POLITECNICO
MILANO 1863

School of Industrial and Information Engineering
Master of Science in Management Engineering

Information diffusion on Twitter during emergency crises: an analysis of replies

Supervisor: Prof. Barbara PERNICI

Master thesis of:
Gerard Fernández González   Matr. 876903

Academic Year 2017 – 2018

Abstract

This thesis is founded on the study of emergency management processes addressed to disasters. Specifically, it is focused on the sector that aims to collect information diffused through social media in order to improve decision-making regarding emergency response situations by humanitarian relief organizations.

The purpose of the thesis is to devise a classification method to automatically extract relevant replies of messages in social media, particularly in Twitter, in the context of natural disasters situations in order to enhance situational awareness of organizations with the information retrieved.

With the aim of evaluating the performance of this classification method, two cases studies are employed. Besides, the evaluation also allows assessing the degree of usefulness in terms of information derived from reply tweets for improving decision-making in emergency management processes.

Sommario

Questa tesi è fondata sullo studio del processo di gestione delle emergenze in riferimento ai disastri. Specificatamente è focalizzata sul settore che aiuta la raccolta di informazioni attraverso le reti sociali con lo scopo di migliorare il processo di decision-making da parte delle organizzazioni umanitarie riguardo la risposta a un'emergenza.

L'obbiettivo della tesi è quello di escogitare un metodo di classificazione per estrarre automaticamente messaggi rilevanti di risposta nei social media, in particolare in Twitter, nel contesto di situazioni di disastro naturale per accrescere lo stato di conoscenza delle organizzazioni con le informazioni raccolte.

Con il proposito di valutare le performance di questo metodo di classificazione, due case studies sono stati impiegati. Inoltre, la valutazione permette di stimare il grado di utilità in termini di informazioni derivanti dalle risposte di tweets per migliorare il processo di decision-making nella gestione delle emergenze.

Acknowledgements

I would like to thank my thesis tutor Prof. Barbara Pernici for her advices and comments, her availability to treat my doubts and questions, and specially for the opportunity she gave me to perform this work alongside her.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This thesis is founded on the study of emergency management processes addressed to disasters. Specifically, it is focused on the sector that aims to collect information diffused through social media in order to improve decision-making regarding emergency response situations by humanitarian relief organizations.

The work accomplished through this thesis is part of a H2020 European project E^2mC [1]Evolution of Emergency Copernicus services, which started in November 2016 to support early warning and rapid mapping[2] with information extracted from social media.

Information gathering by people directly facing a disaster through available sources such as friends, phone calls, official communications from administrations, radio and television and Internet to name a few, is typically performed. Analogous to people, humanitarian organizations try to absorb as much information about the event as possible, since effective planning involving difficult decisions must be rendered within tight time schedules.

Hence, with the aim of overcoming issues that could arise due to the non-availability of relevant information, social media emerges as an opportunity. Today, social media is strongly used by people affected by disasters not only to obtain information from it, but also to generate and share content. Social networks are now considered as key information channels, advantaging other sources mainly because of the immediacy and far-reaching scope in which their messages are distributed.

The gathering of social media messages allows the understanding by humanitarian organizations of the *big picture*, which deals with the creation of

[1] https://www.e2mc-project.eu/

[2] Concept introduced in Section 3.2.1.

high-level summaries that speak about the situation as a whole. The *big picture* concept comprises general estimations in terms of impacted areas, number of people affected and damage to infrastructures.

The conception of *big picture* is related to that of situational awareness, which aims to understand the causes and the resulting consequences that a particular disaster entails. Gaining situational awareness requires a complex process, which involves perceiving, comprehending, and being able to make predictions about the near future. Hence, the interaction of many actors combining various sources of information is essential to gain situation awareness.

Furthermore, social media messages can be classified according to their purpose, some of which can be employed in processes aiming to gain situation awareness. Consequently, it is recognized the existence of two types useful for these tasks: interpersonal communications and citizen sensing. The former can be exploited by organizations to learn, since a group of messages stating the same matter can be used to detect events or discover trends. The latter can be used as a means of information crowdsourcing, in which users act as sensors.

Nonetheless, the use of social media entail challenges to be addressed by organizations. The main three are presented hereunder.

Scale is considered the main challenge when acquiring social media data since it involves receiving, processing, and potentially storing a large number of items arriving at a rapid pace. Emergency managers often use the expression *information overload* to describe this issue, meaning that it is received more data than what they can handle. To overcome this situation, techniques aimed to construct effective predicates for the querying of messages have emerged as one of the major research's topics.

Social media messages tend to be brief and informal. Besides, they often encompass multiple sources, different levels of quality and grammatical correctness, and different languages. All these characteristics lead to the creation of ambiguous texts, which have to be subsequently treated by the use of Natural Language Processing (NPL) techniques.

The use of automatic text categorization by organizations address, at some extent, the problem originated due to the variety of messages by sorting of them into predefined categories. Consequently, messages are abstracted from the particular (a specific message) to the general (a class of messages).

In addition to the mentioned challenges, large organizations have expressed a number of reservations regarding the use of social media, despite perceiving its potential and utility as an information source.

The first reservation includes concerns about lack of personnel time that can be dedicated to monitor social media. The next common objection pertains to data quality, representiveness and veracity, and it is usually expressed as concerns about whether social media information can be trusted or not. Organizations also have reservations about how social media monitoring affects their operations in relation to technological barriers and expectations of the public.

Commonly, all efforts concerning the extraction of relevant information concerning the social network of Twitter are put in the creation of efficient keyword-based predicates based on the texts within messages. The underlying assumption of doing so is that it is believed that retrieved tweets according to the defined predicates will contain useful information either in the text, or/and in the media included (if the case) or/and in the URL provided (if the case) for its use in emergency situations.

A text usually is considered relevant in the case it contains novel descriptive information about the event taking place. This information could be then used by organizations to evaluate the condition of the situation as a basis to conduct their decisions. Besides, the reference to locations in texts is of a paramount importance to be able to frame the information to a specific area and becomes even more crucial in the case of tweets whose GPS geotag is not provided, as later explained.

Media enclosed in tweets through images or videos is also highly used by organizations as a potent means of descriptive information, and even are preferred than those messages just comprising descriptions through text because of their nature. Moreover, most of the times texts also are used to describe the included media, fact that adds more information to tweets into consideration. Hence, the retrieval of messages including media is most of the times considered as priority.

Inserted URL in tweets are used by users to expand, complement or just annotated the source regarding the information provided through the text. Nevertheless, in contrast with the two previous contents, URLs have been demonstrated of not providing any relevant information. This circumstance is due to the fact that most of the times URLs provided are attached to newspapers websites, whose publishing lapse is not immediate as social media posts are. Therefore, information originated from URLs tend to be non-novel, and consequently not relevant for rescue operations teams.

Furthermore, complementary information can also be extracted through users' profile, such as residence location and number of followers, and through the observation of tweets' parameters like number of retweets or likes.

A prior understanding of how information diffuses within the Twitter network is key to be subsequently able to extract it effectively. Hence, studies explaining and predicting how the information is spread, where the information is conveyed, and above all, which type of information is diffused, are constantly found in the literature, which also provide techniques for extraction processes through the optimization of precision and recall levels.

The first task of this thesis consisted of identifying an additional source to extract relevant information diffused from messages within Twitter, in order to exploit its usage. After several analyses, it was observed the convenience in considering replies originated from tweets, as a means of expanding the information provided by them.

Up to now, research studies analysing Twitter dynamics and characteristics had mainly focused towards the study of the single tweet. Nonetheless, no attempt had been yet done regarding the information that could be contained inside the replies of the tweets. Accordingly, since the study about the relevance of tweet's replies as a source of information was considered an opportunity of research, it was decided to focus the thesis towards it.

Hence, descriptive analyses studying the dynamics of networks originated by reply tweets (technically referred as tweet cascades[3]), along with their particular characteristics, are firstly performed in the thesis. Previous understanding about the manner in which replies are diffused, before addressing to their content, is primordial in order to carry out studies focused on the identification of patterns and the development of extraction information processes concerning replies.

Accordingly, in our case of study, the information to be extracted on Twitter from the replies must be related to natural disasters and useful for gaining awareness about the situation. In particular, to be considered relevant, the information extracted from the replies must address any of the following aspects:

- Add relevant descriptive information to the replied tweet.
- Help to contextualize the replied tweet.
- Provide the location, in a precise manner, corresponding to the replied tweet.

[3] Concept introduced in Section 4.1.

Regarding content locations, referring to those locations supplied in texts, they are considered as a key piece of information to be provided, inasmuch as the attachment of precise geographical coordinates to descriptions of events inserted in texts or/and media is useful for the management of crisis operations. Traditional natural language processing (NLP) techniques are mainly used to efficiently extract those locations in tweets.

The possible types of information conveyed in replies are multiple, although they could be summarized into informative, personal opinions and reactions, and questions. The first type entails relevant contents, while the latter two types entail irrelevant ones. Challenges concerning the automatic classification according to the relevance of replies are constantly highlighted throughout the dissertation of the thesis.

This thesis presents a new tool, to be run in Linux OS, able to extend the information already extracted from tweets through the addition of complementary information present in their replies.

The tool is run by the introduction of a database containing a set of tweets' ids as input, which are subsequently analysed and produces a list of ids corresponding to chains of reply tweets containing, a priori, relevant information.

Specifically, the proposed tool is based on the fact that, even if single messages could provide a high deal of useful information to be used in emergency situations, their corresponding replies contribute to enlarge that information and therefore overcome limitations due to the missing data in tweets.

The tool makes use of an automatic classification method of reply tweets, which take into consideration patterns found regarding the structure of the cascade, tweets' parameters values and their provided texts.

Consequently, the proposed tool is based on networks as well as on contents. On the one hand, it is network-based, inasmuch as posting users within cascades are analysed through their id in order to verify if replies are posted by the same user that originates these cascades, with the aim of considering them in subsequent extraction processes. On the other hand, it is also content-based, since it focuses on the analysis of the content in tweets' texts through unigrams and location identification processes to cluster them into categories.

Throughout the development of the tool, several challenges were arisen, making its design complex.

First of all, the motivation of users when replying to tweets deeply depends on the type of the event taking place. For instance, while floods tend to provoke the apparition of messages describing an ongoing situation including preventive purposes, posts regarding earthquakes usually expose the resulting effect of the phenomena. Besides, motivation is also directly affected by the time in which posts are created with regards to the event, since messages issued immediately after the occurrence of a disaster are more prone to provide alert content than those posted days after the finalisation of the event, whose content often deal with donations.

