Offline topic detection and clustering for time limited events in Twitter

Rafael Caballero[1] and Beatriz Jiménez[2]

[1,2]Universidad Complutense, Madrid, Spain

Abstract

Nowadays social media plays an important role in our life. It has become the main way to share all kinds of content, from users' opinions about their favorites TV shows, their daily experiences, or commenting the breaking news. The study of these messages can reveal very important facts to analysts, but only after extracting the important information that is embedded in the huge flows of data. With this purpose, this work presents a method for Twitter exploration that, starting from a set of tweets published during an event, detects the topics that are more relevant based on the users' opinions. Thus, the system analyzes the event, obtaining the highlights of the most commented topics. The proposed method considers first the temporal relation of the tweets, assuming that messages published on the same time window are more likely to be a member of the same topic. Then, those tweets that belong to the same window are clustered by their textual similarity. Once the main topics related to the analyzed event have been obtained, we propose two methods to aggregate and represent a summary of the results, by temporal proximity among them or by the textual relation among their opinions.

1 Introduction

According to the report "Digital in 2018" [38], at the beginning of 2018 the number of active users on Internet reached 4 billions, passing the barrier of 50% of the world's population (about 7'6 billions inhabitants). Also, the number of social media active users is nowadays around 42%.

Twitter [34] is the leading micro-blogging social media. The user messages, known as *tweets*, allowed Twitter users to share only text fragments of 240 characters (140 characters until the end of 2017), one of the main peculiarities between this and other social media. The figures of active users per month in Twitter [33] reached the 330 millions at the end of 2017, and suppose an increment of 4% with respect the previous year, indicating that it is still growing.

Another difference between Twitter and other social media is that no reciprocity is required in order to 'follow' and track other users publications. This characteristic has boosted the use of Twitter by commercial companies to advertise their products, by public figures as channel of communication, and by accounts that offer different services, for instance by mail post services or by digital newspapers. This last use is increasing steadily according to [5], which indicates that the percentage of users that choose Twitter to get informed of the latest news increased by 15% from 2016 to 2017. As the own platform indicates [35], the main use of Twitter according to its users is "*finding something new and interesting*".

Currently, we can find different approaches about Twitter analysis, from the detection of tendencies and events [19], to the classification of events by categories [24], or to the recommendation of news to the user according to their interaction [22]. All of them have to deal with the difficulties and specific features of the social media:

- The large amount of information to be processed: as of 30th of May of 2018, a ratio of 8000 tweets per second were registered [13]. Hence, any message analysis must be able to deal with great volumes of data.
- Variety of information shared by the users: from the famous #HappyMonday to tweets that simply convey personal information that the user wishes to share with their friends. Often the tweets about breaking news are 'buried' under a great amount of information irrelevant to most of the users. Thus, it is a challenge to try to single up useful information out of this noisy and heterogeneous information. The problem is increased due to the lack of context in the messages, the grammar error, the informal language, the use of abbreviations and the many different languages that can be found.

This works focuses on the analysis of the comments in Twitter about real life events. During these events, the users emit tweets, either as a response to the *key moments*, or about the most relevant event *actors*. Detecting these user reactions can be very useful. For instance, during a political debate, the online community manager can be interested on the opinions obtained as reaction to the speech of each candidate.

Thus, the main motivation of this work is to extract the main topics of conversation in Twitter during a real event. Moreover, we would like to provide the user with different options in order to organize and display the information in a way that it becomes useful.

This motivation leads to the following particular goals:

1. The proposed algorithm must be able to detect as different topics the comments about different milestones. For example, if a team scores two goals on a football match, they must be detected as two different topics.
2. Our method must not assume that a key moment provokes only one topic of discussion. We are not interested on the detection of the key moments, but on their impact on Twitter, which can cause a variety of different reactions from the users.
3. Also, the method must be able to find topics not only during the moments of highest publishing rate. This is because we are also interested in, for example, conversations about the different actors that might occur independently of the event key moments.
4. Since the definition of "topic" depends on each particular user, it is important not to filter or exclude topics beforehand. This approach might generate a big number of topics, and therefore our method must allow the user to summary the topics clustered by different criteria.

Next section contextualizes our work. Then, Section 3 presents some general concepts important for understanding the topic detection algorithm, described in Section 4. Although the algorithm is independent of the similarity function, our particular instance of this function is presented in Section 5. Section 6 introduces the two topic aggregation methods proposed to organize and visualize the results. Finally, sections 7 and 8 suggest future lines of research and discuss the work conclusions, respectively.

2 Related Work

Twitter is a very interesting social media because in its continuous message flow we can found useful information to very diverse public: from a company that wish to measure the acceptation of a new product, to users who want to be informed about a TV show that they have missed.

In the literature, Twitter has been applied to solve problems like gender classification using the user profile information [37], study the user's influence from their number of followers [6], or analyze the social media role during a political campaign [16].

Our particular task can be included in the Topic Detection and Tracking (TDT) [1] area, which tries to detect new events among the message flows, studying possible reappearance, relationships or evolution of them. Considering the way the information in Twitter is diffused, research like [14] have shown that it is similar to traditional media, and that 85% of the topics are headlines or persistent news.

Within the TDT area, we can classify the problems by several criteria according to the way we process the data. On one hand, if the tweets are processed as they are published, the task is called *New Event Detection* (NED). On the other hand, if the tweets are analyzed offline, we can call it a *Retrospective Event Detection* (RED) task. Besides, if the event is well defined (we have some data about the event, like time interval, description or type), it is an *specified* event. If we do not have this information, then the event is *unspecified.* Finally, we can distinguish between *supervised* and *unsupervised* learning, according to the information we have about the topics we want to detect, like the categories or the number of topics. In this classification, our task can be described as a RED problem, being an unsupervised task about a specified event.

Among the systems or methods related to our work, we can find *Hotstream* [23], whose main goal is to detect news on real time, offering for each one the set of related messages and a graphic visualization of the temporal activity. This system is based on the TF-IDF model and uses the Named Entity Recognition to boost the importance of words like nouns and hashtags. *TwitInfo* [18] is a system based on the analysis of the frequency of publication of tweets, detecting the peaks of high-volume posted during an specified event and labeling each peak with its more important words and evaluating the general sentiment. Other interesting system is *ETree* [11], which detects topics related to a concrete event, trying to establish the causality relations among them, in order to understand its temporal evolution.

Many systems use clustering techniques to aggregate the tweets on topics, like [3, 29, 2]. In particular, [12] uses this method, first processing the dataset in order to remove stopwords, tweets with too many mentions or short messages. Then, they cluster the messages evaluating the similarity among them using the cosine similarity of a tweet-term matrix and using a hierarchical clustering method. Another clustering based system is [9], which aggregate the tweets using techniques like k-means and Non-Negative Matrix Factorization, once they have removed the outliers tweets. Finally, the application *TweetMotif* [21] detects topics by the identification of sets of words that tend to appear together, and then evaluate the similarity of the rest of the tweets that do not contain this keywords.

Summarizing, we observe that there are two main approaches to detect topics on Twitter:

- *Similarity based:* This kind of systems is based on the tweet comparison, principally using text similarity, to aggregate messages. The weakness of this approach is that they do not consider the temporal relation among the tweets, allowing to cluster together tweets very "distant" temporarily. Besides, this method does not allow to detect as different two topics which provoke similar reactions, like two goals in a football match.

- *Frequency based:* The second approach is based on the analysis of the ratio of publishing of the messages, detecting anomalous situations, which tend to be marked as a possible topic. The main problem of these systems is that they assume that on a peak of frequency there is only one underlying topic. Moreover, the topics that do not occur during these peaks will not be detected, assuming that there are topics only in this local maxima of frequency. Although indeed peaks of frequency correspond with an important moments during the event, they can provoke several reactions that must be detected as different topics. Also, the detection tool must be able to find topics out of these peaks.

