


Framework for the Discovery of Newsworthy Events in Social Media

Fernando José Fradique Duarte, University of Aveiro, Aveiro, Portugal

Óscar Mortágua Pereira, University of Aveiro, Aveiro, Portugal

 <https://orcid.org/0000-0002-1742-6927>

Rui Aguiar, University of Aveiro, Aveiro, Portugal

ABSTRACT

The new communication paradigm established by social media along with its growing popularity in recent years contributed to attract an increasing interest of several research fields. One such research field is the field of event detection in social media. The contribution of this article is to implement a system to detect newsworthy events in Twitter. The proposed pipeline first splits the tweets into segments. These segments are then ranked. The top k segments in this ranking are then grouped together. Finally, the resulting candidate events are filtered in order to retain only those related to real-world newsworthy events. The implemented system was tested with three months of data, representing a total of 4,770,636 tweets written in Portuguese. In terms of performance, the proposed approach achieved an overall precision of 88% and a recall of 38%.

KEYWORDS

Directed Acyclic Graph, Dynamic Programming, Event Detection, Jarvis-Patrick Clustering, KNN Neighbors, Learning, Machine XGBoost, Naïve Bayes, Random Forest, SVM

INTRODUCTION

Social Media services have become a very popular medium of communication and users use these services for various different reasons. In the case of Twitter, a microblogging service, the main reasons found are (Java, Song, Finin, & Tseng, 2007): daily chatter, conversations, sharing information and reporting news. Microblogging services in particular have become very popular due to their portability, immediacy and ease of use, allowing users to respond and spread information more rapidly (Atefeh & Khreich, 2015). The popularity and real time nature of these services and the fact that the data generated reflect aspects of real-world societies and is publicly available have attracted the attention of researchers in several fields (Madani, Boussaid, & Zegour, 2014; Nicolaos, Ioannis, & Dimitrios, 2016). One such field is the field of event detection in Social Media.

Event detection in Social Media has many potential applications, some of which with significant social impact such as in the detection of natural disasters and to identify and track diseases and epidemics (Madani et al., 2014). Another relevant application can be found in the detection of news topics and events of interest or newsworthy, as real-world events are often discussed by users in these services before they are even reported in traditional Media (Papadopoulos, Corney, & Aiello, 2014; Sakaki, Okazaki, & Matsuo, 2010; Van Canneyt et al., 2014). These services however present some challenges, some of which are inherit to their design and usage (Atefeh & Khreich, 2015). In the

DOI: 10.4018/IJOI.2019070103

case of Twitter, the use of informal and abbreviated words, the frequent occurrence of spelling and grammatical errors, data sparseness and lack of context due to the short length of the messages, are just a few examples of these challenges. The diversity and nature of the topics discussed may also pose additional challenges, more specifically in the case of event detection, as most of these topics are of little interest (e.g. daily chatter). The event detection process must therefore be able to filter out these topics in order to retain only those potentially related to events of interest.

The goal of this work and also its main contribution is the implementation of a fully functional system in order to detect newsworthy events using tweets, that is, any real-world event of sufficient interest to the general public in order to be reported by the Media. To achieve this goal a similar methodology already proposed in the literature, namely Twevent (C. Li, Sun, & Datta, 2012) is used as the base of the implementation. The event detection pipeline proposed consists of the following steps: first the tweets are segmented into a set of non-overlapping segments (i.e. n-grams). These segments are then ranked according to a weighting scheme. Only the top K segments are retained for further processing, thus obtaining the event segments. A variant of the Jarvis-Patrick clustering algorithm is then used in order to cluster these event segments into candidate event clusters. Finally, these candidate event clusters are filtered in order to retain only those related to real world events of interest. This filtering step is performed by a Random Forest model.

Also, in order to try to improve the performance of the proposed system, this pipeline is further enriched in comparison to the base methodology, namely concerning its second step (obtaining the event segments) and its last step (filter the candidate event clusters). Concerning the second step, a further enrichment of the weighting scheme used to rank the segments is proposed by leveraging Wikipedia as an additional factor in its computation. This proposal attempts to favor segments according to their potential newsworthiness, by further boosting them up in the ranking relatively to more commonly used and less informative segments, as the latter may tend to dominate the top of this ranking. Regarding the last step, 5 different models were tested in order to assess their applicability to perform the final filtering step. The various features used to train these models were also studied in terms of their interrelationships (i.e. correlation) and relevance and a new engineered feature namely, *rprob* (the probability that a candidate event is in fact related to a real world event of interest) was also introduced. Finally, the implementation of the system was validated using three months of data, corresponding to 4,770,636 tweets created in Portugal and mostly written in the Portuguese language.

BACKGROUND

Event detection in Social Media has been the focus of much research and many different approaches have been proposed in order to solve this task. TvPulse (Vilaça, Antunes, & Gomes, 2015) aims to detect TV highlights using Twitter and publicly available Electronic Programming Guides (EPGs). To achieve this, semantic profiles are created for the Portuguese language and information related to the TV programs is collected from EPGs and processed. These semantic profiles are then used to identify the most representative tweets as highlights of a TV program.

Hotstream (Phuvipadawat & Murata, 2010) aims to collect, group, rank and track breaking news in Twitter. For this purpose tweets are filtered by hashtags (e.g. #breakingnews) or keywords (e.g. breaking news) often used by users to annotate breaking news. Tweets are then grouped together according to a similarity measure computed using TF-IDF along with a boost factor obtained via the use of a Named Entity Recognizer (NER).

In (Popescu, Pennacchiotti, & Paranjpe, 2011) a method is proposed to automatically detect events involving known entities from Twitter. A set of tweets created over a period of time and referring the target entity is collected. The Gradient Boosted Decision Trees framework is then used to decide whether this snapshot describes a central event involving the target entity or not.

Tedas (R. Li, Lei, Khadiwala, & Chang, 2012) is another Twitter based event detection and analysis system that aims to detect new events with a special focus on the detection of Crime and

Disaster related Events (CDE). To achieve this, spatial and temporal meta information is extracted from tweets and then indexed by a text search engine. This index can then be used to retrieve real time CDEs or answer analytical queries.

More recently (Alsaedi, Burnap, & Rana, 2017) proposes an event detection framework to detect large and related smaller scale events, with a special focus on the detection of disruptive events. A Naïve Bayes classifier is used to filter out non-event related tweets and retain only those associated with large-scale events. An online clustering algorithm is then used to cluster these tweets in order to obtain the smaller-scale events. The topics discussed in these clusters are then summarized and represented by their most representative posts or their top terms. Temporal Term Frequency–Inverse Document Frequency is proposed in order to compute a summary of these top terms.