This motivational variety of users leads to the emergence of messages encompassing a huge diversity of text formats within same replies' types[4]. Heterogeneity of text formats is a factor that negatively affects the performance of the classification method, which needs of a great resilience level to be able to overcome such a difficulty.

Finally, locations mentioned in replies usually refer, extending its granularity, to previous ones provided in their parent tweets. In this manner, they often are supplied relatively, for example referring to "town centre", without spelling the explicit name of the city. This fact negatively influences the yield of location identification processes.

The performance and accuracy of the proposed classification method implemented in the tool is evaluated through two case studies at the end of the thesis. The case studies employed correspond to the floods occurred in the Southern England in February 2014 and the storms caused by Hurricane Harvey in Texas in August 2017.

The rest of this thesis is organized as follows:

- Chapter 2 deals with the state of the art, which presents the main findings based on information diffusion reported in research studies, describes the most effectively developed techniques used to analyse tweets, and introduces the processes currently used aiming to classify tweets according to their text.

- Chapter 3 presents the programme in which the thesis is framed (Copernicus), and in particular its application within the Emergency Management Service, that is rapid mapping. Furthermore, it provides a description of the case studies to be considered throughout the thesis.

- Chapter 4 illustrates the methodology followed to collect all the necessary data for performing subsequent analyses. Processes encompassing the

[4] Replies include informative, reactions and personal opinions, and questions types.

collection of tweets and the manual annotation of them are therefore described in this chapter. Moreover, it is introduced the concept of tweets' cascades.

- Chapter 5 comprises the set of analyses addressed to understand the behaviour of tweets originated in natural disaster contexts from a descriptive point of view, considering the structure of their cascades and the relationship between these structures and different tweets' parameters. The main findings derived from the analysis of tables and figures are described at the end of each section.

- Chapter 6 provides a set of analyses focused on gaining insight on the relationship between the relevance of reply tweets (and its corresponding parameters), originated in contexts of natural disasters, and the structure of cascades. Findings derived from these analyses that are considered convenient to be used as patterns aiming to classify reply tweets according to their relevance, are also described in this chapter.

- Chapter 7 proposes a classification method of reply tweets according to their relevance, using the patterns identified in the previous chapter.

- Chapter 8 presents the tool, which implements the classification method proposed in the previous chapter, to be used to automatically classify reply tweets according to their relevance. The main processes that the tool execute, along with a description of its interface, are described. Furthermore, it contains a performance evaluation of the tool through two cases studies.

- Chapter 9 offers the conclusion of the thesis and presents its related future work.

Chapter 2

State of the art

Social media, and in particular Twitter, encompasses characteristics that are of a completely different nature than other networks such as websites or Wikipedia. Therefore, an exhaustive analysis about social media comprising research studies is provided in this chapter, with the objective of understanding the behaviour and features of such networks and their corresponding messages.

This chapter includes the two main topics that will be subsequently used throughout the development of the thesis: information diffusion and tweet's processing.

From an emergency response perspective, spreading of information is important to be understood in order to propagate messages with preventive purposes or effectively collect the information for enhancing the decision-making in operations. Therefore, in this chapter several findings are presented regarding explanatory and predictive models of information diffusion.

Moreover, the learning of how tweets can be efficiently extracted and classified according to their relevance in specific contexts, and the treatment in which they have to be submitted with the aim of obtaining quality information, entail a paramount importance and therefore are also studied in this chapter.

In the following sections, first it is discussed the behaviour of information in terms of diffusion in Section 2.1. In Section 2.2, the main techniques used to analyse tweets, and in particular their text, are described. Finally, in Section 2.3 the processes that tweets undergo to be classified are illustrated.

2.1. Information diffusion

Information is something that informs, or, in other words, an answer to a question of some kind and it is usually conveyed as a content of a message.

Nowadays, massive amount of information is transferred every day in social media networks between users. Social media networks have grown up spectacularly since 2010 and therefore information within text, photos and videos is spread easily and effectively through them. Information diffusion arises as the most valuable aspect of social networks, since by knowing patterns and behaviour of information spreading process could predict the success, popularity, and favourability of various events, persons and opinions.

Lots of researches have been conducted lately in the field of information diffusion. Results of such researches would help organizations to better understand the spreading process of the information, and hence to optimize business performance or to solve issues, among others. However, challenges such as real-time changes of the networks and the complexity of social interactions increase the difficulty to provide a specific mechanism of the spreading process of the information. As a matter of example, some of the most significant factors to model the diffusion are the sentiment of the meme, topic of discussion, the network structure of the user, the physical location of the user and the presence of some influential users in a particular topic.

In mass emergency situations during disasters, rapid information diffusion becomes crucial since gathering and sharing timely information regarding infrastructure, supply of resources and needs, is critical to develop an understanding of existing conditions and coordinate an effective response.

In the present researches, there are two major categories of models for information diffusion modelling: explanatory models and predictive models. The former aims to retrace the spreading path of the information; the latter objective is to predict how a specific diffusion process would unfold in a given social network, based on past results of information diffusion researches.

2.1.1. Explanatory models

As previously explained, our study is focused on the social network Twitter. It is found that part of the research has been focused on modelling the diffusion of information contained in tweets by their retweet rate and how retweets propagate among users, which is related to the explanatory model category.

In (Kawamoto, 2013), it is introduced a stochastic model for the diffusion of tweets by retweets. It is believed that there might exist a simple underlying mechanism to describe the behaviour of tweet diffusion statistically, even if each process of the retweet would depend on the specific details of the user and the characteristics of the original tweet. To achieve that, they decomposed the diffusion of daily tweets into dynamics along the generations of followers (user generation contains all the followers from the user who retweet). Consequently, whenever a user generates a tweet, it will be sent to N_0 followers of the tweet owner, whom are called users in the zeroth generation. Next, n_1 users out of N_0 followers will retweet the original tweet and sent to the followers of n_1, called first generation users. Such a chain of diffusion of a tweet continues until people stop retweeting or all the followers in the last generation are users who have already received the tweet. Besides, the numbers of viewers of the tweet in each generation is expressed as a random multiplicative process, since it is not possible to measure them. Results confirmed that the proposed model is indeed plausible, inasmuch as they found that the multiplicative factors roughly obey lognormal distributions and that diffusion occurs owing to the repetition of cooperative activities along the followers, as directly observed from the actual data of Twitter.

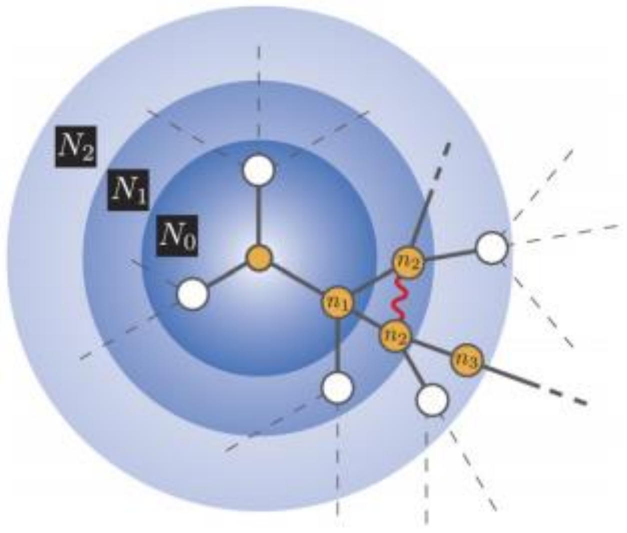


Figure 1: Diffusion network on Twitter. The node at the centre represents the seed account and the linked nodes are the followers. A solid line means that the tweet has diffused through the link by a retweet. Source: Kawamoto, 2013.

A different analysis approach was used in (Zhou et al., 2017). There, a multi-level structure was defined to describe the information diffusion evolution of a popular topic (hot topics) through retweeting networks, containing users tweeting about the topic and retweeting links between them, and where retweeting groups and information cascades grow and interact. Consequently, three important features in the evolution of retweeting network are found: (1) merging effect, meaning retweeting groups that merge with other groups by new retweeting links during the diffusion process; (2) super group phenomenon, which is the largest retweeting group with over 30% users in the retweeting network, and attracts newcomers and merges other groups by the influential users; and (3) centralized topology, which are types of topologies of information cascades that consist of star and multi-centre star topology, but rarely long chain topology in the retweeting network. It indicates that there could be bursting diffusions around influential users, but rarely long-distance diffusions through ordinary users. Furthermore, to find out the most influential users, they are ranked in descending order by number of followers, number of followees and number of tweets.

Results obtained show that during the evolution of each hot topic, a giant group is formed during the diffusion process. Consequently, at the end of the evolution shows a clear polarization: one super group and many small groups. The super group contains about one-third of the users and becomes the diffusion centre of the retweeting network. Besides, users activity is monitored. Users in many traditional online social networks rarely post repeatedly on the same topic. However, users in microblogging network often tweet more than once on the same topic, since over 20% of users participate in at least 2 tweets and 10% in 3 tweets. In addition, the study proves that the trend of the diffusion in the future is influenced by the past. Also, that users participating in many information cascades, and therefore connect them, are usually active users in the microblogging networks; and that the number of retweets caused by users has strong correlation with their number of followers, indicating that popular users often bring a large number of retweets in specific topics.

Other research studies focus specifically on evaluate how the information diffusion develop during humanitarian crises, which constitutes our topic of investigation. Connections in social media networks create cascades in which users share content with those with whom they are connected. In (Yoo et al., 2016), the study of the dynamics of information dissemination during these events and their influence on humanitarian operations is addressed by the test of theoretical propositions regarding the role played by the three key determinants of information diffusion:

- *influence* of the user that originates the information cascade in the network as a function of their social connections;
- *type of content* being shared in these networks and whether it contributes to improving situational awareness during a crisis;
- and *timing* in the introduction of information in these networks with respect to the progression of disaster events.