Our method tries to solve these weaknesses by combining the two approaches in an orderly manner. Thus, we first divide the tweets on sorted sets (named *windows* of tweets), allowing us to separate potential topics with a remarkable temporal distance among them. Then, on every window we evaluate the textual similarity of its tweets, detecting subsets of related messages. Therefore, our approach uses the two characteristics which define related tweets, temporal proximity and textual similarity, being able to detect several topics about a key moment of the event.

3 General notions

In this section we introduce the basic terminology used along the paper. In the following we assume implicitly a dataset D of tweets about the event, and consider as different two tweets in D when they are emitted in different moments, even if the content of the two tweets is the same.

We say that two tweets t_1, t_2 verify $t_1 < t_2$ when t_1 has been published before t_2. Then, the *distance* between two tweets t_1, t_2 is $\text{dist}(t_1,t_2) = |\{\, t' \mid t_1 \leq t' < t_2 \,\}|$, where $|\cdot|$ represents the cardinality of the set.

A *topic* is a set of 'continuous' Twitter messages that present a certain degree of similarity among them. The notion of 'continuous' means that there is no significant gap between two tweets in the same topic.

In particular, in our setting the notion of 'gap' corresponds to a *window* representing an arbitrary number of consecutive tweets, in our case 10,000 tweets. This is a parameter that can be changed in the application configuration, but that in our experiments has proved to reach a good balance between efficiency and topic detection accuracy. We represent this parameter as $\mathcal{W}$.

A key point of our algorithm is the notion of *similarity function.* Although the particular function used in this paper is introduced in Section 5, our algorithm admits any similarity function *sim* verifying:

Definition 1 *We call* similarity *to any function* $sim : D \times D \to [0,1]$ *such that:*

1. $sim(t_1, t_2) = 1 \Leftrightarrow t_1 = t_2$ *(identity of the indiscernibles).*
2. $sim(t_1, t_2) = sim(t_2, t_1)$ *(symmetry).*

Next, we define some auxiliary concepts:

Definition 2 *Given a set of tweets S, we define:*

1. *The* mean similarity *of a tweet t with respect to S, $ms(t, S)$, as*

$$ms(t, S) = \frac{1}{|S|} \sum_{t' \in S} sim(t, t')$$

The same notation can be employed to represent the natural extension of this definition to the mean similarity of two sets S_1, S_2:

$$ms(S_1, S_2) = \frac{1}{|S_1| \cdot |S_2|} \sum_{t_1 \in S_1, t_2 \in S_2} sim(t_1, t_2)$$

2. *The* standard deviation *of tweet t with respect to S, $\sigma(t, S)$ as*

$$\sigma(t, S) = \sqrt{\frac{1}{|S|} \sum_{t' \in S} (sim(t, t') - ms(t, S))^2}$$

3. *The* center *of S, $C(S)$, as the tweet $t \in S$ such that maximizes $ms(t, S)$*

4. *The* kernel *of S at level N, $K_N(S)$ is a subset of S that includes $C(S)$ and its N-1 more similar elements in S.*

Using these concepts, now we can define formally the notion of *topic* as follows:

Definition 3 *A topic T is a subset of the original dataset that verifies:*

1. *There is not pair $t_1, t_2 \in T$, $t_1 < t_2$, such that*
   (a) *dist(t_1,t_2) $> \mathcal{W}$, and*
   (b) *There is no $t' \in T$ verifying $t_1 < t' < t_2$.*
2. *For every $t \in T$, $sim(t, C(T)) \geq ms(C(T), T) - 2\sigma(C(T), T)$*

The first point of the definition indicates that the topic cannot contain any gap of size $\mathcal{W}$ or superior. The second point ensures that there is no tweet too far from the center, that is, no too dissimilar.

4 Topic Detection

The initial dataset is obtained by using the Twitter streaming API, selecting those messages which include certain hashtags or keywords related to the event. In our implementation, the tweets are imported into the NoSQL database MongoDB to ensure horizontal scalability. Before starting the process of detecting topics, the tweets are preprocessed, keeping only original tweets, and removing from the text of the tweet urls, emoticons and mentions. Those tweets whose texts are empty after the preprocessing are removed from the dataset. The Figure 1 represents the general flow of the system.

4.1 Window Processing: Initial Topics

Initially, the tweets are ordered consecutively and divided into windows of $\mathcal{W}$ tweets, which are processed as explained in Figure 2. The Figure has the steps of this phase labeled. The first step (label 1) selects a random subset of the initial $\mathcal{W}$. In our current configuration this value is set to the 30% of the window tweets. The reasons for selecting a sample and not all the tweets in the window are:

- Limit the number of initial topics. The remainder 70% will not generate new topics, only add tweets to topics generated in this step. However, we must reach a balance between keeping the number of topics reasonably small and not missing important topics. In our experiments all the significant topics are detected from samples of 30%.

- Ease the posterior fusion among topics. The tweets in the initial sample can either create a new topic or be included in a existing topic. However, the remainder 70% of the tweets can be included in *several* topics. These *bridge tweets* (tweets in more than one topic) are very useful for merging topics.

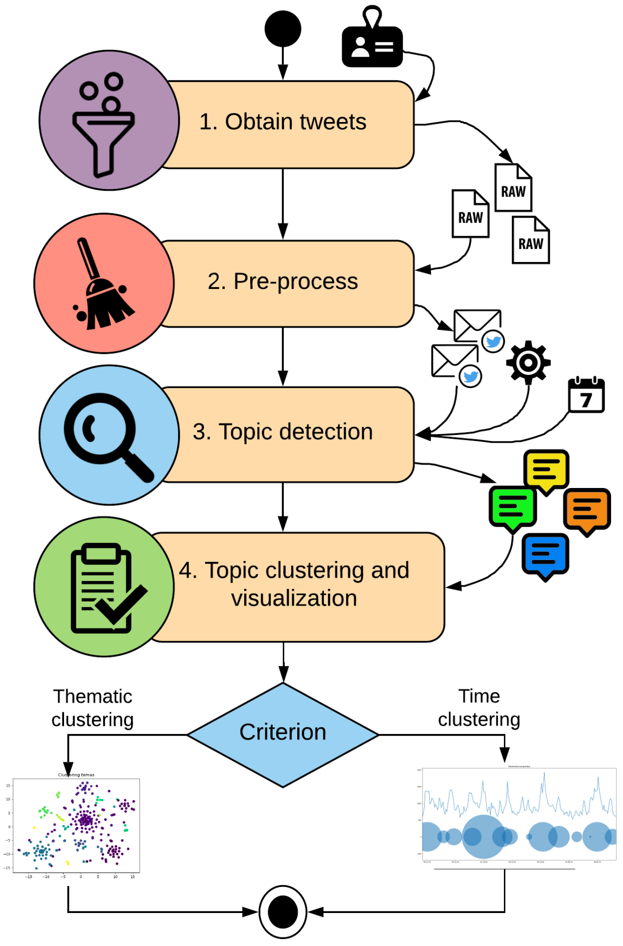


Figure 1: General flow chart

The steps portrayed in Figure 2 are:

1. The random sampling, as described above (label 1).

2. Initial topic detection. In this step (label 2), each tweet can either be added to an existing topic, or, if this is not possible, create a new one.

3. Remove topics with a small number of tweets (< 5), with few users involved (< 5), with very small textual difference (usually bots), or with very short text on average (less than two words on average), represented as label 3.