In (Chang, 2018) a Social Network Analysis Platform (SNAP) is proposed via the implementation of a SocialNetwork API. The developed API intends to facilitate the extraction and processing of Facebook generated data as well as the visualization of the results obtained. This platform can be used for various purposes such as a Customer Relationship Management system (CRM) or as a Management Information System (MIS) for example and can be extended to many other use cases such as to perform sentiment analysis and event detection. This framework can also be used to analyze and describe the networks established between the users in terms of trust, influence and their interrelationships as well as the strength of these relationships.

Contrary to the systems just presented, the system proposed in this work aims to detect all kinds of events provided they are newsworthy and does not target any specific entity or type of event. Also, the framework developed in this work is very specific to event detection and does not take into account the networks established between the users in terms of the relationships they create.

More closely related to this work, Twevent (C. Li, Sun, et al., 2012) is proposed as a segment based event detection framework. Tweets are first split into segments (i.e. n-grams potentially representing semantic units). These segments are then ranked using a weighting scheme and the top K of these are grouped together according to their similarity using a variant of the Jarvis-Patrick clustering algorithm. The resulting candidate events are filtered according to their newsworthiness scores in order to retain only those considered to be related to real-world events. Wikipedia is leveraged to compute these scores and a user specified threshold is used to derive the filtering decision.

The system implemented in this work proposes Wikipedia as an additional factor in the computation of the weighting scheme used to rank the segments. This is done in order to boost segments further up in the ranking according to their potential newsworthiness and counter the possible dominance of more common use ones due to their greater user support. A trained Random Forest model is also used to filter the candidate events as opposed to using a user defined threshold. By using such a model it is expected to better capture the distinctive features that relate a candidate event to a real-world newsworthy event and therefore obtain better results in terms of accuracy.

FRED (Qin, Zhang, Zhang, & Zheng, 2013) further expands Twevent by considering three types of features representing the statistical, social and textual information related to the candidate events obtained and then using these features to train a SVM model to perform the filtering step. This work uses a subset of these proposed features and studies their interrelationships and relevance in order to try to optimize the training process. In this regard a new engineered feature is also proposed. Also, a Random Forest model was used as it obtained the best results out of the 5 different models assessed.

SYSTEM IMPLEMENTATION

The purpose of this work is to implement an event detection system using tweets to detect newsworthy events. Such an event could be any real-world event of sufficient interest to the general public in order to be reported in the Media (e.g. newspapers, online news). Sports (e.g. a football game), political (e.g. elections) or musical events (e.g. summer concerts) are examples of such events. The detection of these events should be conducted independently in time windows t of fixed size (e.g. a day).

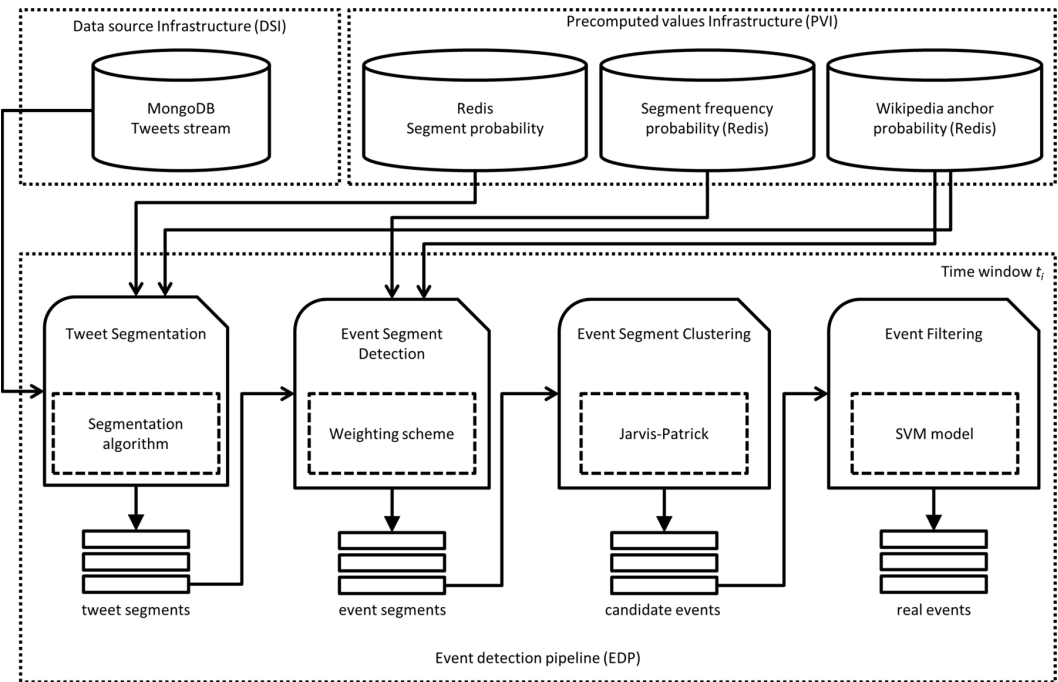
In terms of its architecture, depicted in Figure 1, the system is comprised of three main blocks: the Event Detection Pipeline (EDP), the Precomputed Values Infrastructure (PVI) and the Data Source Infrastructure (DSI). The purpose of these blocks is as follows: the EDP block is composed of four main components named respectively: Tweet Segmentation, Event Segment Detection, Event Segment Clustering and Event Filtering. These components compose the event detection pipeline used to detect events in each time window t . The PVI block is responsible for the computation and storage of the precomputed values required by the system so that they can be easily looked up later on. These values are: the Segment Probabilities (to perform semantic meaningfulness lookups), the Segment Frequency Probabilities (to detect bursty segments) and the Wikipedia Anchor Probabilities (to perform newsworthiness lookups). Lastly, the DSI block is responsible for the pre-processing of the dataset (i.e. the tweets) used to perform the event detection as well as for its storage in an appropriate format for later ease of access and retrieval. A more detailed description of each of these blocks is presented next.