The results of the analysis show that, in this context, cascades on social media networks can advance at a rate that significantly exceeds the speed at which information originates from external sources. Information that is originally posted later, as a disaster intensifies, spread at a lower rate since participation declines over time. Another contribution shown is that information issued by users with high levels of influence diffuses quickly, as previously noted in (Zhou et al., 2017). Also, that cascade originators may be able to increase the speed of diffusion by posting the same information repeatedly in order to raise its visibility. Moreover, it is observed that cascades with fabricated information infect the network at a faster pace and that there is no evidence to assert that cascades carrying content that enhances situational awareness exhibit significantly higher diffusion rates. Furthermore, another pattern identified is that local individuals are more likely to contribute to information and propagate it during humanitarian crises than other individuals.

In (Feng et al., 2015), an analysis of how popular message spread following a certain mechanism is provided. To model this spread of information, it is used the susceptible infected recovered (SIR) model of disease epidemics. It is checked that most of the spreading occurs within the first day of posting, indicating a decaying rate of diffusion over time. It is also found that highly connected nodes are less likely to pass on incoming information, as they need more repeated signals before sharing a message since they are exposed to an overload of messages less likely to view, remember or diffuse. Furthermore, evidence is found to assure that information overload also shortens the visibility duration for popular messages.

Another paper dealing with explaining how information is diffused is (Fowler et al., 2010). Specifically, it provides an exhaustive study on how particular behaviours can create cascades of similar cooperative or uncooperative behaviours in others, spreading from person to person to person, even when reputations are unknown, and reciprocity is not possible. Such a cascade would therefore suggest that social contagion also may play an important role in the evolution of cooperation. Conclusions assert that the fundamental justification for the existence of elaborate ties in social networks may be that these ties may allow humans to spread beneficial messages to benefit others.

Alternatively to above-mentioned analysis of information diffusion during mass emergencies, other papers investigate the usefulness of this information for humanitarian operations to assemble an accurate picture of the situational context across the region of interest. In (Saleem et al., 2014), an investigation of novel situational information (NSI) provided by Twitter posts during disasters is conducted. A relevant finding is that many of the tweets that first reported actionable situational awareness information did not include keywords or hashtags that would have made their discovery through standard Twitter filters possible. Also, the analysis suggests a number of factors that influence the apparition of NSI:

- Danger: People is not willing to post purely informational content when their immediate safety is threatened.

- Emotional distance: Casualties (injuries and deaths) are only reported by those who achieve sufficient emotional distance from the victims.

- Mobility: Populations that have limited mobility provide less coverage of key points of interest.

- Gradual disaster: Damage that changes over time (e.g., floods) is difficult to report in a systematic and interpretable way.

- Diffuse damage: Disasters characterized by homogenous damage are difficult for social media to accurately report on.

To sum up, some of the findings presented in articles regarding the diffusion by retweets are that diffusion occurs because of the followers' repetition of cooperative activities, that the apparition of long-distance diffusion through non-influential users is not usual, and that the trend of the diffusion in the future is influenced by the past. Furthermore, with regards of the speed of diffusion it is found that social media cascades advances at a major rate than information originated from other sources, that the fact of posting the same information several times increases diffusion's speed, that cascades with fabrication information infect the network at a faster pace, and that most of the spreading occurs within the first day of posting.

2.1.2. Predictive models

Much fewer studies corresponding to predictive models are found in the literature compared to the explanatory ones.

In (Yang et al., 2010), taking as motivation the idea that active interaction network is of higher value than the follower network with respect to analyses of information diffusion, it is presented results of network analyses via users' ongoing social interactions as denoted by mentions. In particular, it is constructed a novel model to capture the three major properties to predict information diffusion from the tweets themselves: speed, whether and when the first diffusion instance will take place; scale, the number of affected instances at the first degree; and range, how far the diffusion chain can continue on in depth. Taking together it is seen a clear theme that the mention rate of the person tweeting is a strong predictor of all aspects of information diffusion in Twitter.

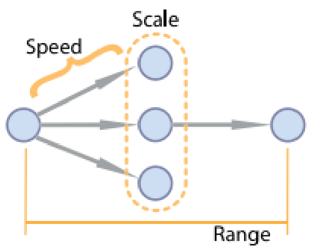


Figure 2: Three measures of local diffusion tree. Source: Yang et al., 2010.

An article which verifies the hypothesis that information diffusion influences link creation between users is exploited in (Li et al., 2016). In social networks, link prediction is a critical task that plays an essential role in the whole network growth. It can be applied in many fields including user recommendation, community detection, network growth modelling and so on. The hypothesis explaining that when one user observes a piece of information released by an unrelated user, he may be interested in the content or the user releasing the information and then may try to create a new social relation with the unrelated user is finally confirmed. Hence, they assert that the diffusion feature of observation number, meaning the number of observation of posts by an unrelated user from one user, is definitely helpful in link prediction task.

An analogous approach is followed in (Antoniades et al., 2015). There, the authors focus on the co-evolutionary dynamics in Twitter, in particular in the relationships building among users. The studied situation is called Tweet-Retweet-Follow (TRF) and is explained like this: being R an S follower and L an R follower, the probability of L to start following S in the case that R choose to propagate an S tweet to his own followers (so then L).

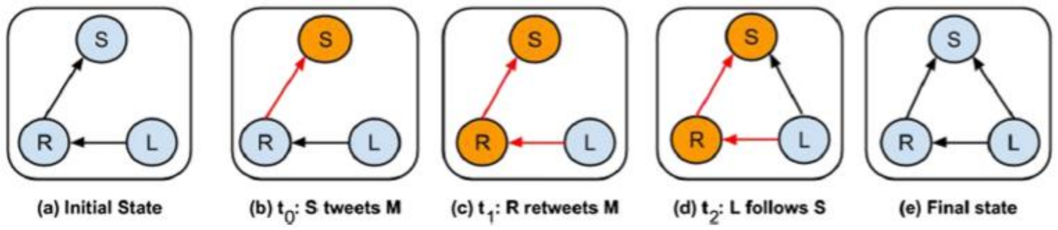


Figure 3: Network co-evolution: A Tweet-Retweet-Follow event. Source: Antoniades et al., 2015.

In the study, it is compared the likelihood with which a user gains new followers when there are no recent retweets of his messages compared to the case that he gains new followers when at least one of his messages has been recently retweeted. It is shown that it is much more likely for a user to get a new follower if his tweets are retweeted than if they are not, and that TRF events occur in practice and they are responsible for a significant fraction (about 20%) of the new edges in Twitter. Also, that more than 80% of TRF events occur in less than 24h after the corresponding retweet. The main factors that affect the probability of a TRF event are reciprocity and the total number of retweets received by the Listener (user L). Furthermore, users may also unfollow another based on the twitting activity of the latter. It is showed that 60% of the unfollow events occur during the first hour of posting some content by the Speaker (user S).

To sum up, in terms of the propagation of information it is found that the mention rate of a user is a strong predictor of the speed, scale and range that a tweet will undergo. Also, in terms of followers' link creation it is presented that the number of posts of a user observed by an unrelated user is a predictive factor, and that is more likely for a user to get a new follower in the case his/her tweets are retweeted than if they are not.

2.2. Tweet analysis

Tweet text is the place that most of the times convey the information that a user wants to share within the social media, since although photos, videos and/or embedded links in the tweet could provide also information, text is generally used to describe them. This fact makes that text analysis in social networks posts has a paramount importance when automatic relevant classification is necessary to perform, mainly due to the great amount of data in, for example, Twitter.

In the literature it is found numerous papers and research studies regarding the analysis of texts from posts in social media networks. Having established a topic of interest, the ultimate target is to select efficiently and effectively a set of tweets that are task relevant using their textual content. The main Natural Language Processing (NLP) techniques for tweets analysis are information extraction, geolocation, automatic summarization, semantic enrichment and supervised classification.

Information extraction is the process of identifying within text instances of specified classes of entities with the objective of transforming unstructured and noisy tweets into structured information. These specified classes of instances have to be established in advance and are related to the topic of interest. The main usefulness to perform information extraction in tweet texts is that it makes the information more accessible for further processing.

However, the increasing diversity of languages used on Twitter introduces a new level of complexity to Information Extraction systems. Natural Language Processing (NPL) tools are limited to a small number of languages, usually only English. In (Al-Rfou et al., 2015), it is demonstrated how to build massive multilingual, for 40 major languages, Named Entity Recognition (NER) annotators with minimal human expertise and intervention using Wikipedia and Freebase. The method learns distributed word representations (word embeddings) which encode semantic and syntactic features of words in each language, by relying on language-independent techniques. Finally, its performance is demonstrated by a comparative analysis using machine translation, with highly-consistent results.

Geolocation is the activity of associating a location to the messages using the text as indicator. Studies have demonstrated the value of extracting the locations referenced in the text, firstly recognizing the toponyms mentioned and then disambiguating them to the exact locations they refer to.

Location information is not only a valuable attribute to better analyse political, economic or social trends, but is critical to understanding the impact of a disaster during mass emergencies situations, including where the damage is, where people need assistance and where help is available. Hence, tweets content allows to estimate or/and precisely identify locations by attaching geographical terms to emphasize the description about an event.

The main challenge of extracting locations from the text in Twitter posts is based on the specific nature of these texts compared to traditional ones, which contain limited characters (short), are noisy and generally are decontextualized. Identification problems mainly derive from the quality of the content, as users often use shorthand and non-standard vocabulary for informal distribution, and the fact that users do not always introduce the obvious location names. Hence, imprecisions come on account of ambiguities that exist between location names and common names and among location names themselves.

Some articles in the literature treat geolocation and attempt to gain insight on how locations could be better estimated. In (Ao et al., 2014), a method is developed to improve the accuracy of event location estimation. To do so, they denote three kinds of location for each post: (1) content-based location, the location provided in texts (geolocation); (2) posting location, location posted by the GPS of the mobile phone; and (3) registration location, that is the location entered in user profiles. Euclidean Distance is adopted to measure the error distance, which quantifies the distance between the estimated and actual locations. It is verified that the approach only concerning registration location in messages are the least relevant because of its largest Average Euclidean Distance (5.708). Also, that the value from content location (0.227) is dramatically lower than the one from posting location (3.209). Finally, the proposed algorithm effectively locates events by combing these three kinds of locations with an average error distance of 0,224.

Another paper describing a scenario for rapid mapping in an emergency event (Francalanci et al., 2017), points out the problematic derived from the imprecision of the locations mentioned in tweets since they could be too general, citing for example a city or a region, even if the location is effectively identified. Furthermore, it stresses the relevance of the gazetter (a geographical dictionary or directory used in conjunction with a map or atlas) used, since it could not cover equally all the target location and could contain errors. Gazetters such as GeoNames and OpenStreetMap are used in the case studies presented, including a brief comparison between them. Note that the principal objective of gazetters is not just to provide information on named features, but to translate between informal and formal systems of places.