In order to check whether a tweet can be added to a topic, we use the following conditions.

Criteria 1 *Let t be a tweet, and let T be an existing topic. Then, to decide if t must be included in T:*

1. *Take a random sample S of $\mathcal{M}$ tweets from T ($\mathcal{M} = 5$ in our configuration). If there are not enough tweets, $S = T$.*

2. *For each tweet in S, we compute:*

   - $\overline{s} = ms(t, S)$.

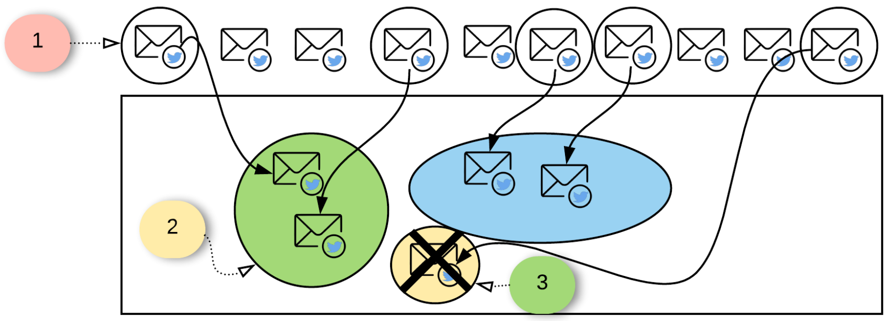


Figure 2: Initial random sampling and topics

- *$S' = \{sim(t,t') > \mathcal{U}_i \mid t' \in S\}$, where $\mathcal{U}_i$ is a initial threshold similarity parameter. The idea is to get a subset of S with really similar tweets to t.*

3. *The, the tweet t is included in T when:*

   - *$\overline{s} > \mathcal{U}_i$, that is, t is quite similar (on average) to S, and*
   - *$|S'| > \mathcal{M}'$. That is, we not only require that t is similar to S on average, but also that t is similar to a minimum amount of tweets in S (with $\mathcal{M}' \leq \mathcal{M}$, in our case $\mathcal{M}' = 4$). This condition is only required when $|S| > \mathcal{M}'$.*

In any other case, that is, if the tweet t is not considered part of any existing topic T, a new topic T' is created, with t as initial tweet, $T' = \{t\}$.

4.2 Window Processing: Adding the Remainder Tweets

After the previous phase, the 30% of the tweets have generated the initial list of topics. In this phase we add the remainder 70% to the already existing topics.

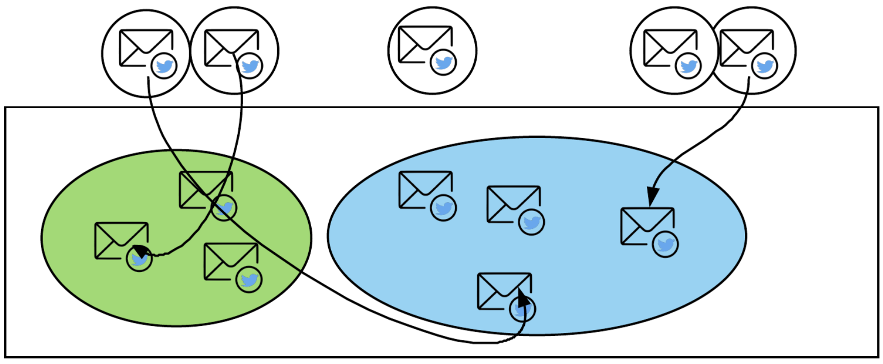


Figure 3: Adding the remainder tweets

The conditions to include the tweets in the topics are analogous to those described in Criteria 1, except for the following differences:

- The same tweet can be included in more than one existing topic.
- If no existing topic fulfills the conditions, then the tweet is discarded (no new topics are created during this phase).

Figure 3 explains this process.

4.3 Window Processing: Compacting Topics

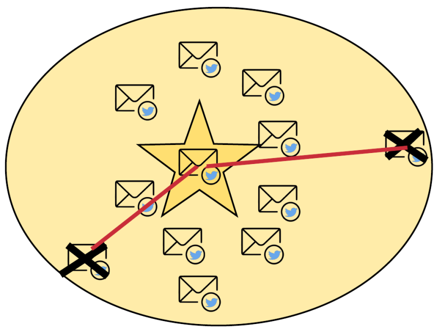


Figure 4: Compacting topics

The topics obtained so far can contain outliers, that is tweets that should not be in the topic. This happens due to the random component of the group S selected to determine if the tweet must be included in the topic. It can occur that this subset S is far away from the topic center and 'accept' as new member a tweet that is not similar enough.

Thus, once the topics have been established, we exclude the outliers by requiring a minimum average similarity with respect to the topic center, following the second condition of Definition 3.

In Figure 4 the center is represented as the central star and the topic outliers are marked cross by black lines.

4.4 Window Process: Inner Fusion

The last step in the window process phase is the fusion between topics. The idea is to merge two topics that, although detected initially as different, are in fact very similar, as Figure 5 represents.

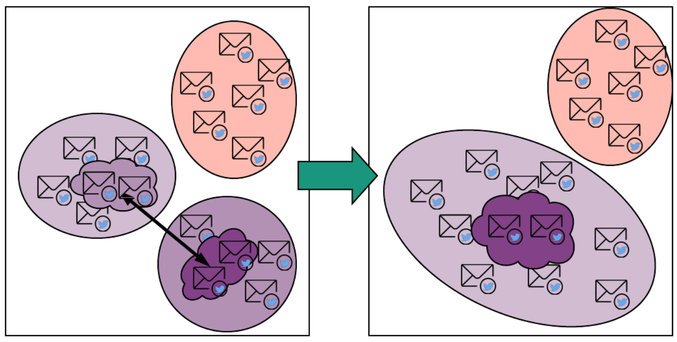


Figure 5: Inner fusion of topics

Criteria 2 *Two topics T_1,T_2 in the same window are merged into a new topic, if at least one of the following conditions hold:*

1. *The number of tweets in the intersection of the two topics, that is the number of* bridge tweets *represents at least a certain proportion $\mathcal{P}_{if}$ of the number of tweets in the union of the topics, that is $|T_1 \cap T_2| > \mathcal{P}_{if} \cdot |T_1 \cup T_2|$.*

2. *The mean similarity of the kernels of the two topics is above a threshold of inner fusion $\mathcal{U}_{if}$, that is*

$$ms(K_N(T_1), K_N(T_2)) > \mathcal{U}_{if}$$

The first condition requires a certain proportion of bridge tweets in the two topics, while the second one ensures that the two kernels are similar enough to be considered as the same topic.

4.5 Outer Fusion

During the division of the initial dataset in windows, sometimes a topic can be split into two adjacent windows. To solve this problem, one possibility is to overlap the windows. Instead of that, we have found that a reasonable solution is simply to extend the notion of fusion to topics occurring in contiguous windows, as described in Figure 6. The conditions of the out fusion are similar to those of Criteria 2, except for the criterion 2.1, because there are no bridge tweets in the case of topics in different windows. Instead, this criterion is replaced by a new one, and require that both conditions hold instead of any of them:

Criteria 3 *Two topics T_1,T_2 in consecutive windows are merged into a new topic, if at the following two conditions hold:*

1. *Consider $S = T_1 \cup T_1$. Then, the number of tweets removed after compacting S following Subsection 4.3, should not exceed a certain proportion $\mathcal{P}_{of}$ of the tweets in S.*
2. *The mean similarity of the kernels of the two topics is above a threshold of outer fusion $\mathcal{U}_{of}$, that is*

$$ms(K_N(T_1), K_N(T_2)) > \mathcal{U}_{of}$$

The new first criterion ensures that the merge of the two topics do not generate a large number of outliers, because this would mean that the new merged topic is too scattered. In our implementation we have found that $\mathcal{P}_{\text{of}} = 0.1$ gives good results.

