0.7238</td>
</tr>
<tr>
<td>Our approach (topics from LDA)</td>
<td>80.98</td>
<td>19.56</td>
<td>31.10</td>
<td>0.2112</td>
</tr>
<tr>
<td>Schinas et al. [27]</td>
<td>59.12</td>
<td>11.91</td>
<td>18.66</td>
<td>0.1877</td>
</tr>
<tr>
<td>Dao et al. [31]</td>
<td>86.23</td>
<td>59.13</td>
<td>70.15</td>
<td>0.6011</td>
</tr>
<tr>
<td>Zeppelzauer et al. [14]</td>
<td>2.52</td>
<td>1.88</td>
<td>2.15</td>
<td>0.0236</td>
</tr>
<tr>
<td>Brenner et al. [33]</td>
<td>3.86</td>
<td>12.85</td>
<td>5.93</td>
<td>0.0475</td>
</tr>
<tr>
<td>(b) 2nd challenge: Soccer events in Madrid or Hamburg.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our approach (topics compiled by the domain expert)</td>
<td>88.53</td>
<td>93.49</td>
<td>90.76</td>
<td>0.8465</td>
</tr>
<tr>
<td>Our approach (topics from LDA)</td>
<td>91.21</td>
<td>79.71</td>
<td>84.00</td>
<td>0.7684</td>
</tr>
<tr>
<td>Schinas et al. [27]</td>
<td>87.05</td>
<td>66.56</td>
<td>74.64</td>
<td>0.6745</td>
</tr>
<tr>
<td>Dao et al. [31]</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Zeppelzauer et al. [14]</td>
<td>72.46</td>
<td>17.25</td>
<td>22.99</td>
<td>0.1993</td>
</tr>
<tr>
<td>Brenner et al. [33]</td>
<td>83.24</td>
<td>69.60</td>
<td>72.59</td>
<td>0.6493</td>
</tr>
<tr>
<td>(c) 3rd challenge: Demonstration and protest events of the Indignados movement in Madrid.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our approach (topics compiled by the domain expert)</td>
<td>88.91</td>
<td>90.78</td>
<td>89.83</td>
<td>0.7380</td>
</tr>
<tr>
<td>Our approach (topics from LDA)</td>
<td>90.76</td>
<td>84.20</td>
<td>86.11</td>
<td>0.3302</td>
</tr>
<tr>
<td>Schinas et al. [27]</td>
<td>88.43</td>
<td>54.61</td>
<td>66.87</td>
<td>0.4654</td>
</tr>
<tr>
<td>Dao et al. [31]</td>
<td>86.15</td>
<td>47.17</td>
<td>60.96</td>
<td>0.4465</td>
</tr>
<tr>
<td>Zeppelzauer et al. [14]</td>
<td>73.94</td>
<td>46.87</td>
<td>47.58</td>
<td>0.3088</td>
</tr>
<tr>
<td>Brenner et al. [33]</td>
<td>22.88</td>
<td>33.48</td>
<td>27.19</td>
<td>0.1988</td>
</tr>
</tbody>
</table>

Text in italics denotes the performance achieved by our approach when a domain expert compiles topics. Bold defines optimal performance. Comparison is being performed between the approach where the domain expert compiles topics (bold and italics) and all other approaches (values in bold).

created by domain experts consisted of manually selected keywords, based on our observation of topics and aggregate statistics. According to Table 6, the proposed methodology is effective and outperforms all other approaches that participated in SED. In all three challenges, as expected, topics created by domain experts produced (slightly) better results, since these topics were manually selected from LDA produced topics, and augmented with tags of high occurrence believed to be useful.

Topics identified automatically by LDA also provide good results, with the exception of Challenge 1. Unlike Challenges 2 and 3, Challenge 1 is about technical events (mainly conferences) described by a diverse vocabulary and often comprising relatively few photos, thus resulting in topics that contain concepts from irrelevant photos. In any case, our methodology outperformed the other approaches in almost all Challenges and evaluation metrics, and won the 1st place in the overall competition.

6. Conclusions and future work

Throughout this paper we presented a methodology and the respective algorithm for the semantic detection of events from document streams that originate from Web Social Media. We argued that the “event” concept is hard to specify, as it is a socially defined concept, and we defined it in the context of our work. We presented a novel parameterizable event detection algorithm that captures the social definition of events and identifies events expressed in document streams, and we successfully integrated it with named entity recognition, topic modeling through LDA, topic clustering and semantic evaluation methods.

We support our thesis by extensively testing and evaluating the proposed methodology and algorithm on a dataset containing more than 7 million blogposts, as well as through an international Event Detection Challenge, where our methodology outperformed other approaches. Through evaluation we indicate that the proposed approach: a) accurately detects interesting events, b) creates semantically rich representations of the detected events, c) can be tuned to different perceptions of the event concept, and d) is suitable for online event detection on very large datasets, with expected complexity of O(N).