A new approach to enhance information extraction from social media that relies upon the geographical relations between twitter data and flood phenomena is analysed in (Herfort et al., 2014). The authors apply a geographical focus to prioritize crisis-relevant information from social networks. Flood phenomena such as hydrological features and models of terrain and affected areas are considered in the model since they are generally valid for every flood scenario. The conclusion explains that the locations of flood-related twitter messages and flood-affected catchments match to a certain extend. In particular this means that mostly people in regions affected by the flooding posted twitter messages referring to floods. This is considered as remarkable since there are far more tweets posted in greater distance to flood-affected regions compared to the ones posted in the proximity of floods.

In (Xu et al., 2016), a participatory sensing-based model for mining spatial information of urban emergency events is introduced. Hence, real spatial information is determined based on locations in messages and GIS (Geographic Information System) information. Then, the study explores data mining and semantic analysis methods to obtain valuable information on public opinion and requirements. With the geo-tagged and time-tagged data, the collected tweets can be classified into different categories, while public opinion and requirements can be obtained from the spatial and temporal perspectives to enhance situation awareness.

Automatic summarization is the task of shortening a text document, in our case a set of tweets, in order to create a summary including the major points of the original document. When applied to Twitter, the value would come from extracting relevant representative tweets from a time-ordered sample of tweets to generate a coherent and concise summary of an event. Even if the easiest way to extract tweets related to an event is through a search query, it results in a significantly large stream of tweets for popular events containing fairly limited information. This fact highlights the need for efficient methods to select the tweets which carry the relevant information. Nevertheless, even if the task has been presented because of its relevance as technique the analysis of texts, this thesis will not use it in their subsequent applications.

Semantic enrichment is the process of connecting different text expressions which refer to the same semantic concept with the aim of clustering tweets by their meaning. Learning and modelling the semantics of individual Twitter activities is important because the number of tweets published every day is continuously growing so that users need support to benefit from Twitter information streams.

Entity linking task for tweets deal with semantic enrichment since it attempts to map each entity mention in a tweet to a unique entity, that is the meaning we would like to extract. This task is generally considered as a bridge between unstructured text and structured machine-readable knowledge base and represents a critical role in machine reading program. Entity linking for tweets is particularly useful, considering that tweets are often written in an informal style and its length limitation. Existing entity work can roughly be divided into two categories. Methods of the first category resolve one mention at each time; in contrast, methods of the second category take a set of related mentions as input and figure out their corresponding entities simultaneously. A method corresponding to this second category is proposed in (Liu et al., 2013). The method is believed to work effectively on tweets which are short and often noisy, and it integrates mention-entity similarity, entity-entity similarity, and mention-mention similarity to address the information lack in a tweet as a distinguished characteristic.

As entity linking, Latent Dirichlet Allocation (LDA) is an unsupervised, self-learning topic modelling approach used to analyse the textual content of social media posts. This method allows the extraction of latent topics from the text corpus without the necessity for a priori knowledge about the event, which is required with most previous approaches, particularly keyword-based ones. The extracted topics can then be interpreted with respect to their relevance to, in our case, a natural disaster. The number of topics strongly influences the topic-word distributions in terms of the granularity of the topics: higher number of topics leads to smaller topics, while lower number of topics leads to larger topics. An example where LDA approach is applied to assess the damage caused by natural disasters with spatial and temporal analysis is presented in (Resch et al., 2017). Conclusions point out the convenience of using such method in emergency situations since priori knowledge is not required and events may not be characterized through afore known keywords.

Supervised classification is the task of automatically assign text documents, tweets in our case of study, to pre-defined classes, in our case specific events. Generically speaking, the common approach to building a text classifier is to manually label some set of documents to pre-defined categories or classes, and then use a learning algorithm to produce a classifier. This classifier can then assign classes to future documents based on the words they contain. The main bottleneck of building such a classifier is that a large number of labelled training documents is needed to build accurate classifiers.

In (Castillo, 2016), a presentation of the evaluation metrics used for supervised classifiers is introduced. Classification is usually measured in terms of efficiency and effectiveness. Accuracy is probably the simpler metric for classification effectiveness, since it corresponds to the probability that an item is classified correctly. Classification accuracies reported in the literature of social media during crises range from 60% to 90%. Precision is a measure of specificity, since it corresponds to the probability that an item that have been classified to a class, actually belongs to that class. Recall is a measure of sensitivity, since it corresponds to the probability that an item that actually belongs to a class is classified to another class. Furthermore, a popular metric combining both is the F-value, which can be defined as the performance of the classifier.

$$F - value = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

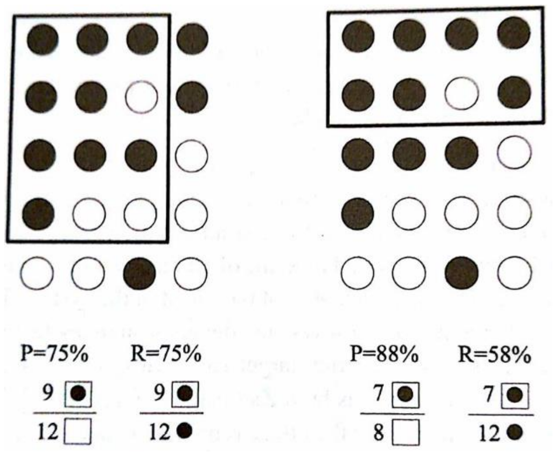


Figure 4: Two examples of precision (P) and recall (R) calculation. Messages are represented by circles. Messages classified are inside rectangles. Messages relevant are filled in black; irrelevant in white. Source: Castillo, 2016.

Up to now, the most outstanding research contributions around supervised classification on Twitter are based on machine learning algorithms such as Naïve Bayes, Maximum Entropy and Support Vector Machines (SVM). Even though at the end, in this thesis, no algorithms are used to classify tweets, it is provided a brief discussion about the machines learning algorithms presented in (Khan et al., 2010).

In (Khan et al., 2010), K-nearest neighbour (k-NN) algorithm is considered valuable for its simplicity, but it is computationally intensive, and its accuracy is severely degraded by the presence of noisy or irrelevant features, as the case of tweets. Naïve Bayes classifier is advantageous since it requires a small amount of training data to estimate the parameters necessary for classification. However, the main disadvantage of this algorithm is its relatively low classification performance compared to other discriminative algorithms such as the SVM with its outperformed classification effectiveness.

Support vector machines (SVMs) is therefore one of the discriminative classification methods which are commonly recognized to be more accurate. The SVM method need both positive and negative training set which are uncommon for other approaches. The document representatives which are closest to the decision surface are called the support vector. Hence, the performance of the classification remains unchanged if documents that do not belong to the support vectors are removed from the set of training data. The SVM classification method is outstanding from others for its effectiveness. Furthermore, it can cull out most of the irrelevant features. However, the major drawback of the SVM is their relatively complex training, especially problematic when emergency situations arise, and rapid response is needed.

An automatic method for extracting information from microblog posts using supervised classification is described in (Imran et al., 2013). Specifically, they focus on extracting *information nuggets* which are brief, self-contained information items relevant to disaster response. The proposed system needs to detect messages that may add situational awareness information. To this end, four categories of messages were considered: personal only, informative direct, informative indirect and other. Manual classification process for the labelling of tweets is carried out by crowdsourcing workers using the CrowdFlower platform. Only informative messages were selected to undergo the next step which deals to cluster them to another four classes depending on their context: caution and advice, causalities and damage, donations and information source. Furthermore, various types of information were extracted from these classes. As example, in caution and advice nuggets there was extracted location and time references, the caution message, source and type of caution. Finally, a set of multi-label classifiers were trained to automatically classify a tweet into these mentioned classes using Naïve Bayesian approach, employing a number of binary, scalar and test features. The binary features consisted of whether a tweet contains or not @ symbols, URL, hashtags, emoticons, and numbers. The scalar features only consisted of a numeric feature indicating the tweet length. The text features consisted of sparse linguistic features along with Twitter specific stopwords: unigrams, bigrams, Part of Speech (POS) tags and Verbnet, an ontology for verbs. The results showed that this approach allows to extract structured information nuggets from unstructured posts with good, but not excellent, precision and recall.

In (Imran et al., 2016), it is presented the publication of a Twitter corpus consisting of more than 52 million crisis-related messages collected during 19 different crises is presented to address the basic necessity of human-annotated data of creating domain adaptations (fully explained in Section 2.3) when dealing with supervised classification. Hence, human annotations from volunteers and crowd-sourced workers are of two types: first, the tweets are annotated to identify a set of categories; second, the tweets are annotated to identify out-of-vocabulary (OOV) terms and their corrections and normalized forms since the accuracy of NLP techniques would largely improve if we can identify the informal nature of the language in tweets and eventually be useful for humanitarian organizations. These lexical variations are divided into typos/misspellings, single-word abbreviation/slangs, multi-word abbreviation/slangs, phonetics substitutions and words without spaces.

Event detection in emergency contexts is another relevant topic of research that attempts to minimize the response time in which rescue teams can initiate to operate their activities. For this reason, even if it not directly treated in this thesis, an overview introducing two papers has been convenient to be provided in this chapter.

Hence, (Laylavi et al., 2017) introduces a novel method for detecting event-specific and informative tweets that are likely to be beneficial for emergency response in near real time, since tweet relatedness to events evaluation is believed to be necessary along with assessment of its informative quality to be used by emergency service practitioners. The method is characterized by using term-classes, a set of event-specific words of comparable frequency which are defined from the extraction of common patterns. The performance of this method outperforms other machine learning approaches and is considered as a noticeable improvement on identification tweets techniques.

In (Sakaki et al., 2013), an algorithm is proposed to monitor tweets and to detect a target event at real-time. To detect a target event, a classifier of tweets is devised based on features such as the keywords in the text, the number of words, and their content. To classify a tweet as a positive class or a negative class, it is used a support vector machine, so that three groups of features for each tweet are prepared: features A (statistical) consider the number of words in a tweet text and the position of the query word within the tweet; features B (keyword) take into consideration the words in a tweet; and features C (word content) highlight the words before and after the query word. Performance results showed that the highest values of recall and precision comes from features A, and that features B and C do not contribute much to the classification, in the case of earthquake occurrence and using the keywords "earthquake" and "shaking" as queries.