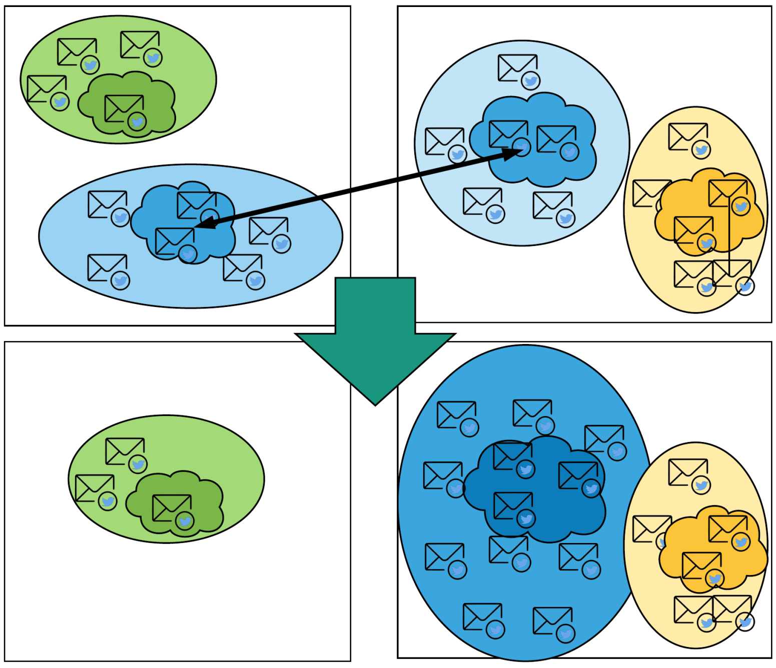


Figure 6: Outer fusion topics

Its worth commenting two particularities of the out fusion approach:

- It might happen that, after splitting the topic tweets between two windows, one of the splits, or even both, do not reach the minimum size to be detected as a topic. In these cases we could miss part of the topic (if one split is not detected as topic), or the whole topic (when both splits are undetected as topics). However, this did not happen in our experiments and we think is very unlikely in the case of relevant topics which tend to include a large number of tweets.

- Since our method analyzes each pair of adjacent windows, a topic that spreads across many different windows can be detected. This is important because some 'long lasting topics' are discussed all along the event, and deserve to be included in the list of final topics. Notice that detecting these 'long lasting topics' do not preclude the detection of other topics, because our proposal allows detecting several topics that occur simultaneously.

5 TF-IDF based similarity

As mentioned on Section 3, a key aspect of our algorithm is the ability to measure the degree of similarity between each pair of tweets of the dataset. Is important to remark that in our implementation we only consider the tweet text, after excluding stopwords. However, our approach is independent of the similarity function, which might be replaced by any other, for instance considering images, that satisfies the properties indicated in Definition 1.

5.1 Vector space model

There are a lot of measures for evaluating text similarity proposed in the literature, which consider different representations of texts (also known as documents) of the dataset. In particular, we have decided to represent the documents following the popular *vector space model* [27]. In this model, given a set of documents $D = \{d_1, d_2, \ldots, d_n\}$ and a set of terms on D (in our case terms correspond to words) $T = \{t_1, t_2, \ldots, t_m\}$, each document d_i is represented as a m-dimension vector as follows:

$$\vec{d_i} = (w_{1i}, w_{2i}, \ldots, w_{mi})$$

where w_{ji} is the weight of the term t_j on the document d_i, which quantifies the importance of the term on the document.

To compute the weights w_{ji}, one of the most common approaches is the method known as *Term Frequency-Inverse Document Frequency* (TF-IDF) [26, 17]. This method evaluates two aspects: the frequency of a term t on the document d, $tf(t, d)$, and the inverse frequency of t on the corpus of documents, $idf(t, D)$, quantifying its global importance. In particular, this second factor reduce the weight of the terms that occur very frequently. Then, w_{ji} is defined as the product of $tf(j, i)$ and $idf(j, D)$.
In our implementation, we have used an adapted version of its formulas that prevents dividing by zero and applies smoothing factors:

$$tf(t, d) = 1 + log(f(t, d)) \qquad\qquad idf(t, D) = log\frac{1 + |D|}{1 + df(t, D)} + 1$$

where $f(t, d)$ is the number of times that the term t is in the document d, and $df(t, D)$ is the number of documents in D that contain the term t.

5.2 Documents similarity

Once we have defined how we represent the tweets, it is needed to quantify the similarity among them. As each document is represented by a vector, a common way to measure the similarity degree between two vectors is to calculate the angle they form in the associated

m-vectorial space. Specifically, it is usual to compute the cosine of this angle, which in general can take any value in $[-1, 1]$, but in our case is defined in $[0, 1]$, because there are no negative values on the document vectors.

Definition 4 *Let t_1 and t_2 be two texts, and let $\vec{t_1}$, $\vec{t_2}$ be their vectorial representations, then we define*

$$cos\_sim(t_1, t_2) = \frac{\vec{t_1} \cdot \vec{t_2}}{|\vec{t_1}| \cdot |\vec{t_2}|}$$

this definition corresponds to the well-known formula that expresses the cosine of the angle formed by two vectors as the dot product of the vectors divided by the product of their modules.

Thus, a value of 1 is obtained when the vectors form a 0 degree angle, meaning that the documents are essentially the same, while values near to 0 mean that they are quite dissimilar. This cosine similarity is computed easily, performs well when there are sparse matrices (like our case) and it is independent of the length of the texts, as mentioned in [9].

5.3 Similarity measure adaptation

A characteristic of the cosine similarity when applied to corpus of small texts such as tweets, is that, given a fixed text, most of the similarities are very close to zero as shown in Figure 7, which shows the similarity between one random sentence of the book *It* by Stephen King and the rest of the book sentences.

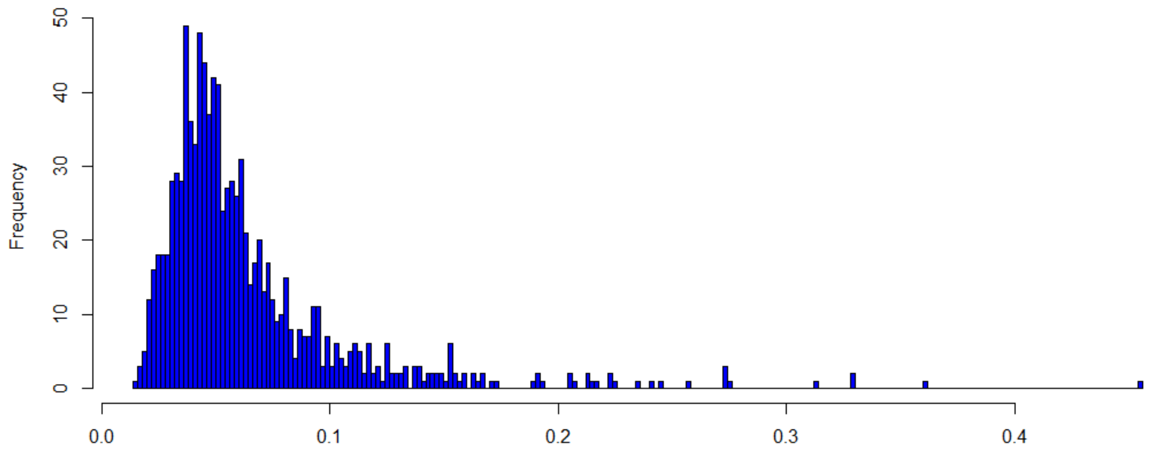


Figure 7: Similarity degree distribution of a random sentence excluding the similarity 0

The large sets of texts with very small similarity with respect to the given sentence s, at the left of the Figure, correspond intuitively to different topics with respect to the topic of s. Thus, their similarity might be considered zero to favor the outliers detection. In particular, we have chosen to define as zero the similarity values under 0.1.