Event Detection Pipeline

Tweet Segmentation Component

The goal of this component is to partition a tweet into a set of non-overlapping and consecutive segments, the so called tweet segments proposed in (C. Li, Weng, et al., 2012). A segment is similar to an n-gram in meaning except that when referring to a segment it is also implied that these n-grams could possibly refer to semantically meaningful units such as entities (e.g. the name of a person or a place) or collocations (words frequently used together). *Cristiano Ronaldo* (a famous Portuguese football player) and *Figueira da Foz* (a city) are examples of segments. For the purpose of event detection these segments convey more information than their constituent components when taken separately. As an example the segment *Cristiano Ronaldo* conveys more information than just *Cristiano* or *Ronaldo*.

Figure 1. Architecture of the system



In order to achieve this in an efficient way, a segmentation algorithm was implemented using dynamic programming. This was achieved by first considering each of the n-grams as a node and then linking these nodes together by directed edges according to the position in which they occur in the text, therefore composing a Directed Acyclic Graph (DAG). An example of this is depicted in Figure 2, where the tweet “the cat is fast” is shown, decomposed into all of its possible n-grams up to order 3 ($n = \{1, 2, 3\}$, i.e. unigrams, bigrams and trigrams).

More formally, given a DAG $G = (V, E)$ where V denotes the set of vertices or nodes of the DAG and E the set of its edges, two more special nodes named *start* and *end* are defined and linked accordingly to the other nodes, see Figure 2. The optimum segmentation can therefore be solved as the maximum cost path search between the node *start* and the node *end*, where the cost $Cost(e_i)$ of each directed edge $e_i \in E$ linking vertices u and v (i.e. $u, v \in V$) is calculated using Equation 1, which denotes the measure of the cohesiveness of a segment. In the case of the special nodes *start* and *end*, their cost is set to zero. The pseudo code of the iterative cost computation algorithm is depicted below.

Cost(start) = 0
for $v \in V \setminus \{\text{start}\}$ in linearized order
Cost(v) = $\max_{u \in \{\text{incoming edges of } v\}} (\text{Cost}(u) + C(v))$

$$C(s) = L(s) * e^{Q(s)} * S(SCP(s)) \quad (1)$$

In Equation 1 depicted above, $L(s)$ is a function used to give moderate preference to longer segments, see Equation 2, $Q(s)$ is the probability that segment s appears as an anchor text in the Wikipedia articles that contain it (i.e. Wikipedia Anchor Probability) and $SCP(s)$, computed as shown in Equation 3 is the Symmetric Conditional Probability proposed in (Silva & Lopes, 1999). Furthermore, S stands for the sigmoid function, $Pr(\cdot)$ denotes the prior probability of segment s and $Pr(w_1 \dots w_i)$ stands for the prior probability of the segment composed by the set of words $w_1 \dots w_i$.

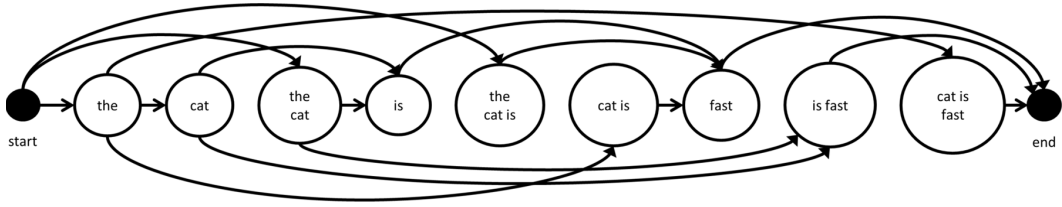
$$L(s) = \begin{cases} \frac{|s| - 1}{|s|}, & |s| > 1 \\ 1, & |s| = 1 \end{cases} \quad (2)$$

$$SCP(s) = \begin{cases} \log \left(\frac{Pr(s)^2}{\frac{1}{n-1} \sum_{i=1}^{n-1} Pr(w_1 \dots w_i) Pr(w_{i+1} \dots w_n)} \right), & |s| > 1 \\ \log(Pr(s)), & |s| = 1 \end{cases} \quad (3)$$

Event Segment Detection Component

The goal of this component is to rank the tweet segments according to a weight scheme. This ranking is then leveraged in order to select only the top K , also called event segments, for further processing as it would be computationally expensive to process the full set of segments found in a time window t (e.g. due to the potentially huge number of tweets posted in a single day). In Twevent, the base

Figure 2. Representation of a tweet as a linearized DAG



system considered for the implementation, this weight w_b is computed for each of the tweet segments according to Equation 4, where $P_b(s, t)$ denotes the bursty probability of segment s in time window t and is computed as shown in Equation 5 and $u_{s,t}$ denotes the user support of that same segment (i.e. the number of unique users that posted tweets containing segment s in time window t). The \log of the user support of the segments $u_{s,t}$ is intended to rate higher in the ranking the segments more bursty and with a higher user frequency while also allowing for the more bursty segments with a moderate user support to be ranked higher than the remaining ones.

$$w_b(s, t) = P_b(s, t) * \log(u_{s,t}) \quad (4)$$

$$P_b(s, t) = \begin{cases} 0, & f_{s,t} \leq E[s|t] \\ 1, & f_{s,t} \geq E[s|t] * 2\sigma[s|t] \end{cases}$$

$$S \left(10 * \frac{f_{s,t} - (E[s|t] + \sigma[s|t])}{\sigma[s|t]} \right), \text{ otherwise} \quad (5)$$

In Equation 5 depicted above, $E[s|t] = N_t p_s$ stands for the expected frequency of segment s in time window t (i.e. the number of tweets containing segment s), where N_t denotes the total number of tweets posted in time window t and $\sigma[s|t] = \sqrt{N_t p_s (1 - p_s)}$ represents the standard deviation as modeled by the Gaussian distribution that models $P(f_{s,t})$, the probability of observing frequency $f_{s,t}$ of segment s in time window t , see Equation 6.

$$P(f_{s,t}) \sim \mathcal{N}(N_t p_s, N_t p_s (1 - p_s)) \quad (6)$$

In practice and during the testing phase of the system, using this ranking scheme, it was however detected that the position of the segments in the rank seemed to be mostly dominated by their user support as depicted in Table 1, where the top 10 ranked segments for two randomly chosen days are listed top down according to their position in the rank (i.e. the first element in the list is ranked 1, the second is ranked 2, and so on), along with the counts for their user support (the columns on the right). Swear words were elided from the listing and are denoted with the _ symbol instead.