To sum up, in terms of geolocation it is presented a new algorithm that efficiently locates events by combining the three types of locations registered in Twitter, and that situational awareness can be enhanced by considering geo-tagged data along with time-tagged data. With regards to supervised classification, it is stressed that support vector machines (SVMs) is one of the most accurate classification methods. Furthermore, it is found that the approach of classifying messages into several categories and subcategories according to their text, and finally selecting a set of these subcategories allows to extract structured information from unstructured posts with good performance. With regards to event detection, it is found that keywords in tweet's texts feature is the one which achieve highest values of recall and precision, and that the term-classes method outperforms other machine learning approaches for the identification of tweets.

2.3. Tweet text processing

After presenting in the previous section (Section 2.2) the main techniques used by researchers to analyse documents, and in particular texts in social networks posts during natural disaster events, it is proceeded with the description of the overall process carried out with the aim of classifying tweets.

- Data collection: Due to the openness of Twitter, it is possible to have access to publicly available tweets and retrieve them with the associated metadata in response to specific queries performed through the Twitter Application Programming Interface (API). The fact that the default setting for the visibility of Twitter profiles is "public" causes than less

than 10% of users choose to change to "private" setting, and therefore making the vast majority of tweets retrievable.
There are three main types of APIs provided by Twitter, from which Streaming API emerges as the preferred one to monitor tweets in real-time, as it is needed in emergency responses. Different criteria, for example in terms of keywords, location boxes, creation time or user characteristics, are then introduced in the queries to retrieve data.

- Data sampling and cleaning: Tweet sampling is the process of selecting a reasonable subset of tweets from a dataset in order to investigate the characteristics of the entire dataset. The sampling process is performed in different filtering steps, which vary depending on the ultimate analysis target. Common filtering measures take into consideration the language of tweets and the spambot posts elimination, this latter usually made by computing the ratio between number of followers and number of followers from the user.
  Data cleaning is the pre-processing step necessary to achieve uniform textual contents. Its application to Twitter data is essential, since tweets are extremely prone to different types of noise and redundancy. For instance, there are huge numbers of emoticons, user mentions and Internet links within the text field which may negatively influence the performance of analysis. However, this process also brings negative impacts since in some cases part of the data removed/changed could have provided some kind of useful information. Common cleaning tasks deal by removing multi-dots, hashtags signs, numbers or links as well as lowercase conversions and the merging of multi-spaces. A complete overview of all the possible cleaning functions is presented in (Castillo, 2016).

- Domain adaptation creation: As previously noted, supervised learning algorithms require labelled data to learn accurate classifiers. However, in the case of emergency situations and specifically when a new disaster arises, labelled tweets are not available as each event is unique and therefore new labelled data must be created, and therefore time-consuming manual annotation is usually required. This fact causes a great challenge since real-time actuation to help rescue operations is required. Hence, partial utilisation of labelled data from prior disasters is common to guarantee a rapid response.

- Features extraction: The pre-processed data have to be transformed into a vector format for further calculation steps and before the classification task. This vector improves the scalability, efficiency and accuracy of the classifier and must provide specific features of the tweet text, which have to be decided according to the type of information that is needed to select. The selected features retain original physical meaning and provide a better understanding for the data and the learning process. Examples of features could be statistical focused such as the number of total words in a tweet, content related like the number of appearances that a specific word appears in a tweet, or morphologic as n-grams, which is the contiguous sequence of n items from a tweet text.
  Feature selection is therefore an obligatory step to perform just before the classification process, either making use of non-supervised or supervised methods, starts.

- Classification: This task deals with the division from the original set of tweets into these estimated to be useful and the ones that not, so that clustering tweets by their usefulness. The main techniques used to classify tweets have been already described and analysed in Section 2.2. In order to know if a tweet is candidate to be useful, as noted before, it is necessary to select a set of features to extract from tweets and, in the case of supervised classification, to create a domain adaptation for machine learning.

- Analysis and conclusions: Once the retrieval of theoretical useful tweets finish, it is possible to initiate the data analyses, which are varied and can be numerous since they largely depend on the study that it is being carried out.
  After the analysis, some conclusions are reached. In the case of natural disasters, common targets are the identification of damage zones, the indication of suggested actions to be done by rescue teams or severity assessment of the event.

In the literature, many papers describe the overall process for the specific case they are treating.

For example, in (Laylavi et al., 2017) data collection is performed using Tweepy, a Python-based library that interacts with the Twitter API server and it is used as filters location (bounding box) and creation time of tweets (time window) as parameters. Filters such as text language, spambots posts elimination and source (web, tablet and mobile phone) are applied in the sampling step. Cleaning is done by removing retweets and other elements from the text like non-ascii characters. Human annotation by experts to identify informative tweets within the set to train the model is carried out just before running the supervised classification, that use term-classes as feature.

Similarly to (Laylavi et al., 2017), in (Resch et al., 2017) the collection of tweets is realized by filtering the location and the time window. Noise reduction is sequentially done performing tokenization, lowercasing the tokens, removing URLs, numbers, special characters, short words, stop words, unique words, later by handling synonyms and finally stemming. This pre-processed data is then transformed into a vector to apply LDA method. Both spatial and hot spot analysis is carried out afterwards.

A simpler approach is used in (Francalanci et al., 2017), on which tweet selection is accomplished using keywords in different languages. The cleaning task encompass the elimination of all retweets and posts not written in Italian and the exclusion of tweets from users that were not likely to be present in the event. Furthermore, geolocation based on the tweet text is performed by searching for specific keywords using entity recognition for locations with the aim of extracting the geographical coordinates of the described place. However, no method is used to automatically classify the tweets leading to no need to establish any feature and create the domain adaptation. Image and geolocation analyses are finally carried out to validate the proposed model. Due to the straightforward nature of the process, the accuracy of results is not as high as the two previous explained cases.

An article seeking for the enhancing of data collection methods is (Fresno et al., 2015). Since typical information filtering techniques are keyword-based approaches which entail low recall level if message not contain an initially considered event-related keyword, it is delved into the use of three additional features of tweets, namely *user*, *geolocation* and *temporal* information on which the authors rely on to discover new keywords which are related with the natural hazard. The idea is based on two hypotheses: if a user posted a tweet about an event in an affected area, it is expected that their immediately previous or later messages will be related with the event; if a tweet about a natural hazard in a specific geolocation and time is found, it is expected that tweets within a nearby geolocation and posted at the same time will be also related to the event. Pseudo-Relevance Feedback (PRF) is used on the method since it has been proven to be effective in many information retrieval tasks, and then it is re-examined the PRF assumption considering the social information stated in the hypotheses. In the paper also is introduced a Temporal-KLD for considering the Twitter temporal aspect within the PRF process. Results show that especially the use of temporal information can have a significant impact in the extraction performance, improving recall values. Moreover, the use of social information for query expansion so as to discover new keywords related to an event to help boosting the performance of the tweet retrieval is confirmed.

In (Fohringer et al., 2015), there is a proposed methodology that leverage social media content to support rapid inundation mapping. A tool called *PostDistiller* is developed, which combines various filtering approaches with regard to selective contextual information reduction and visualization methods. A visual interface facilitates the exploration of filtered posts with the purpose of deriving specific quantitative or qualitative data, which consist of four components that allow to filter posts/photos, depict single posts/photos and attached information, provide a map with the posts location and to store extracted information in the database. The tool consists of three components: (1) PostCrawler, for the retrieval of posts; (2) PostStorage, for persistent storage; and (3) PostExplorer, for the exploration and extraction of information from single posts.

Another study that deals with visual approaches is (MacEachren et al., 2011). They implement a geovisual analytics application, called SensePlace2, focused on place-time-attribute based information foraging and visually-enabled sense-making in support of crisis management. Hence, the tweets retrieved are just those that include geographic location in any way, either inferred through specific hashtags or by automated entity extraction methods. Locations that are extracted are then georeferenced using GeoNames. SensePlace2 supports overview and detail maps of tweets, filtering of tweets, and analysis of changing issues and perspectives over time and across space as reflected in tweets. The purpose of the interface is to support an understanding of spatial and temporal patterns of events through analysis of geo-located Twitter database. It contains four views: query window, map, time-plot/control and the task list, which allow users to label results of a query and store them.

Chapter 3

Description of the case study

In the previous chapter it has been demonstrated that social media adopt different behaviours compared to other sources of information consequence of the immediacy and volume of their messages. Consequently, more and more social networks posts have become an opportunity to extract data, and therefore are being used as a source of crowdsourced information in a huge number of applications.

The application in which the thesis' case study is based on is related to humanitarian crises situations. Hence, the importance that information extracted from social networks take in these kinds of contexts, and the manner in which information for emergency response activities during natural disasters is used, are respectively highlighted and discussed in this chapter.

This chapter also deals with the presentation of Copernicus, the European programme supporting humanitarian crisis situations. In a nutshell, Copernicus is a programme aimed to develop information services. The case studied is found within the project Evolution of Emergency Copernicus services (E^{2}mC), which goal is to extend the support for the activities of the existing Copernicus Management Service (EMS), providing the rapid mapping to operators using additional information derived from social networks.

The presentation of the case studies in which the thesis is based on, along with a brief exposition of its related Copernicus service activation, is also introduced in this chapter.

Therefore, in the following sections, first it is introduced the Copernicus programme, commenting its target, mission and applications, in Section 3.1. In Section 3.2 the Emergency Management Service (EMS) of the programme, and in particular its rapid mapping module, is depicted. Finally, in Section 3.3 and Section 3.4 the case studies, along with its corresponding activation, are described.

3.1. Copernicus programme

Copernicus is a programme launched by the European Union (EU) that aims to develop information services based on satellite Earth Observation and in-situ data, not related to space.

Consequently, the programme is implemented by the European Commission (EC), and receives the support from the European Space Agency (ESA) for the Earth Observation side, while from the European Environment Agency (EEA) for the in-situ data one.

The objective of Copernicus is to monitor and forecast the state of the environment on land, sea and in the atmosphere, in order to support climate change mitigation and adaptation strategies, the efficient management of emergency situations and the improvement of the security of citizens. Hence, vast amount of global data from satellites and from ground-based measurement systems are used to provide information to service providers, public authorities and international organizations to improve the quality of life of the people.

The information provided thanks to Copernicus is used to improve people's safety in cases of occurrence of natural disasters such as floods or forest fires, and thus help to prevent losses regarding lives and properties, and damages to the environment.