In the case of similarities over 0.1, the problem is that the frequency decreases too fast, while we would prefer to have a smoother gradient of similarity to avoid that some similar tweets might be discarded as dissimilar.

In order to achieve this, we have defined our own adaptation of the cosine similarity. To do this, we examine the probability distribution depicted in Figure 7. According to some authors [31], it can be approximated by an inverse normal, or other exponential related distributions. Thus, we have chosen to use the logarithm as smoothing function:

$$sim(t_1,t_2) = \begin{cases} log_{10}(cos\_sim(t_1,t_2)) + 1 & if \quad cos\_sim(t_1,t_2) \geq 0.1 \\ 0 & if \quad cos\_sim(t_1,t_2) < 0.1 \end{cases}$$

A conceptual justification of this definition is the following diagram:

$$\begin{array}{rcl} cos\_sim \longrightarrow & [0,\dots,0.1,\dots,1] & \\ & \downarrow & \\ log_{10} \longrightarrow & [\underbrace{-\infty,\dots,-1}_{0},\underbrace{\dots,0}_{+}] & \xrightarrow{+1} [0,1] \end{array}$$

That is, the similarity function defines as zero those cosine similarities below 0.1, and the decimal logarithm of the cosine similarity plus one otherwise. Adding one to the logarithm allows us to keep the similarity interval in [0,1] as required by Definition 1. It can be checked that this function also verifies the symmetry and the identity of indiscernibles properties required by this definition.

6 Topic Clustering and Visualization

The previous algorithm can generate a large list of topics for a given event. This can be unpractical for a user who expects a short summarization of the event. A first approach could be to filter meaningless topics, or at least to establish some topic ranking. However, we have found that the notion of 'importance' depends on the particular user. Instead, we have decided to keep all the topics found, but aggregate them into clusters of related topics, that can be easily presented to the user. In particular, we present two methods for clustering topics. The first one, relates topics using a suitable concept of *temporal density.* The second one, relates topics based on their *thematic similarity.* We have found that different events can be better understood using either one or another of this methods. If the event is more actor-oriented (song contexts or political debates, for instance), the thematic clustering usually is more informative. However, in events whose developments is determined by particular circumstances (when a team scores a goal during a football match), often the temporal density is preferable.

We start by defining the concept of topic summary, employed in both clustering methods.

6.1 Topic Summary

In order to present the topics to the user, we must choose the tweet which better represents the topic content. We have already defined the center of a set of tweets (Definition 2.3). This tweet is used during the compaction phase, and represents a topic without including additional information that could avoid the detection of outliers. Although this is convenient to this purpose, the center tweet is too succinct from the perspective of the user. Instead, in the visual representation we choose the *topic summary*, a tweet, usually longer than the topic center, and that in general conveys better the topic thematic.

Definition 5 *Let T be a topic with m words, and $t \in T$ a tweet. Let $\vec{t}$ be the TF-IDF vector of t with respect to T as defined in subsection 5.1. We call* TF-IDF average *of a tweet t to the expression*

$$\frac{\sum \vec{t}}{m}$$

Then, the summary of T is the tweet $t \in T$ with minimum TF-IDF average.

The idea is that a small TF-IDF average indicates that t is very similar to the rest of the tweets in the topic, because smaller values in the TF-IDF vector components correspond to common words in the topic which are the words that best define the topic thematic. However, if we simply consider the sum of the vector, tweets with few words would have an advantage. Thus, we consider the average to achieve independence of the tweet length.

In the next subsection, we describe the two clustering methods proposed to show the topics to the user aggregated in a meaningful way.

6.2 Time Clustering

This visualization aggregates topics occurring around the same key moment. Although topics can last an arbitrarily long time interval, in this visualization we consider the topic as occurring on its *time center*, which corresponds to the median of the publication times of all the tweets in the topic.

Then, the overall idea is to show together topics whose time centers occur almost simultaneously. To formalize the concept of 'almost simultaneous', we define a notion of *temporal density.* Given a time interval $\mathcal{I}$ we define the density at $\mathcal{I}$ as

$$\text{density}(\mathcal{I}) = \frac{\#topics(\mathcal{I})}{|\mathcal{I}|}$$

where $\#topics(\mathcal{I})$ represents the number of detected topics whose time center lies in $\#topics(\mathcal{I})$, and $|\mathcal{I}|$ represents the duration of the interval in some time unit, in our case in minutes.

We call *moment of density* d to any aggregation of topics that occur in an interval $\mathcal{I}$ such that density$(\mathcal{I}) \geq d$. Our clustering algorithm starts looking for the moment with highest possible density. If our time unit is one minute and the total number of topics detected is T, then this maximum density is T topics/min. If this (highly improbable) scenario happens, then the algorithm finishes, producing as output this single moment of maximum density. Otherwise, the required density is reduced by increasing the duration of the interval and decreasing the number of topics required simultaneously. The ratio of density reduction depends on a parameter n established by the user, which indicates the *granularity* of the clustering algorithm.

The algorithm 1 describes the process followed to obtain the moments at each level k, with $0 \leq k < n$. Given the number of levels $n \geq 0$, the total number of topics $|T|$, the duration of the event D, and the centers C of the topics found, the algorithm starts defining two constants:

$$t = \sqrt[n-1]{D} \qquad\qquad g = \sqrt[n-1]{|T|}$$

where g represents the factor of reduction in the number of topics, and t the factor of increase in the duration of the interval. Thus, for each k, $nm = t^k$ represents the length of the time interval, and $nt = \frac{|T|}{g^k}$ the number of topics to be found. Then, the algorithm computes the density associated to this level k as $d = \frac{nt}{nm}$. Notice that for $k = 0$, $nm = 1$, $nt = |T|$, we obtain the maximum density $d = |T|$, while for $k = n - 1$, $nm = D$, $nt = 1$, which represents the minimum density ($d = \frac{1}{D}$, that is, one topic in the whole event).

The auxiliary function *searchMoments(C,d)*, not included for the sake of space, simply looks for moments of density greater or equal to d using the time centers of C. These intervals constitute the *moments of level k*, and are added to variable *moments*, while their centers are removed from C.

The idea is that each moment summarizes the topics that reflect some important instant of the event, which is more important (in the sense that has caused more discussion) for smaller values of k.

Figure 8 shows the frequency diagram corresponding to the tweets emitted during the second half of the football match between the clubs Paris Saint Germain and Real Madrid (6th of March, 2018). We have labeled the main key moments, which are associated to the

Algorithm 1: Density-based clustering of topics

```
Input:
    n: number of levels
    D: event duration, in minutes
    —T—: Total of topics found
    C: Temporal centers of the topics found
Output: moments: a list with moments found at level $k$ with $0 \leq kn$
$g = \sqrt[n-1]{|T|}$; $t = \sqrt[n-1]{D}$; $k = 0$;
moments = [];
while $k < n$ and $len(C) > 0$ do
    $nm = t^k$;
    $nt = \frac{|T|}{g^k}$;
    $d = \frac{nt}{nm}$;
    moments_k = searchMoments($C$,$d$);
    for $m$ in moments_k do
        moments.append((m,k));
        $C$.removeTopics(m);
    end
    k = k+1;
end
```

local maxima. The blue dots in the bottom part show the temporal center of the topics found by our tool. We have drawn a red circle around what visually seem to be the intervals with a greater density of moments.