As it can be seen above in Table 1, the ranking positions of the segments seem to follow the same pattern for the two depicted days (i.e. segments with a greater user support are ranked higher). The only noticeable exception to this pattern occurs on the 14th and is highlighted using the * symbol. This is somewhat expectable considering that commonly used words are in general boosted by their usually greater user support $u_{s,t}$. Furthermore none of the segments listed is of particular interest in terms of the information it can potentially convey to the event detection process. Considering that from these only the top K are retained for further processing, this would mean that many informative segments would be excluded from further analysis in favor of more commonly used ones. In terms of tweet analysis this can become even more problematic as much of the topics discussed are about personal and trivial matters (i.e. heavy use of common words).

Wikipedia can be leveraged in order to attenuate this issue. More specifically, segments are boosted according to their Wikipedia anchor probability. This means that segments appearing more often as anchors (i.e. links to other articles) in Wikipedia and therefore also more likely to be informative in terms of event detection, will potentially be boosted up in the rank. This in turn would somewhat counter the apparent dominance caused by the user support of the segments already discussed. The new proposed weighting scheme is depicted in Equation 7, where $Q(s)$ denotes the Wikipedia anchor probability of segment s . Table 1 presents the list of the top 10 ranked segments computed for the same two days as before using the revised weight scheme. As it can be seen (right side of the table) the top ranking no longer seems to be dominated by the user support. Also, some of the segments listed such as *neymar* (football player), *brasil* (country), *david luiz* (football player) and *meo arena* (musical festival) seem to be clearly more informative.

$$w_b(s, t) = P_b(s, t) * \log(u_{s,t}) * e^{Q(s)} \quad (7)$$

Also, considering the specificity of Twitter (e.g. the limit imposed to its posts, use of informal language, etc), it is to be expected that in many cases the same entity may be referred to in several different forms. One such example is the use of a longer form such as *Cristiano Ronaldo* to refer to a specific person or just a shortened form such as *Ronaldo* or *Cristiano* to refer to that same person. Given that longer segments are preferred (more informative) over shorter ones, this means that the shorter segments may be dropped out during the ranking operation due to their lower ranking (in the

Table 1. Top 10 ranked segments (original weighting on the left, revised weighting scheme on the right)

06-14-2015		06-24-2015		06-14-2015		06-24-2015	
amanha	1750	es	1614	neymar	247	ganda	310
ver	1714	sei	1402	brasil	305	sdds	191
vai	1420	sempre	1388	portugal	516	ask.fm	17
dormir	1328	bue	1228	david luiz	39	cristiano araujo	39
–	1186	melhor	1228	peru	76	es	1614
assim	1108	mim	1152	mase	128	bue	1228
tempo	1055	acho	1118	colombia	59	sei	1402
exame*	994	ti	1102	meo arena	39	sempre	1388
fds*	979	aqui	1042	portugues	230	bora	184
ta*	1082	nunca	909	amanha	1750	melhor	1228

example above the segments *Cristiano Ronaldo*, *Ronaldo* and *Cristiano* are possibly competing with each other for a position on the ranking).

The downside effect of this and continuing with the previous example, is that if only the segment *Cristiano Ronaldo* is retained for further processing, all the context (.e. the tweets associated to the segment) relative to the other two segments is completely lost, possibly hindering the event detection process as this context is not taken into account. In order to try to prevent this, after the selection of the top K segments to retain for further processing, all unigrams contained in the K segments of order $n > 1$ (i.e. bigrams and trigrams) that were dropped out are added back to the ranking. In the previous example this means that if the segment *Cristiano Ronaldo* was retained for further processing, both segments *Ronaldo* and *Cristiano* will also be added back to be further processed in case they were dropped out. This tries to maximize the context related to the same entity and also possibly related to the same event in order to help the event detection process.

Event Segment Clustering Component

The goal of this component is to cluster related event segments into candidate events. To compute these candidate events, a variant of the Jarvis-Patrick clustering algorithm, which takes only the k parameter into account (i.e. the number of nearest neighbors to examine for each point) was implemented. This clustering algorithm was chosen because it is a non-iterative algorithm and therefore more efficient, as the clusters can be computed in a single pass and also because it is deterministic, meaning that the same results will be obtained every time (Jarvis & Patrick, 1973). The similarity measure $sim_t(s_a, s_b)$ used to cluster the event segments is computed as depicted in Equation 8.

To perform this computation, time window t is further sub-divided into M sub-time windows $t = \{t_1, \dots, t_M\}$. For each of these sub-time windows the similarity between segment s_a and s_b is computed by leveraging the context in which they appear (i.e. the tweets containing the segment). This is achieved by concatenating all the tweets posted in sub-time window t_m containing segment s_i into a pseudo document $T_t(s_i, m)$, which is further converted to the TF-IDF scheme and then compute the cosine similarity $sim(T_t(s_a, m), T_t(s_b, m))$ between these pseudo documents. Also, $w_t(s_i, m)$ weights the importance of sub-time window t_m relatively to segment s_i , see Equation 9, where $f_t(s, m)$ denotes the frequency of segment s in sub-time window t_m .

$$sim_t(s_a, s_b) = \sum_{m=1}^M w_t(s_a, m) w_t(s_b, m) sim(T_t(s_a, m), T_t(s_b, m)) \quad (8)$$

$$w_t(s, m) = \frac{f_t(s, m)}{\sum_{m'=1}^M f_t(s, m')} \quad (9)$$

Event Filtering Component

The goal of this component is to perform the final filtering step in order to filter the candidate events obtained in the previous step and from these retain only those related to real-world newsworthy events, also referred to as real events. A classification model was used in order to perform this filtering step. The expectation was that a good set of representative features, extracted from the candidate events obtained, could be leveraged in order to train this model. For this purpose 5 different models were assessed.

Concerning the features used, these consist of a subset of the features proposed in (Qin et al., 2013) and extracted from the candidate events e , namely: *seg*, *edge*, *wiki*, *sim*, *df*, *udf*, *rt*, *men*, *rep*, *url*, *tag* and *dup*, see Table 2. A new engineered feature *rprob* (i.e. the probability that candidate event e is a real event) is also proposed. This feature is computed as follows: the K-Nearest Neighbors (KNN) algorithm is first fitted over the annotated training dataset. For each sample (candidate event cluster), its 5 nearest neighbors are then used in order to compute this probability, see Equation 10, where w_i denotes the weight of class i for candidate event e_i and is computed as depicted in Equation 11. The distance used was the euclidean distance, denoted as $dist(e_i, e_k)$. y_k denotes the label (class) of the nearest neighbor candidate event cluster e_k and can take the value 1 (class 1) to denote the positive class (i.e. the candidate event e_k is associated with a real world newsworthy event) or 0 (class 0) to denote otherwise. This feature was introduced in order to try to leverage highly local information (that obtained from the 5 closest clusters in terms of their similarity with the target cluster) into the filtering process.