Copernicus is considered a user driven programme, since the information services provided are free and open available to all the users, although previous authorisation must be granted before accessing them. Therefore, the programme is at fully disposal of citizens, public authorities and policy makers, scientists, entrepreneurs and business.

The declared programme's mission is multiple, taking into account the following points:

- Achieve major societal goals: Copernicus facilitates the management of threats such as climate change or food shortages, and enable progress towards societal goals in several areas.

- Global benefit: Copernicus supports regional, national and international efforts to identify, respond and adapt to global phenomena such as pollution and state of the seas.

- Foster research and innovation: Copernicus supports cutting-edge research and development by supplying information as well as operational products and services to scientists.

- Contribute to the economy: Studies have stated that Copernicus stimulates European enterprise to explore new growth, business opportunities and foster job creation.

- Enhance EU position: Copernicus provides Europe with an autonomous capacity for Earth Observation, important fact since independent information impact on policies and decisions.

Copernicus services support a broad range of environmental and security applications. Some of these applications whose subsequent result translates into benefits for the people are found in agriculture, climate change, development and cooperation, energy, environment, health, insurance, blue economy, tourism, transport (air, land and water), security, urban and regional planning and civil protection and humanitarian aid.

This thesis is focused on the last-mentioned application (civil protection and humanitarian aid), which is based on Emergency Management Service. Hence, the aim of next section is to provide a presentation regarding this service.

3.2. Emergency Management Service (EMS)

The Copernicus Emergency Management Service (EMS) aims to provide information for emergency response in relation to different categories of disasters. These disasters' categories encompass meteorological and geophysical hazards, deliberate and accidental man-made disasters, and others humanitarian disasters.

The objective of the service is to provide information throughout the entire development of the disasters, consequently accounting for prevention, preparedness, emergency response and recovery activities.

The Emergency Management Service provides maps, combining hazard information with socio-economic data to support early warning aiming to reduce the impact of a potential disaster and crisis management activities, as well as post-disaster needs assessment, recovery planning and monitoring of reconstruction and rehabilitation programmes.

There are three modules constituting the Copernicus Emergency Management Service:

- European Flood Awareness System (EFAS), which is the first operational system that monitors and forecasts flood events across Europe. EFAS aims at delivering added value information to the national hydrological services while at the same time providing a unique overview on the current and forecast flood situation to the emergency response coordination centre. It provides maps of floods probabilities up to 10 days in advance as well as detailed forecasts using real time data. EFAS information can also contribute to improve flood extent monitoring.

- European Forest Fire Information System (EFFIS), which consists of a modular web geographic information system that provides near real-time and historical information regarding forest fires in Europe, Middle East and North Africa. Fire monitoring in EFFIS encompasses the full fire cycle, providing information on the pre-fire condition as well as assessing post-fire damages.

- Copernicus EMS – Mapping, which provides all actors involved in the management of natural disasters, man-made emergency situations and humanitarian crises, with timely and accurate geospatial information derived from satellite remote sensing and completed by available in-situ or open data sources.
  The information generated by the service either can be used as supplied or it may be further combined with other data sources. Nevertheless, in both cases it supports geospatial analysis and decision-making processes of emergency managers.
  Furthermore, a validation methodology which aims to check the sample of service outputs produced, along with monitorization of user satisfaction are included in order to improve the overall quality of the service.
  Copernicus EMS – Mapping is provided during all phases of the emergency management cycle in two temporal modes:

- Risk & Recovery Mapping, consisting of geospatial information in support of emergency management activities not related to immediate response. This applies in particular to activities dealing with prevention, preparedness, disaster risk reduction and recovery phases.

- Rapid Mapping, which is introduced hereunder, in Section 3.2.1.

3.2.1. Rapid mapping

Rapid mapping consists of the on-demand and fast provision (within hours or days) of geospatial information in support of emergency management activities immediately following an emergency event.

The service is based on the acquisition, processing and analysis in rapid mode of satellite imagery and other geospatial raster and vector data sources. Additionally, the products offered are standardised following a set of parameters the user can choose when requesting the service. Therefore, the user can choose between three different map types and two production modes (service levels). These maps provided through satellite imagery are then used to include into them information in terms of images and texts retrieved from social networks, which in turn are associated to affected places and areas. This information derived from social media is then used by rescue teams to gain situational awareness about disasters and to identify those locations that are in a critical state, with the ultimate aim of conducting the necessary operations.

Maps differ according to the service level requested. In the case of service level 1 (SL1), maps are provided within some hours after delivery and quality approval of imagery. For service level 5 (SL5), all map types are typically provided within five working days.

Maps can be requested individually or in combination with other map types. The three offered map types are the following:

- Reference maps: Provide a quick updated knowledge on the territory and assets using data prior to the disaster. The content consists of selected topographic features on the affected area, in particular exposed assets and other available information that can assist the users in their specific crisis management tasks. A reference map is normally based on a pre-event image captured as close as possible prior the event.

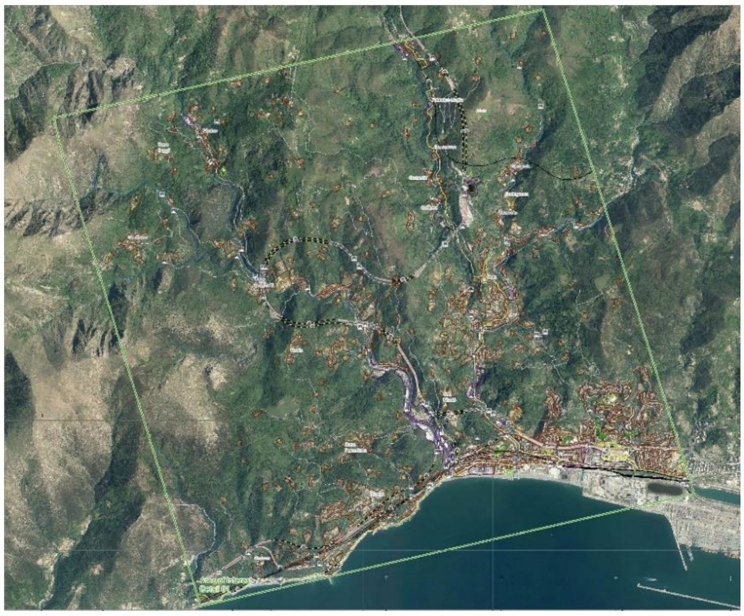


Figure 5: Reference map. Source: copernicus.eu.

- Delineation maps: Provide an assessment of the event extend and are derived from satellite from satellite post-disaster images. They vary depending on the disaster type and the delineation of the areas impacted by the disaster.

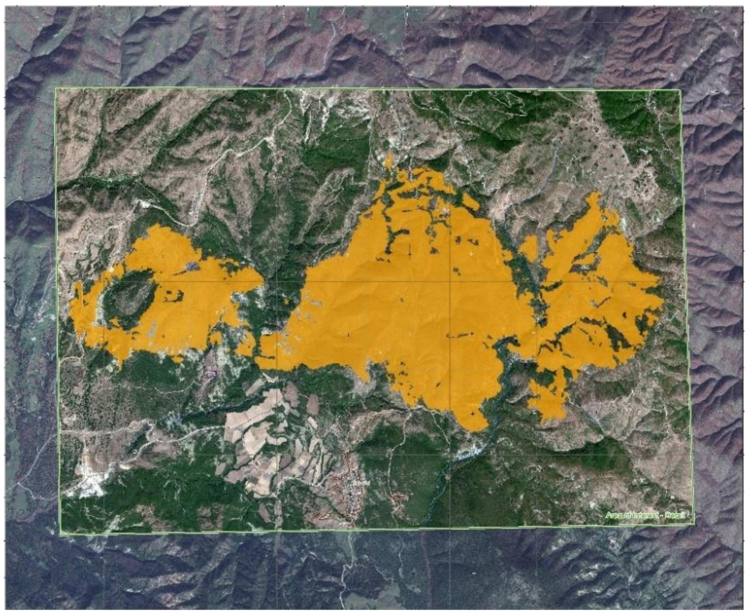


Figure 6: Delineation map. Source: copernicus.eu.

- Grading maps: Provide an assessment of the damage grade and are derived from post-event satellite images. Grading maps include the extend, magnitude or damage grades particular to each disaster type. They may also provide relevant and up-to-date information that is specific to affected population and assets.

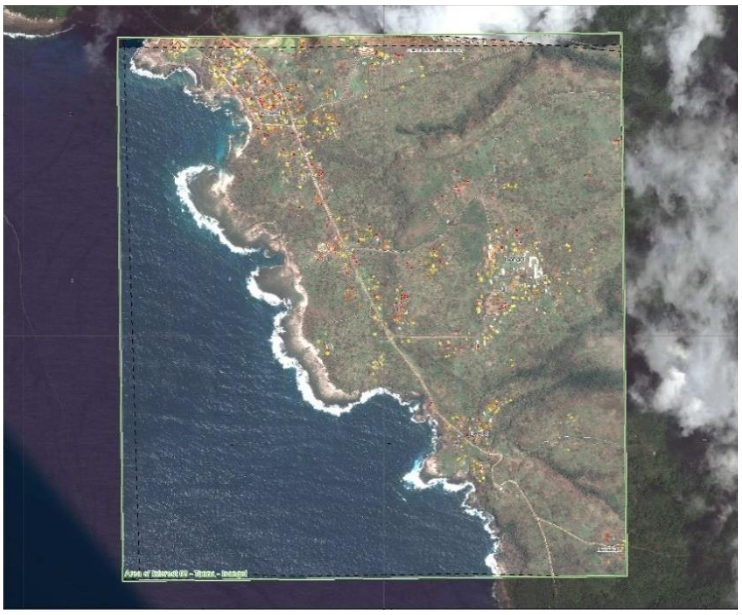


Figure 7: Grading map. Source: copernicus.eu.

The four main challenges that directly affect the performance of rapid mapping services regarding the information retrieved from social networks are the following:

- Volume: Acquiring social media data involves receiving, processing and storing large number of items at a rapid pace. Hence, data collection processes for rapid mapping purposes are needed to be efficiently and resiliently implemented.

- Variety: Social media messages carrying relevant information for rapid mapping purpose are heterogeneous. Hence, cluster systems that retrieve precisely the information needed, and in turn entailing high recall levels, are required.