The Figure 9 is the graphical output produced by our clustering algorithm for this dataset. The colored circles correspond to the moments identified by our time clustering algorithm, which indeed correspond to highest density moments. Notice that four of the five key moments have been detected. The figure also shows the summaries of the topics in the first moment, including their temporal centers. The algorithm was tried with n=5, and the four main moments correspond two level k=2, while the two smaller, yellow circles correspond to two moments of level k=3, the first one aggregating a second pull of topics related to the first key moment, and the second one associated to the comments emitted at the end of the match. Levels 0,1 and 4 are empty in this example.

6.3 Thematic Clustering

Another way of aggregating topics is looking for thematic relationships, instead of the temporal proximity used in the previous subsection. This second clustering approach is based on the concepts presented in Section 5.

In particular, we start considering the TF-IDF similarity with two particularities:

- Only the tweets included in the detected topics are considered instead of the whole dataset.
- Only the TF factor is computed, without multiplying by the IDF factor.

Regarding the second point, as explained in subsection 5.1, the role of IDF is to reduce the weight of those words that occur very frequently. However, when the topics have been

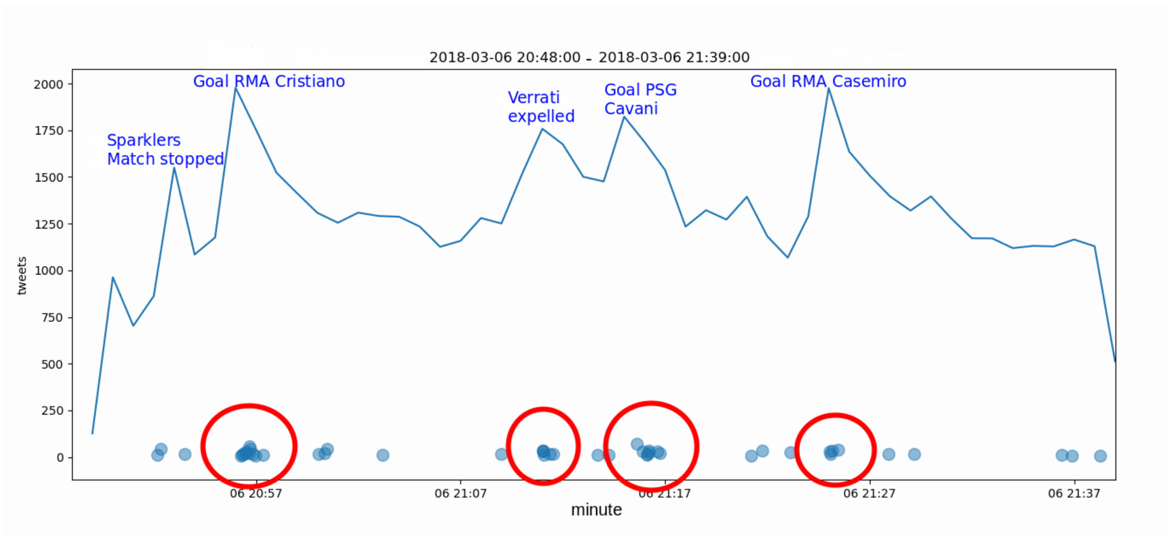


Figure 8: Second half of the PSG vs RMA football match, including frequency diagram, key moments, and detected topics

already detected, the words that are repeated, often correspond to the terms that characterize a topic, and that allow us to relate two or more topics as similar.

For instance, consider that G is the set of topics in a football match that correspond to a scored goal. If we included the multiplication by the IDF factor while we are aggregating thematically the topics, the weight of the word "GOAL" will be diminished, and will not serve to cluster the topics in G. Thus, using only the TF factor allows aggregating topics with the same keywords.

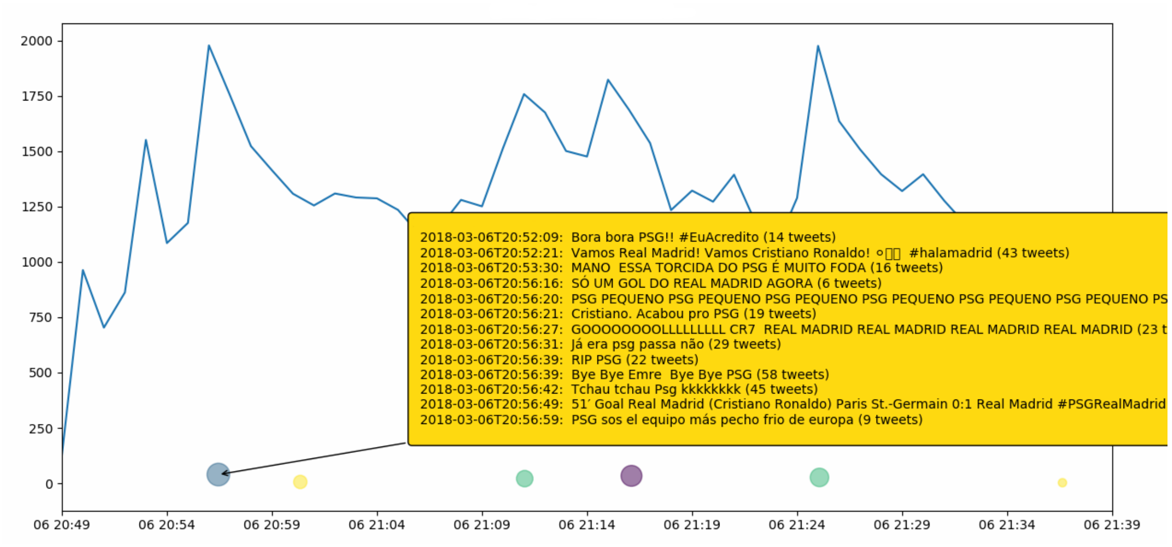
After obtaining the TF vectors, we apply the k-means clustering algorithm to the vectors that represent the center of each topic. Notice that we only consider the centers because the topics already include similar tweets. As usual in k-means, the problem is to determine the value k, that is, the number of clusters or, in our case, thematics. We have used two criteria indexes, the Silhouette index [25], and the Calinski and Harabaz index [4], to determine the optimum k such that $3 \leq k \leq 20$. In case of disagreement between the two indexes, our experiments suggest that the best option is to choose the maximum of the proposed k's.

Our final goal is to represent the clusters obtained by k-means in a 2D graphic, representing each cluster in a different color. However, the vectors representing the topic centers are multidimensional vectors, with as many dimensions as words appear in the dataset. Thus, we need to apply dimensionality reduction techniques. This is done in two steps:

1. First, *Latent Semantic Analysis* (LSA) [15] is employed to reduce the number of dimensions to one hundred, a suitable dimension number according to [28]. LSA is a well-known statistical model used in information retrieval that groups words occurring in similar contexts using singular value decomposition (SVD) [8].

2. Then, t-SNE [36] is used to reduce the number of dimensions to 2. This technique builds a model that assigns a probability to each pair of vectors according to their similarity. It also establishes a model over the output dimensions (2D in our case). Then, it tries to minimize the differences between both probability distributions following an iterative process. The technique is very suitable for a visual representation of the data, but is inefficient when dealing with high dimensional vectors. This justifies the application of the first step.