$$rprob(e_i) = \frac{w_1}{w_0 + w_1} \quad (10)$$

$$w_i(e_i) = \sum_{k=1}^5 \begin{cases} \frac{1}{dist(e_i, e_k)} & , y_k = i \\ 0 & , otherwise \end{cases} \quad (11)$$

The importance of these features was then assessed using a Random Forest Classifier ensemble, see Figure 3. Four of these features were found to be more discriminative than the rest, specifically: *rprob*, *tag*, *wiki* and *sim*. The least discriminative features found were *rt* (percentage of tweets that are retweets) and *dup* (percentage of duplicated unigrams found amongst those representing the cluster). In terms of correlation, the pair (*seg*, *edge*) presents an almost perfect correlation (0.996) as well as the pair (*df*, *udf*) (0.93) and the pair (*rep*, *men*) (0.95). A somewhat strong correlation also exists between the pairs (*seg*, *df*) (0.73), (*seg*, *udf*) (0.72), (*edge*, *df*) (0.75) and (*edge*, *udf*) (0.74). The pair (*url*, *tag*) presents a more moderate correlation (0.51).

Taking this analysis into account the features *rt* and *dup* were dropped due to their lower importance, as well as the features *seg*, *rep* and *df* due to their high correlation with other features. This was done in order to try to simplify the model (prevent overfitting) on one hand and to try to remove any adverse effect due to high correlation between the features. The remaining features, specifically *edge*, *wiki*, *sim*, *udf*, *men*, *url*, *tag* and *rprob* were then used in order to train the filtering models.

In order to choose the most suitable model for the filtering step, 5 different models were assessed, specifically: Support Vector Machines (SVM), Boosted Trees with XGBoost (Chen & Guestrin, 2016), Naïve Bayes (Gaussian), Random Forest and Logistic Regression. The Scikit-Learn implementations of these were used (Pedregosa et al., 2011). The hyperparameters of the models were tuned using Grid-Search and the resulting models were trained using cross validation and tested. As the training dataset used was highly unbalanced (a total of 1,664 candidate event clusters with only 67 of these pertaining to class 1), the models were trained and assessed using both the unbalanced dataset as well as a more balanced version of the training dataset (a total of 335 samples using all of the 67 samples representing class 1 i.e. a 80/20 ratio).

The test results obtained for class 1 only, in terms of precision, recall, F1-score and the Area Under the Receiver Operating Characteristic Curve (ROC AUC), are depicted in Table 3 (the results obtained using the balanced version of the dataset are presented on the top row). Weights, as computed

Table 2. List of candidate features used to train the 5 models assessed

Feature	Description	Formula
seg	Average number of segments of candidate event e . $Gset(t)$ denotes the set of candidate events e' computed in time window t , S_e denotes the set of segments of e and $ S_e $ the number of these segments.	$\frac{ S_e }{\max_{e' \in Gset(t)} (S_{e'})}$
edge	Average number of edges of e . E_e denotes the set of edges of e and $ E_e $ the number of these edges.	$\frac{ E_e }{\max_{e' \in Gset(t)} (E_{e'})}$
wiki	Average newsworthiness of e .	$\frac{\sum_{s=1}^{ S_e } Q(s)}{ S_e }$
sim	Average similarity of the edges of e . g denotes an edge and $sim_t(s_a, s_b)$ is computed as shown in Equation 8, where s_a and s_b denote the two segments linked by edge g .	$\frac{\sum_{g=1}^{ E_e } sim_t(s_a, s_b)}{ E_e }$
df	Percentage of tweets related to e relative to all tweets created in time window t . N_t . $T(e)$ denotes the set of tweets containing event segments of e in time window t .	$\frac{ T(e) }{N_t}$
udf	Percentage of users related to e (i.e. users that posted tweets containing at least one segment of S_e and denoted as $U(e)$) relative to the number of users U_t who posted tweets in time window t .	$\frac{ U(e) }{U_t}$
rt	Percentage of tweets that are retweets e . $T(rt)$ denotes the subset of tweets from $T(e)$ that are retweets.	$\frac{ T(rt) }{ T(e) }$
men	Percentage of tweets with user mentions in e . $T(men)$ denotes the subset of tweets from $T(e)$ that contain user mentions.	$\frac{ T(men) }{ T(e) }$
rep	Percentage of tweets that are replies in e . $T(rep)$ denotes the subset of tweets from $T(e)$ that are replies.	$\frac{ T(rep) }{ T(e) }$
url	Percentage of tweets containing url links in e . $T(url)$ denotes the subset of tweets from $T(e)$ that contain url links.	$\frac{ T(url) }{ T(e) }$
tag	Percentage of tweets containing hashtags in e . $T(tag)$ denotes the subset of tweets from $T(e)$ that contain hash tags.	$\frac{ T(tag) }{ T(e) }$

continued on following page

Table 2. Continued

Feature	Description	Formula
dup	Percentage of duplicated unigrams out of the segments of e (the unigrams were stemmed for this purpose). d_e denotes the number of duplicated unigrams out of the segments of e and u_e the total number of segments of e that are unigrams.	$\frac{d_e}{u_e}$
rprob	Probability that candidate event e is a real event	see Equation 10

by the *class_weight* function provided by Scikit-Learn were also used in order to try to attenuate the imbalanced nature of the dataset. The models that performed the best in both versions of the dataset were the Random Forest model and the XGBoost model. The SVM model on the other hand performed relatively well on the balanced version but poorly on the unbalanced one. Naïve Bayes and Logistic Regression performed the worst. Table 4 presents the detailed results for the Random Forest and for the XGBoost models obtained on the unbalanced version of the dataset.

Precomputed Values Infrastructure

Segment Probabilities

The segment's prior probabilities, denoted by $Pr(.)$ in Equation 3 are used during the tweet segmentation phase in order to try to obtain semantically meaningful units. The tweets posted during 2015 were used in order to derive these values as shown in Equation 12, as these are not provided for the Portuguese language by any online service. $C(w_1...w_n)$ denotes the counts of the n-gram $w_1...w_n$ while N denotes the total number of n-grams of the same order as n-gram $w_1...w_n$ as found in the corpus. In total 25,952,065 n-grams along with their pre-computed probabilities were stored in a Redis instance.