- Velocity: Social media streams arrive at a faster rate than documents in other collections. Hence, since rapid mapping services demand of immediate information to be implemented, systems must be able to process large datasets in low-latency or real-time contexts.

- Veracity: False information in social media is frequently cited as one of the major obstacles in emergency operations, and consequently could affect the accuracy of rapid mapping development. Hence, policies based on trust and reputation of sources must be implemented to the system to avoid this type of challenge.

3.3. Southern England 2014 floods

The case study analysed is based on the floods that affected the southern part of United Kingdom during February 2014. Between the period from December 2013 to February 2014, the south of England saw heavy rainfalls associated with severe storm which caused widespread flooding, power cuts and major disruptions to transport. Meteorological studies reported that these storms were responsible for the wettest December to January period since 1876. The flood phenomena ranged from coastal flooding, pluvial flooding, fluvial flooding to groundwater flooding. The worst affected areas were Somerset, Devon, Dorset, Cornwall and the cities of Datchet and Egham in the south west and the Thames Valley in the south east.

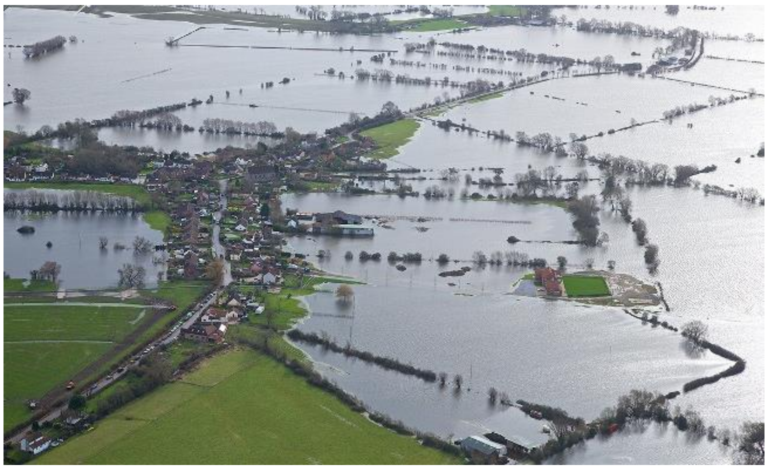


Figure 8: Aerial image of Somerset on 2th February 2014. Source: Tim Pestridge.

The dataset originated from this case study will be used as basis to perform the analyses presented in Chapter 5 and 6, and therefore to devise the resulting classification method introduced in Chapter 7. Furthermore, it will be employed to evaluate the tool presented in Chapter 8.

The collection and subsequent annotation processes of the tweets corresponding to this case study are described in Chapter 4.

3.3.1. Event activation

In particular, this case study belongs to the EMS activation "EMSR069: Floods in Southern England", which objective was to perform rapid mapping activities during the flood emergency.

The activation started on February 10, 2014 at 10:10 hours (UTC). The affected country was obviously referred as "United Kingdom of Great Britain and Northern Ireland" and the area studied was defined as "Somerset Levels, Berkshire, England".

The activation reason provided by the EMS was the following one: *Since the end of January heavy rainfall have caused severe floods in different districts of southern England. Two severe flood warnings remained in place in the Somerset Levels and river levels were expected to continue rising along the Thames*.

Regarding the number of maps provided segmented by types, 13 reference maps and 22 delineation maps were supplied. Nonetheless, no grading maps were provided during the activation.

The main areas on which the activation focused on, and therefore the ones on which maps were based, were: Worcester, Kenley, Hambledon, Bridgewater, Staines and Maidenhead.

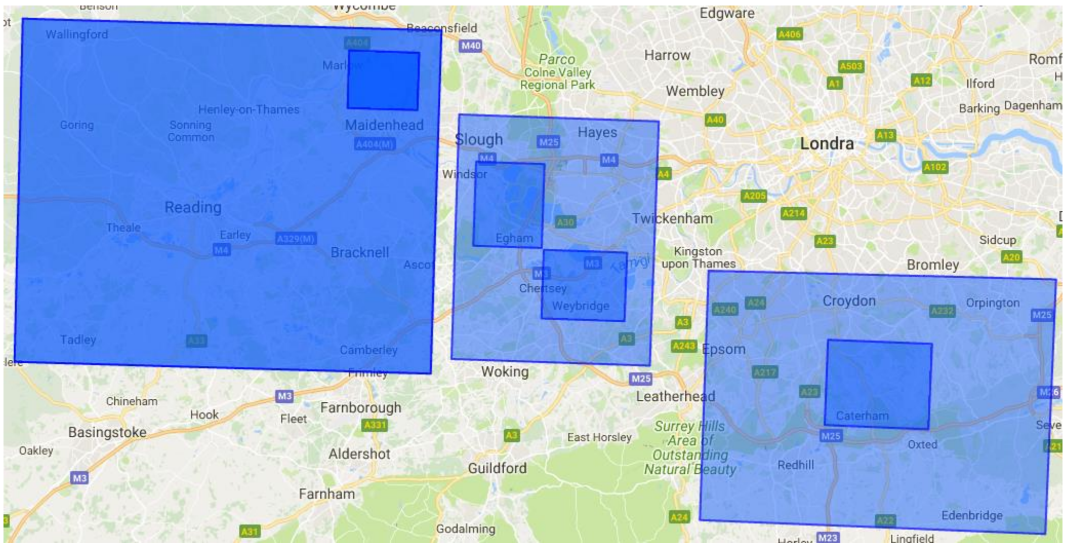


Figure 9: Map of the EMSR069 Rapid Mapping activation. Affected areas are within the blue rectangles. Source: copernicus.eu.

3.4. Hurricane Harvey 2017 storms in Texas

The case study analysed is based on the storms caused by Hurricane Harvey that affected Texas (United States) during August 2017. The widespread and catastrophic effects of Hurricane Harvey resulted in one of the costliest natural disasters in United States history. Throughout Texas, approximately 336,000 people were left without electricity and tens of thousands required rescue in that August of 2017. Besides, 103 people died in storm-related incidents. More than 48,700 homes were affected by Harvey throughout the state, including over 1,000 that were destroyed and more than 17,000 that sustained major damage.

The dataset originated from this case study will be only employed to evaluate the proposed classification method, introduced in Chapter 7, implemented in the tool in Chapter 8.

3.4.1. Event activation

In particular, this case study belongs to the EMS activation "EMSR229: Hurricane Harvey in Texas", which objective was to perform rapid mapping activities during the storm emergency.

The activation started on August 25th, 2017 at 17:00 hours (UTC). The affected country was obviously referred as "United States of America" and the area studied was defined as "Texas, USA".

Figure 10: Rescue operations on August 28th 2017 in Houston (Texas). Source: USA Today.

The activation reason provided by the EMS was the following one: *Harvey made landfall as a Category 4 hurricane on August 25, 2017 at 11:00 p.m. EDT at San Jose Island, Texas. Harvey caused prolonged heavy rains, flooding and storm surge along the Texas coast. President Donald J. Trump issued a Major Disaster Declaration for the state of Texas, making federal funding available for emergency work and to affected individuals and businesses owners who sustain damage as a result of the storm.*

Regarding the number of maps provided segmented by types, 28 delineation maps were supplied. Nonetheless, no grading and reference maps were provided during the activation.

The main areas on which the activation focused on, and therefore the ones on which the maps were based, were: Austin, Beaumont Orange, Houston, La Fayette, Lake Charles, Mathis and San Antonio.

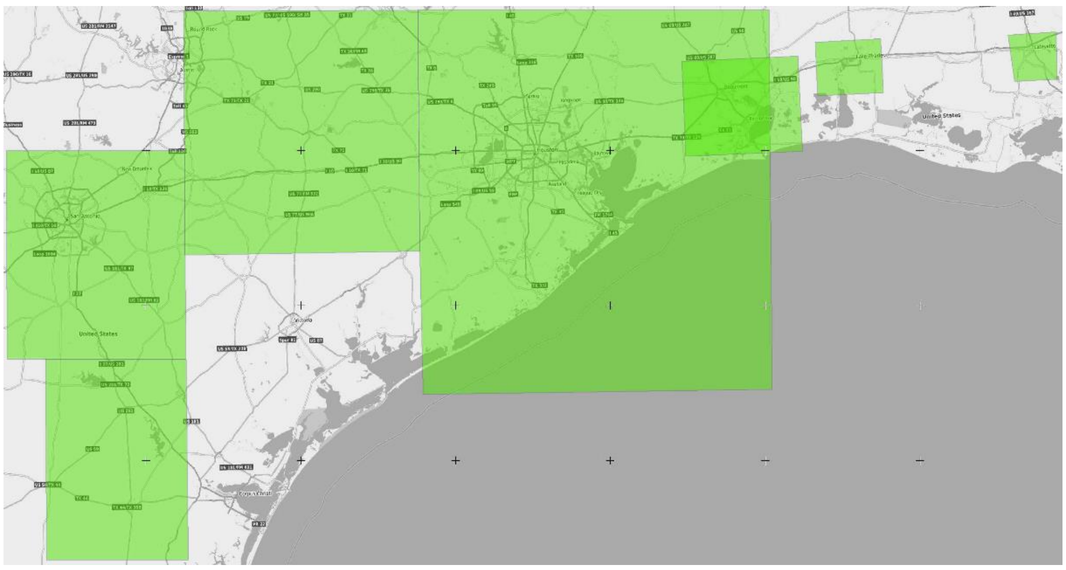


Figure 11: Map of the EMSR229 Rapid Mapping activation. Affected areas are within the green rectangles. Source: copernicus.eu.

Chapter 4

Reply's tweets network

The aim of this chapter is to describe the methodology followed to collect all the necessary data [5]to perform the subsequent analyses, presented in Chapter 5 and 6. Firstly, an overview about tweet cascades is introduced. A tweet cascade is defined the network of all the replies originated from an original tweet, namely that tweet that is not a reply to any other tweet. Afterwards, the chapter encompasses the description of the processes corresponding to the collection of tweets, and therefore the construction of their corresponding cascades, the annotation of the tweets according to their relevance, and finally a depiction of the resulting database.

In the following sections, first is it introduced the concept of tweet cascades corresponding to replies, consistently referenced throughout the thesis, in Section 4.1. The processes carried out regarding data collection and tweets annotation are described in Section 4.2 and Section 4.3, respectively. Finally, in Section 4.4 the resulting database file from the processes, and therefore to be used to perform subsequent analyses, is presented.