Figure 10 shows the thematic clustering visualization for the tweets obtained during the final of a famous Spanish song contest. The central cluster, in red, corresponds to topics



```
2018-03-06T20:52:09:  Bora bora PSG!! #EuAcredito (14 tweets)
2018-03-06T20:52:21:  Vamos Real Madrid! Vamos Cristiano Ronaldo! ○▯▯  #halamadrid (43 tweets)
2018-03-06T20:53:30:  MANO  ESSA TORCIDA DO PSG É MUITO FODA (16 tweets)
2018-03-06T20:56:16:  SÓ UM GOL DO REAL MADRID AGORA (6 tweets)
2018-03-06T20:56:20:  PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PSG PEQUENO PS
2018-03-06T20:56:21:  Cristiano. Acabou pro PSG (19 tweets)
2018-03-06T20:56:27:  GOOOOOOOOLLLLLLLLL CR7  REAL MADRID REAL MADRID REAL MADRID REAL MADRID (23 t
2018-03-06T20:56:31:  Já era psg passa não (29 tweets)
2018-03-06T20:56:39:  RIP PSG (22 tweets)
2018-03-06T20:56:39:  Bye Bye Emre  Bye Bye PSG (58 tweets)
2018-03-06T20:56:42:  Tchau tchau Psg kkkkkkkk (45 tweets)
2018-03-06T20:56:49:  51′ Goal Real Madrid (Cristiano Ronaldo) Paris St.-Germain 0:1 Real Madrid #PSGRealMadrid
2018-03-06T20:56:59:  PSG sos el equipo más pecho frio de europa (9 tweets)
```

Figure 9: Second half of the PSG vs RMA football match, including frequency diagram, and detected moments, including the topic summaries of the first moment

not closely related. This is usual because t-SNE tends to separate dissimilar vectors, which therefore are usually far from the center. Consequently, well-formed clusters appear usually in the periphery of the diagram. In this example, they correspond to each of the contest singers, showing the effectiveness of this technique in actor-driven events. The only exception is the the small blue cluster which corresponds to general comments at the beginning of the contest.

However, in this type of events the temporal clustering is less informative, which justifies that we include both clustering algorithms in our proposal. allowing the user to choose the best representation depending on the nature of the event.

7 Future Work

During the development of the work, different lines of research have arisen.

Topic validation

The usability of the proposed method depends on the accuracy of the detected topics. However, this is an unsupervised technique and checking the results is far from straightforward. In some events, such as football matches, we have compared the list of topics detected by our tool with the time-line of the match published on the online sport sites, finding that all the main highlights are detected. However, a more elaborated validation would require the manual identification of topics by different people.

Extending the notion of similarity

Although the experiments produced good results regarding the detection of topics, an open question is considering other features of the tweets in order to extend and improve our notion of similarity beyond the textual similarity. The images, in particular, play a very important role in Twitter and in general in all the social media. However they are usually discarded. To consider images in the similarity two steps must be considered. The first step, and possibly the most complicated, is to define suitable definitions of similarity for images, which can include scaling and parceling, checking features like the proportion of every color, or using techniques based on the detection of contours [30]. The second step is then to redefine the concept of similarity between two tweets, where now each tweet is represented by two components $x = (x_1, x_2)$, with x_1 the text and x_2 the image. The definition of the

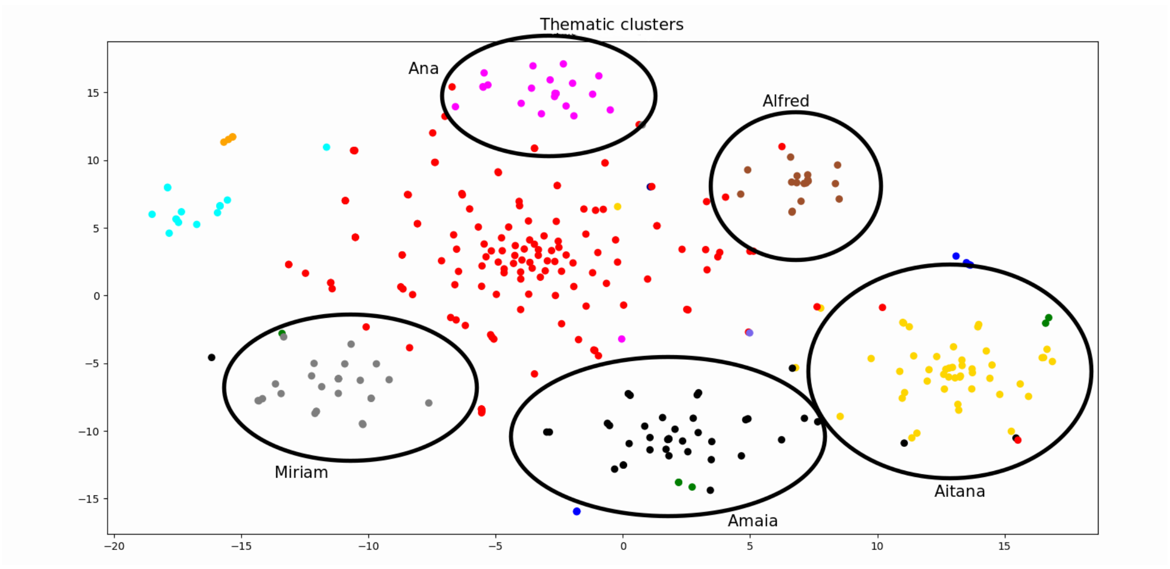


Figure 10: Song context thematic clusters

similarity *sim*, will depend both on the concept of similarity associated to each component, like the *coefficient of general similarity* [10], which is defined as:

$$sim(x,y) = \frac{1}{\sum_{k=1}^{d} w(x_k,y_k)} \sum_{k=1}^{d} w(x_k,y_k)\ sk(x_k,y_k)$$

where x, y represent, in our setting, the vectors with the two components (text and images) of the two tweets ($d = 2$). The function $w(x_k, y_k)$ takes the value 0 or 1 depending on weather the comparison of the values makes sense, and the functions s_k represent the similarity function for each component.

Improving efficiency

Using MongoDB makes our proposal scalable in terms of space. However, the time required for large datasets can become excessive. To overcome this limitation, from our idea of using windows of tweets rise in a natural way the idea of processing each window in parallel, distributing the data among several computers, thus increasing heavily the performance in terms of time of the algorithm of detecting topics.

Derived words and synonyms

The disambiguation of words, often carried out by a process of steaming, converts derived words into their semantic root, which is often useful for detecting similarity. However, this technique is not very successful when applied to tweets, due to their short length and lack of context. Our last line of future research proposes to improve this method by using Word Embedding [20]. Using the similarity of words (and taking into account the context) of this model, we can implement a version of the similarity that solves the problem of derived words and synonyms. A recent research [7] employs this idea for detecting topics in Twitter. However, in our opinion, these works can be improved if they would consider also the temporal proximity as a relevant factor when deciding if two tweets belong to the same topic or not.

Our initial experiments using deep learning algorithms based on the library TensorFlow [32], show that this technique indeed can be employed to detect words that occur in the same context. Figure 11 depicts an example of the visual representation of the contextual relation of words in a dataset of tweets published during the last American presidential elections.