The proposed methodology can be applied for discovering and summarizing interesting events to all web users. The incorporation of named entity and topic identification into the core of our method allows the presentation of personalized information to people that are interested only in certain entities or topics. In addition, our algorithm allows the representation of events at different levels of granularity. For example, one may be interested in knowing whether a major sports event is taking place, while others may be interested in identifying all the issues related to this particular event (gold medal winners, scandals like positive doping results, opening and closing ceremonies, etc.).

An issue that deserves further discussion is the named entity recognition task, which is a key factor in our methodology. Despite all the research efforts in this domain, the effective recognition of interesting named entities in previously unknown free text documents remains an open problem. Another issue we faced in our experiments was the high rate of false positives in the named entity recognition task. We were partially able to reduce the rate of false positives by using gazetteers, constructed from open linked data, however, such solution poses other difficulties. Even if we assume that the open linked data corpora (DBPedia.org, Geonames.org, etc.) contain a complete list of all location names and all major companies, incorporating such a list in our approach may result into limiting the disambiguation potential of the method. It may also limit the set of possible discovered entities only in the set of entities contained in the gazetteer. Another major problem in this area is the entity
disambiguation task. We are processing data coming from all over the world, therefore the problem of finding similar literals with different meaning, or different literals that describe the same entity should be addressed. This problem is even more intense in the case of person entities, where maintaining a complete list of names is extremely difficult (at least for medium sized projects/teams). An efficient solution to the above discussed problems would come from the prevail of a truly semantic web, where web pages appropriately describe the existing entities by embedding semantic information using microformats, RDFa and other technologies.

Future directions of our research include the improvement of the named entity recognition and disambiguation process and the creation of a rewarding mechanism that is based on personalized choices, in order to automatically select the optimized values of the algorithm’s parameters based on user type. Moreover, fuzzy logic for the event definition and event identification may be employed. Another interesting direction is the dynamic selection of the number of topics created by LDA that will better respond to different granularity levels according to user preferences, as well as ranking of events according to their importance, based on user’s subjective criteria. Analysis and event detection of multilingual document collections is another interesting field, as well as the detection of intentionally or unintentionally seriously flawed web reports of events. In the near future, we also consider integrating semantic information from more sources, such as HTML tags and microformats, as well as focusing on related fields, such as event forecasting. Finally an issue that requires further investigation is the quantitative analysis and distinction between the time a real life event takes place the time it is being reported on the web, as this can be greatly affected by the nature of the event and the web medium where it is being reported.

References


Dr. Konstantinos Vavliakis received his PhD and Diploma of Engineering from the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki in 2013 and 2007, respectively. He is a research associate in CERTH/ITI. Since 2007 he has participated in various EU, national and private sector funded research projects as a research associate with CERTH/ITI and with the Diagnosis Centre of Ormylia Foundation, in the areas of data mining, social media analysis, semantic web, e-health and knowledge management for cultural heritage.

Dr. Andreas L. Symeonidis is an Assistant Professor of the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki and a Faculty Affiliate with the Informatics and Telematics Institute in Thessaloniki, Greece. He received his diploma and PhD from the Department of Electrical and Computer Engineering at the Aristotle University of Thessaloniki in 1999 and 2004, respectively and concluded his postdoctoral research on the evaluation of agent efficiency in 2005. His research interests include Agent-Oriented Software Engineering (AOSE) and Software Engineering (SE) Processes, data mining and knowledge extraction, intelligent systems, social networks and evolutionary computing. Dr. Symeonidis’ work has been published in over 60 papers, book chapters, and conference publications. He is the co-author of the book “Agent Intelligence through Data Mining”, published under the Springer Science and Business Media (ISBN: 0-387-24352-6, August 2005).
Prof. Pericles A. Mitkas received his Diploma of Electrical Engineering from Aristotle University of Thessaloniki in 1985 and an MSc and PhD in Computer Engineering from Syracuse University, USA, in 1987 and 1990, respectively. Between 1990 and 2000, he was a faculty member of the Department of Electrical and Computer Engineering at Colorado State University in USA. Currently, Dr. Mitkas is Professor and Head of the Dept. of Electrical and Computer Engineering at Aristotle University of Thessaloniki and Director of the Information Processing Laboratory. He is also a faculty affiliate of the Information Technologies Institute at the Center for Engineering Research and Technology—Hellas (CERTH), where he directs the Laboratory on Intelligent Systems and Software Engineering. Prof. Mitkas’ research interests include databases and knowledge bases, data mining, software agents, enviromatics and bioinformatics. His work has been published in over 220 papers, book chapters, and conference publications. He is the co-author of a book on Agent Intelligence through Data Mining.