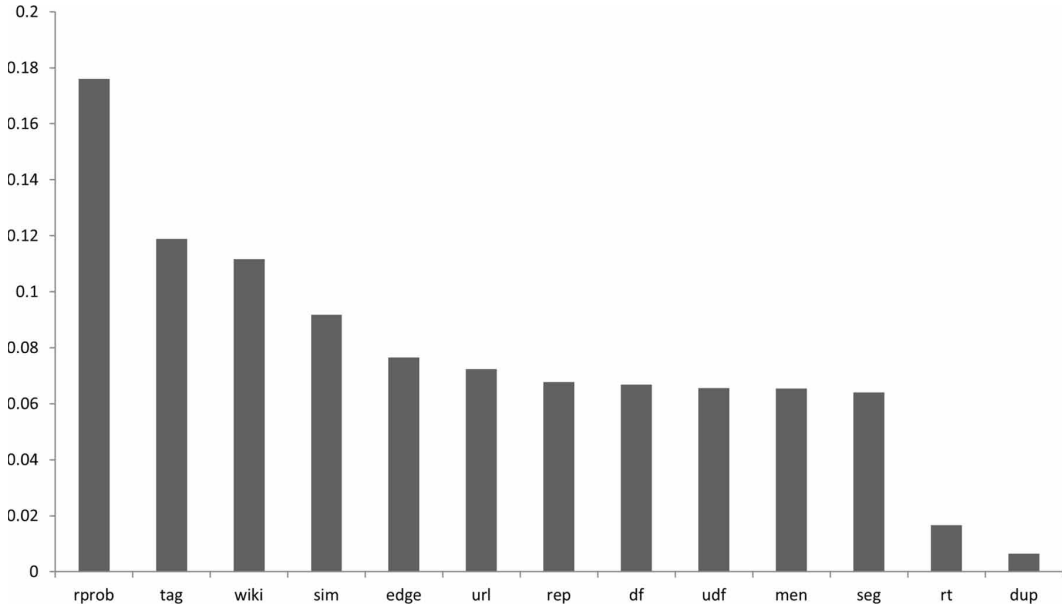
$$Pr(w_1 \dots w_n) = \frac{C(w_1 \dots w_n)}{N} \quad (12)$$

Wikipedia Anchor Probabilities

The Wikipedia anchor probabilities, denoted as $Q(s)$ in Equation 1 are used both during the tweet segmentation phase and the event segment detection phase to derive the newsworthiness of segments. In order to compute these probabilities the latest Portuguese Wikipedia dump (i.e. the ptwiki-latest-pages-articles.xml.bz2, 02-Aug-2018) was used. Equation 13 depicts this computation with $A(s)$ denoting the number of Wikipedia articles where segment s appears as an anchor and N the total number of articles containing s . Only text contained inside anchor blocks encoded as `[[.]]` in the dump file and excluding images was considered as a potential anchor text candidate. Redirect and disambiguation pages were also left out and not processed. The longest form to designate the anchors was always preferred (e.g. in `[[mm|millimeter]]` millimeter was chosen). This was done with the intuition that a longer segment tends to be more descriptive. In total 305,992 anchor designations up to n-grams of order 3 were persisted to a Redis instance.

$$Q(s) = \frac{A(s)}{N} \quad (13)$$

Figure 3. Feature importance



Segment Frequency Probabilities

The segment frequency probabilities, denoted as p_s and computed as shown in Equation 14 are used in the event segment detection phase to detect bursty segments. In the equation depicted, N_t denotes the number of tweets created within time window t , $f_{s,t}$ denotes the frequency of segment s within t (i.e. the number of tweets created in t that contain s) and L denotes the number of time windows t containing segment s in the corpus (tweets collected in 2015). In total 6,175,302 segments along with their pre-computed probabilities were stored in a Redis instance.

$$p_s = \frac{1}{L} * \sum_{t=1}^L \frac{f_{s,t}}{N_t} \quad (14)$$

SYSTEM TESTING

Dataset and Experimental Setup

The dataset used was collected from the Twitter Search API for the TVPulse project (Vilaça et al., 2015) and consists of a set of tweets created in Portugal and mostly written in the Portuguese language. Two subsets of this dataset were used: data collected from 07-01-2016 to 09-30-2016 (4,770,636 tweets) were used to test the system and data collected from 05-14-2015 to 06-24-2015 (3,581,466 tweets) were used to tune, train and test the filtering models. In terms of big international events these periods were dominated by the 2016 Summer Olympics, the UEFA EURO 2016 and the 2015 Copa America. The set of tweets collected in 2015 (16,550,792 tweets) was used to compute the Segment Probabilities and Segment Frequency Probabilities pre-computed values. All tweets were normalized by removing links, hashtags, user mentions, punctuation, numbers, accentuation and character repetitions and converting the remaining text to lowercase.

Table 3. Results obtained for the balanced (top row) and the unbalanced (bottom row) training datasets

Model	Precision	Recall	F1-score	ROC AUC
SVM	0.90	0.69	0.78	0.84
	0.42	0.38	0.40	0.68
XGBoost	0.80	0.62	0.70	0.79
	0.75	0.46	0.57	0.73
Naive Bayes	0.62	0.38	0.48	0.66
	0.29	0.54	0.38	0.74
Random Forest	1.00	0.54	0.70	0.77
	0.83	0.38	0.53	0.69
Logistic Regression	0.44	0.85	0.58	0.79
	0.00	0.00	0.00	0.5

Table 4. Detailed results obtained on the unbalanced training dataset (Random Forest and XGBoost)

Model	Classes	Precision	Recall	F1-score
Random Forest	0	0.98	1.00	0.99
	1	0.83	0.38	0.53
XGBoost	0	0.98	0.99	0.99
	1	0.75	0.46	0.57

In terms of the annotation process, a total of 4,630 candidate events were manually annotated by one of the authors (1,664 of these were used to tune, train and test the filtering models and the remaining 2,966 were used to derive the performance metrics of the tests performed on the system). The general annotation guideline followed was that a candidate event should only be labeled as referring to a real-world newsworthy event if most or all of the event segments describing it were related to that event and the event was clearly newsworthy. In all other cases it should be labeled otherwise.

The system was parameterized as follows: the size S_t of each time window t was fixed to be a whole day, the size S_m of the sub-time windows t_m was set to 2 hours and the values used for K and k were $\sqrt{N_t}$ and 3 respectively, see Table 5. To perform the tests the system was deployed in a guest environment running Ubuntu 16.04 LTS with 5 allocated processor cores, 5 GB of RAM and 80 GB of disk. VMware Player was used as the virtualization software.