Observe that the technique represents opposite terms as very close, such as love/hate or good/bad. This is not a problem, because tweets such as “Trump attitude is bad” and

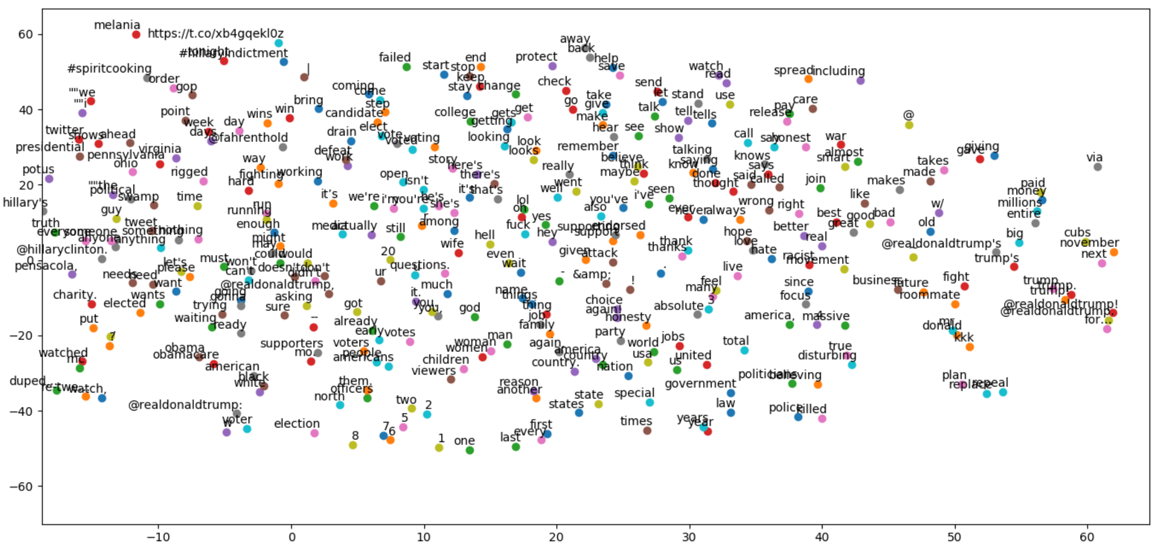


Figure 11: TensorFlow relation of tweets words

"Trump attitude is good", will have a higher similarity in the new setting, which is desirable since, indeed, both tweets correspond to the same topic.

8 Conclusions

This work has presented a technique for analyzing the Twitter messages about any particular event. The application takes as input the tweets associated to the event, and aggregate them into topics of similar content, allowing the user to have a global vision both of the key moments and of the key actors of the event.

The proposed framework uses a notion of similarity based in the model of representation of documents in a vectorial space and the numerical model TF-IDF. However, this function could be replaced by any other similarity function verifying certain conditions, such that producing a value in the interval [0,1] for each pair of tweets.

Our work tries to combine the two alternatives proposed usually in the literature for detecting topics: temporal proximity and textual similarity. Each of these proposals, when considered separately, present some flaw. The temporal proximity, based on frequency analysis, locates the key moments of the event, but it does not extract the messages emitted as reactions to these key moments. On the other hand, the textual similarity can aggregate tweets occurring on different times but with similar contents, thus mixing up different news. Our framework considers the temporal proximity by dividing the initial dataset into windows of chronologically consecutive tweets. The second proposal is then applied to these windows, looking for textual similarity, but now inside of a temporally limited interval. In this way, our framework eludes the identification of tweets with similar content that occurred in different moments as part of the same topic (as far as they do not occur in the same window). At the same time, we distinguish among the several topics associated to every key moment. In this way we fulfill the goals 1 and 2 proposed in the introduction. Our topic search does not focus on the frequency of tweets, thus allowing the detections of general comments independent of the key moments, for instance about the key actors of the event, thus achieving goal 3.

An important conclusion of this work is that the key moments of a real event must not be confused with their impact in the social media. A key moment usually generates several topics, which can include a direct mention of the key event, but also many comments about its origin, implications and related opinions. Hence, limiting the search of topics to those corresponding to the key moments implies to reduce, and thus to miss, the variety of the comments emitted by the social media users.

Our approach generates a great amount of topics. Instead of filtering these topics, discarding some of them and therefore missing some maybe important information, we have chosen to aggregate the topics during the presentation to the user. This aggregation can be done either relating thematically close topics, or using temporal aggregation. In this way we achieve the goal 4. We have found that each of the two aggregations match a different type of event. The temporal aggregation corresponds to key moment-driven events, such as sport competitions.The thematic aggregation matches better actor-driven events such as political debates, song contests, or general comments about some trending topic.

The complete code of our tool, together with some datasets can be found at `https://github.com/bejiol/TFM`.

Acknowledgement

The work is partially supported by the *Spanish Mineco* project *PID2019-106254RB-I00.*

References

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study final report. 11 2000.

[2] N. Alnajran, K. Crockett, D. McLean, and A. Latham. Cluster analysis of twitter data: A review of algorithms, 01 2017.

[3] H. Becker, M. Naaman, and L. Gravano. Learning similarity metrics for event identification in social media. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 291–300, New York, NY, USA, 2010. ACM.

[4] T. Caliński and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[5] P. R. Center. News use across social media platforms 2017, 2017.

[6] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *ICWSM*, 2010.

[7] X. Dai, M. Bikdash, and B. Meyer. From social media to public health surveillance: Word embedding based clustering method for twitter classification. In *SoutheastCon 2017*, pages 1–7, March 2017.

[8] S. T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 1990.

[9] D. Godfrey, C. Johns, C. Meyer, S. Race, and C. Sadek. A Case Study in Text Mining: Interpreting Twitter Data From World Cup Tweets. *ArXiv e-prints*, Aug. 2014.

[10] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857–871, 1971.

[11] H. Gu, X. Xie, Q. Lv, Y. Ruan, and L. Shang. Etree: Effective and efficient event modeling for real-time online social media networks. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 300–307, Aug 2011.

[12] G. Ifrim, B. Shi, and I. Brigadir. Event detection in twitter using aggressive filtering and hierarchical tweet clustering. 1150:33–40, 01 2014.

[13] Internet live stats, 2018.

[14] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 591–600, New York, NY, USA, 2010. ACM.

[15] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.

[16] A. Larsson and H. Moe. Studying political microblogging: Twitter users in the 2010 swedish election campaign. 14:729–747, 08 2012.

[17] C. D. Manning, P. Raghavan, and H. Schütze. Scoring, term weighting, and the vector space model. pages 100–123, 01 2008.

[18] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *CHI*, 2011.

[19] M. Mathioudakis and N. Koudas. Twittermonitor: Trend detection over the twitter stream, 01 2010.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[21] B. O'Connor, M. Krieger, and D. Ahn. Tweetmotif: Exploratory search and topic summarization for twitter, 2010.

[22] O. Phelan, K. McCarthy, and B. Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 385–388, New York, NY, USA, 2009. ACM.

[23] S. Phuvipadawat and T. Murata. Breaking news detection and tracking in twitter. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 120–123, Aug 2010.

[24] A. Ritter, Mausam, O. Etzioni, and S. Clark. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1104–1112, New York, NY, USA, 2012. ACM.

[25] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[26] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, NY, USA, 1986.

[27] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975.

[28] Scikit Learn v0.19.1. Truncated svd, 2017.

[29] A. Sechelea, T. D. Huu, E. Zimos, and N. Deligiannis. Twitter data clustering and visualization. In *2016 23rd International Conference on Telecommunications (ICT)*, pages 1–5, May 2016.

[30] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *J. Electronic Imaging*, 13(1):146–168, 2004.

[31] S. Tata and J. M. Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *SIGMOD Rec.*, 36(2):7–12, June 2007.

[32] TensorFlow, 2017.

[33] Twitter. Q4 2017: Selected company metric and financials, 2018.

[34] Twitter. Twitter, 2018.

[35] B. Twitter, 2018.

[36] L. van der Maaten and G. E. Hinton. Visualizing data using t-sne. 2008.

[37] M. Vicente, F. Batista, and J. P. Carvalho. Twitter gender classification using user unstructured information. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–7, Aug 2015.

[38] We are social. Digital in 2018: World's internet users pass the 4 billion mark, 2018.