Results

The results were obtained via the following procedure: first the system was used in order to compute the events for the testing periods considered. Then both the candidate events computed before and after the filtering step were manually inspected and labeled as being related to real-world newsworthy events or not. This was done in order to obtain Me , the total number of candidate events found by the system prior to the filtering step and considered to be related to real-world newsworthy events and also to calculate the number of correct Te and incorrect Fe classifications respectively concerning the real events obtained by the system after the filtering step (i.e. the final result).

These values were then used to derive the precision and recall measures of the system as shown in Equation 15 and Equation 16 as well as the F1-score. Candidate events considered to be related to the same real-world event were counted independently in order to simplify the process (i.e. two candidate

events related to the same real-world event count as two correct classifications as opposed to just one). It should be noted that concerning the computation of the recall, M_e serves as an approximation to the real number of real events present in the dataset as this number cannot be feasibly derived by manual inspection of the whole dataset.

$$precision = \frac{Te}{Te + Fe} \quad (15)$$

$$recall = \frac{Te}{Me} \quad (16)$$

Table 6 lists these results for the periods tested, where each column represents the following: TCE (total candidate events) denotes the number of candidate events computed prior to the filtering step, MCE (Manual candidate events) denotes the number of candidate events obtained prior to the filtering step, found to be related to real-world newsworthy events after manual inspection, FCE (filtered candidate events) denotes the same as the MCE column, with the exception that the candidate events inspected were the ones obtained after the filtering step. The results presented concern the performance metrics relatively to class 1 only, obtained by the Random Forest classifier trained on the balanced version of the training dataset, as this was the model that performed the best. Table 7 presents the detailed results for the 2 models that performed the best (Random Forest and XGBoost). Some of the real events identified by the system are also presented in Table 8 (the segments are separated by commas).

In terms of the performance of the system regarding processing time, depicted in Table 9, it can be seen that the components presenting the biggest bottleneck are the Event Segment Clustering component (ESC) and the Tweet Segmentation component (TS), taking in average 1.28 minutes and 1.03 minutes to compute respectively. The average processing time per time window (53,229 tweets on average) was 2.32 minutes.

DISCUSSION

In terms of the overall results obtained, as depicted in Table 6, the system presents a good precision of 88% but a fairly low recall of 38%. It can also be seen that these values vary somewhat amongst the different periods tested. Some variation can also be observed regarding the number of real events manually identified prior to the filtering step (the values shown in the MCE column), with a somewhat noticeable drop observed during the third period tested, corresponding to September with 51 real events identified. This somewhat low number of real events detected may be due to the sparsity of the dataset used (only 4,770,636 tweets spanning 3 months).

Concerning the differences observed in the precision and recall results obtained for the different periods tested, one possible reason may be due to insufficient training data, as not all types of events can be covered and these in turn may be characterized differently in terms of the values of the features of the respective candidate events obtained, according to their impact or nature. As an example of this the UEFA Champions League or even the Copa America (the events dominating the training data) may be more related to the UEFA EURO (July) event due to their similar nature than to the Summer Olympics (August). This in turn could explain the reason why July obtained the best performance measures for both precision and recall.

With respect to the quality of the real events obtained by the detection system, two remarks are noteworthy: 1) the textual representation of these events is composed in many cases of references to entities such as people (e.g. *michael phelps* in *e1*), events (e.g. *supertaca europeia* in *e2*) and football

Table 5. Parameterization used to test the system

Parameter	Description	Value
S_t	The size of time window t	1 day
K	The top-k bursty segments to retain	$\sqrt{N_t}$
k	The k-nearest neighbors to consider for Jarvis-Patrick clustering	3
S_m	The size of sub-time window t_m	2 hours

Table 6. Results obtained by the Random Forest classifier trained on the balanced training dataset

Period	TCE	MCE	FCE	Precision (%)	Recall (%)	F1 Score (%)
07-2016	1007	67	36	92	49	64
08-2016	1014	64	21	81	27	40
09-2016	945	51	23	87	39	54
Total	2966	182	80	88	38	53

Table 7. Overall results obtained by the 2 models that performed the best

Classifier	Class	Precision (%)	Recall (%)	F1 Score (%)
Random Forest	0	96	1	98
	1	88	38	53
XG-Boost	0	96	99	97
	1	65	35	46

Table 8. Examples of real events identified

ID	Segments	Event
e1	michael phelps, natacao, phelps	2016 Olympic Games
e2	sevilha, real madrid, supertaca europeia, real, madrid, penalty	UEFA Europa League final
e3	telma monteiro, bronze, telma, medalha de bronze, medalha	2016 Olympic Games, bronze medal
e4	Benfica, golo, tobias figueiredo, carrillo, nacional, marca, Jonas, marcar, raul jimenez, jogador, jimenez	Football game

clubs (e.g. *real madrid* in *e2*), that further help describe and contextualize the event, see Table 8 and 2) some of these events present mixed events or several words unrelated to the event identified (e.g. *joao souse*, *venezuela*, *caracas*, *tiago apolonia*, *estados unidos*, *tenis de mesa*, *tenis*, *natacao*, *forte*, *joao*, *del potro* where at least two events related to the 2016 Summer Olympic Games appear mixed together, namely table tennis and swimming).

Lastly, Table 10 depicts how the results obtained by the system implemented in this work can be related to the results obtained by similar implementations. Overall the system implemented in

Table 9. Running average times of the Event Detection Pipeline components

Total	TS	ESD	ESC	EF
2.32 m	1.03 m	.59 ms	1.28 m	.033 ms

Table 10. Results of the various systems

System	#Evs	Precision	Recall	N. Tweets
Twevent	101	86.1%	--	4,331,937
FRED	146	83.6%	22.9%	31,097,528
This work	80	88%	38%	4,770,636

this work detected less real events, only 80, when compared to the other two systems and achieved a higher precision and a higher recall. Twevent did not use a model to perform the filtering process and therefore the recall value is not listed. This comparison serves for the purpose of illustration only, as the datasets used by the different systems were not the same.

CONCLUSION

This work presented the implementation of an event detection system to detect newsworthy events using tweets. In terms of its main contributions, Wikipedia was proposed as an additional factor in the weighting scheme used to rank the segments, in order to favor them according to their potential newsworthiness. This proposal was validated empirically. Furthermore, 5 models were tested in order to assess their applicability to compute the real events (the filtering step). The features proposed to train these models were also analyzed in terms of their interrelationships and importance in order to optimize the learning process. In this regard a new engineered feature *rprob* was also proposed. The implemented system was tested on 4,770,636 tweets mostly written in the Portuguese language. The precision obtained was 88% with a recall of 38%. In terms of comparison with similar systems, the system implemented obtained higher precision and also a higher recall. Future work will focus on a more thorough assessment of the real impact of the change proposed to the weighing scheme used to rank the segments. Also, the results obtained in terms of precision and recall shall be further validated using data annotated by independent annotators.

ACKNOWLEDGMENT

This work is funded by National Funds through FCT - Fundação para a Ciência e a Tecnologia under the project UID/EEA/50008/2013 and SFRH/BD/109911/2015.

REFERENCES

- Alsaedi, N., Burnap, P., & Rana, O. (2017). Can We Predict a Riot? Disruptive Event Detection Using Twitter. *ACM Transactions on Internet Technology*, 17(2). 10.1145/2996183
- Atefeh, F., & Khreich, W. (2015). A Survey of Techniques for Event Detection in Twitter. *Computational Intelligence*, 31(1), 132–164. doi:10.1111/coin.12017
- Chang, V. (2018). A proposed social network analysis platform for big data analytics. *Technological Forecasting and Social Change*, 130, 57–68. 10.1016/j.techfore.2017.11.002
- Chen, T., & Guestrin, C. (2016). XGBoost : A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). San Francisco, CA: ACM.
- da Silva, J. F., & Lopes, G. P. (1999). A Local Maxima method and a Fair Dispersion Normalization for extracting multi-word units from corpora. In *Proceedings of the 6th Meeting on the Mathematics of Language* (pp. 369–381). Academic Press.
- Jarvis, R. A., & Patrick, E. A. (1973). Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Transactions on Computers*, 22(11), 1025–1034. doi:10.1109/T-C.1973.223640
- Java, A., Song, X., Finin, T., & Tseng, B. (2007). Why We Twitter: Understanding Microblogging Usage and Communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis* (pp. 56–65). San Jose, CA: ACM. doi:10.1145/1348549.1348556
- Li, C., Sun, A., & Datta, A. (2012). Twevent: Segment-based Event Detection from Tweets. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (pp. 155–164). Maui, HI: ACM Press. doi:10.1145/2396761.2396785
- Li, C., Weng, J., He, Q., Yao, Y., Datta, A., Sun, A., & Lee, B. (2012). TwiNER: Named Entity Recognition in Targeted Twitter Stream. *Proceedings of the 35th Annual Special Interest Group on Information Retrieval Conference*. doi:10.1145/2348283.2348380
- Li, R., Lei, K. H., Khadiwala, R., & Chang, K. C. C. (2012). TEDAS: A Twitter Based Event Detection and Analysis System. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering* (pp. 1273–1276). IEEE. doi:10.1109/ICDE.2012.125
- Madani, A., Boussaid, O., & Zegour, D. E. (2014). What’s Happening: A Survey of Tweets Event Detection. In *Proceedings of the Third International Conference on Communications, Computation, Networks and Technologies* (pp. 16–22). Nice, France: Academic Press.
- Nicolaos, P., Ioannis, K., & Dimitrios, G. (2016). Detecting Events in Online Social Networks: Definitions, Trends and Challenges. In *Lecture Notes in Computer Science: Vol. 9580. Solving Large Scale Learning Tasks. Challenges and Algorithms* (pp. 42–84). Cham: Springer; doi:10.1007/978-3-319-41706-6_2
- Papadopoulos, S., Corney, D., & Aiello, L. M. (2014). SNOW 2014 Data Challenge: Assessing the Performance of News Topic Detection Methods in Social Media. *Proceedings of the SNOW 2014 Data Challenge*.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, É. et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. doi:10.1007/s13398-014-0173-7.2
- Phuvipadawat, S., & Murata, T. (2010). Breaking news detection and tracking in Twitter. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (pp. 120–123). IEEE. doi:10.1109/WI-IAT.2010.205
- Popescu, A.-M., Pennacchiotti, M., & Paranjpe, D. (2011). Extracting events and event descriptions from Twitter. In *Proceedings of the 20th International Conference companion on World Wide Web* (pp. 105–106). Academic Press. doi:10.1145/1963192.1963246
- Qin, Y., Zhang, Y., Zhang, M., & Zheng, D. (2013). Feature-Rich Segment-Based News Event Detection on Twitter. In *Sixth International Joint Conference on Natural Language Processing* (pp. 302–310). Nagoya, Japan: Asian Federation of Natural Language Processing.

Sakaki, T., Okazaki, M., & Matsuo, Y. (2010). Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web* (pp. 851–860). Academic Press. doi:10.1145/1772690.1772777

Van Canneyt, S., Feys, M., Schockaert, S., Demeester, T., Develder, C., & Dhoedt, B. (2014). Detecting Newsworthy Topics in Twitter. In *Proceedings of the SNOW 2014 Data Challenge* (pp. 25–32). Seoul, South Korea: Academic Press.

Vilaça, A., Antunes, M., & Gomes, D. N. (2015). TVPulse: detecting TV highlights in Social Networks. *10th Conference on Telecommunications*.

Fernando José Fradique Duarte, Master, Informatics Engineering, University of Aveiro, graduated in Informatics Engineering, University of Aveiro in 2015.

Óscar Narciso Mortágua Pereir graduated in 1983 in Electronics and Telecommunications Engineering at the University of Aveiro (UA). Then, he devoted several years to research and development activities in several private companies. In 2013 he completed the doctoral program MAP-i and obtained a PhD degree in computer science. Since then he is an Auxiliary Professor in the Department of Electronics, Telecommunications and Informatics at the UA. Currently, he is also a researcher at the Telecommunications Institute in Aveiro. His main research activities are focused on Information Security, Big Data and IoT. He has published more than 40 papers in international conferences, international journals and book chapters. He has been also playing several roles in international conferences and journals, such as in TPC, TPC Co-Chair and Editorial Board.

Rui L. Aguiar is full Professor at the University of Aveiro where he received his PhD degree in 2001 in electrical engineering. He has been an adjunct professor at the INI, Carnegie Mellon University and is invited researcher at Universidade Federal de Uberlândia. His current research interests are centered on the implementation of advanced communication systems and he has more than 400 published papers. He is a member of the steering Board of the Network 2020 ETP. He has served as Technical and General Chair of several conferences and is Associate Editor of several journals. He is a member of ACM and a senior member of IEEE.