4.1. Tweet cascades

A tweet cascade (also called tweet tree) is the set of tweets in which one of these tweets is not a reply to any other tweet (main parent) and each one of the rest of the tweets is either a reply to this main parent tweet, or is part of a chain of replies whose first tweet is a reply to the main parent tweet.

[5] The data to be collected corresponds to the Southern England 2014 floods case study.

To facilitate the understanding of cascades, levels are used to describe them. A level can be defined as a position in the chain of tweets within cascades. Therefore, the main parent tweet is found in level 0 (L0). Replies to that level 0 tweet would shape level 1 (L1). Consequently, replies to level 1 tweets would shape level 2 (L2). Hence, replies to level i would shape level i+1. The highest number of level that has a tweet belonging to it in a cascade is defined as the maximum level reached of the cascade.

Definition:

$The\ representation\ of\ a\ tweet\ cascade\ is\ a\ tuple\ (L, T, C, X, MLR), where:$

- $L = \{0,1,2, \dots, n\}\ is\ the\ set\ of\ levels;$
- $\mathrm{T} = \{T_0, T_1, \dots, T_n\}\ is\ the\ set\ of\ level\ tweets, in\ which\ T_i = \{t_1, t_2, \dots, t_{li}\}\ is\ the\ set\ of\ tweets\ of\ level\ i;$
- $\mathrm{C} = \{C_1, C_1, \dots, C_n\}\ is\ the\ set\ of\ level\ chains, in\ which\ C_i = \{c_1, c_2, \dots, c_{li}\}\ is\ the\ set\ of\ chains\ of\ level\ i;$
- $X{:}\,T \times C\ is\ the\ function\ assigning\ the\ next\ tweet\ to\ each\ pair < t, c >;$
- $MLR = n\ is\ the\ maximum\ level\ reached\ of\ the\ cascade.$

$Obs{:}\ The\ set\ of\ tweets\ of\ level\ 0\ is\ T_0 = \{t_1\}\ for\ every\ cascade.$

An example of a tweets' cascade containing tweets in four levels is represented through a tree diagram in the figure below (Figure 12).

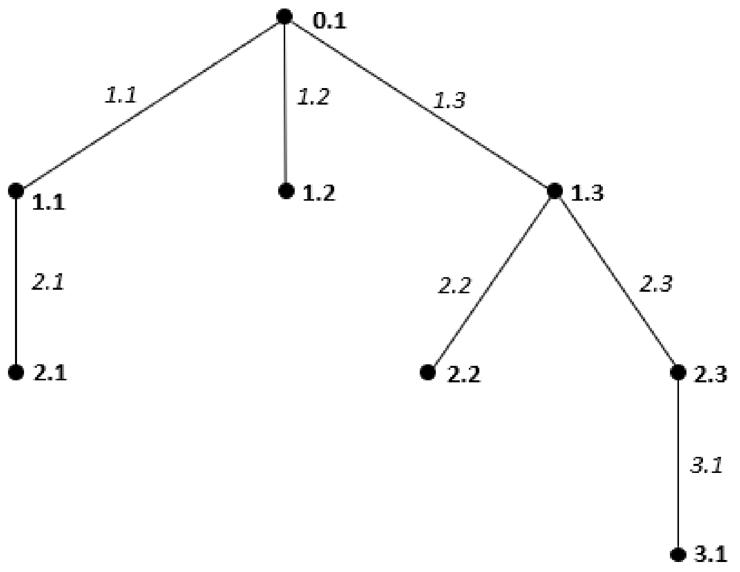
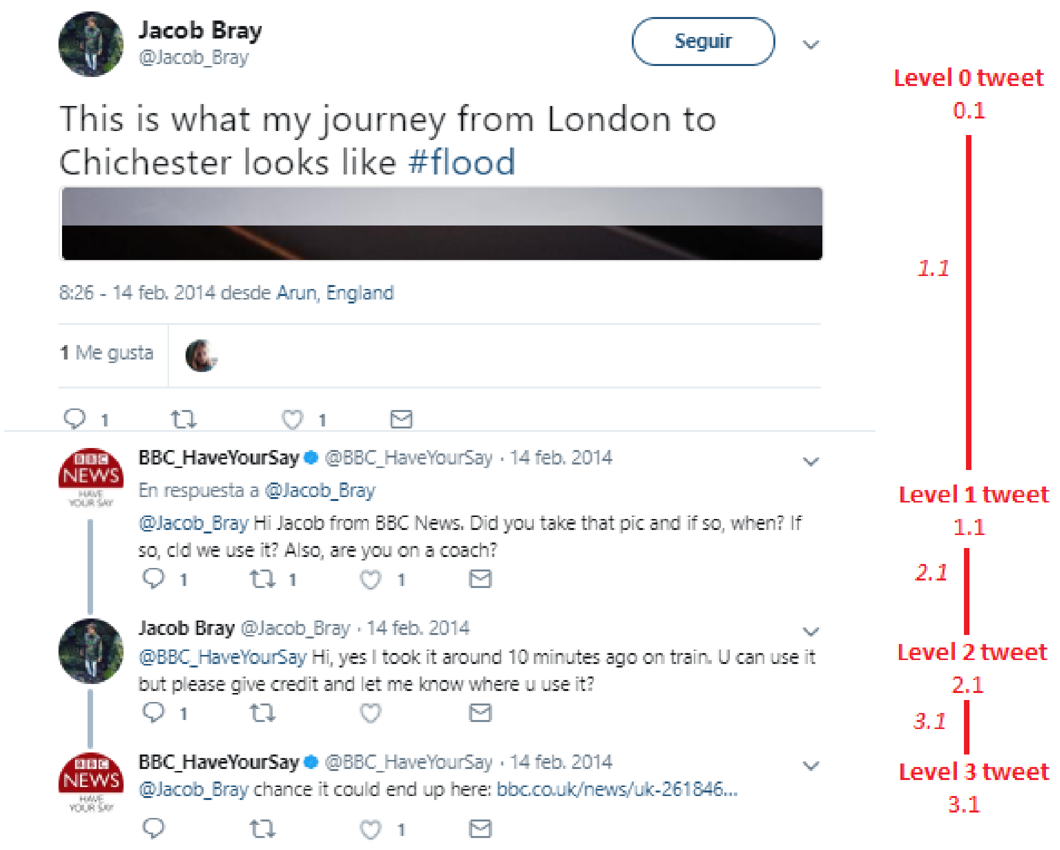


Figure 12: Tweets' cascade example in tree representation.

- $L = \{0,1,2,3\}$;
- $T = \{T_0, T_1, T_2, T_3\},\ in\ which\ T_0 = \{t_1\}, T_1 = \{t_1, t_2, t_3\}, T_2 = \{t_1, t_{2,}, t_3\}, T_3 = \{t_1\}$;
- $C = \{C_1, C_2, C_3\},\ in\ which\ C_1 = \{c_1, c_2, c_3\}, C_2 = \{c_1, c_{2,}, c_3\}, C_3 = \{c_1\}$;
- $MLR = 3$.

Next figure (Figure 13) presents another example of a tweets' cascade containing tweets in four levels represented through the Twitter web interface.



Jacob Bray
@Jacob_Bray

This is what my journey from London to Chichester looks like #flood

8:26 - 14 feb. 2014 desde Arun, England

1 Me gusta

BBC_HaveYourSay @BBC_HaveYourSay · 14 feb. 2014
En respuesta a @Jacob_Bray
@Jacob_Bray Hi Jacob from BBC News. Did you take that pic and if so, when? If so, cld we use it? Also, are you on a coach?

Jacob Bray @Jacob_Bray · 14 feb. 2014
@BBC_HaveYourSay Hi, yes I took it around 10 minutes ago on train. U can use it but please give credit and let me know where u use it?

BBC_HaveYourSay @BBC_HaveYourSay · 14 feb. 2014
@Jacob_Bray chance it could end up here: bbc.co.uk/news/uk-261846...

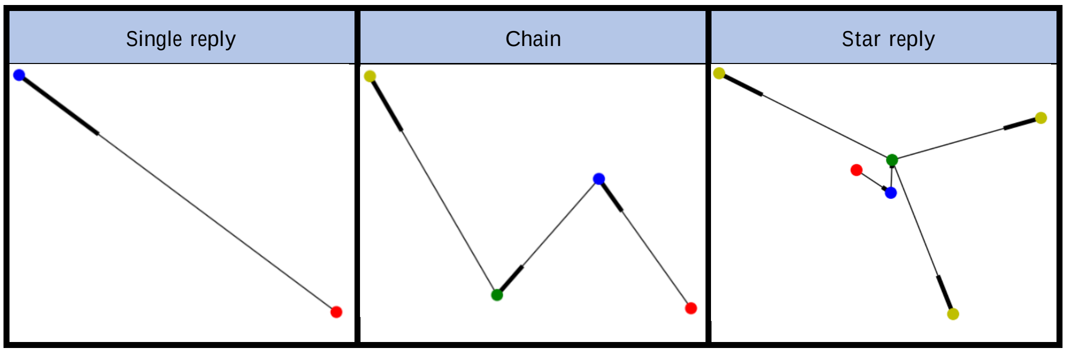
Figure 13: Tweet cascade example in Twitter interface. Parameters previously described are highlighted in the right side. Source: Twitter.

A tweet cascade could be seen as a conversation between people, while a single tweet as a message from a person. In some cases, it will be useful for our study just consider single messages, although in other cases considering the entire conversation will be required to make any kind of conclusion. Hence, the advantage of analysing tweets through cascades is threefold:

- Its structured approach allows the performance of accurate further analyses.
- It provides meaning through which tweets belonging to the same chain can be contextualized.
- Additional information can be extracted by looking to the entire set of tweets within cascades.

Cascades are often explained or/and analysed by considering their structure form. The seven different forms that a cascade can adopt are the following:

- Not developed, when no tweets are found in L1 (no replies).
- Single reply, when there is just 1 tweet in L1.
- Chain, when there is just 1 tweet in L1 and just 1 tweet in every level after L1.
- Star reply, when there is just 1 tweet in L1 and 2 or more tweets in any level after L1.
- Star, when there are 2 or more tweets in L1 and no more levels reached.
- Star to chain, when there are 2 or more tweets in L1 and just 1 tweet in every level after L1.
- Star to stars, when there are 2 or more tweets in L1 and 2 or more tweets in any level after L1.



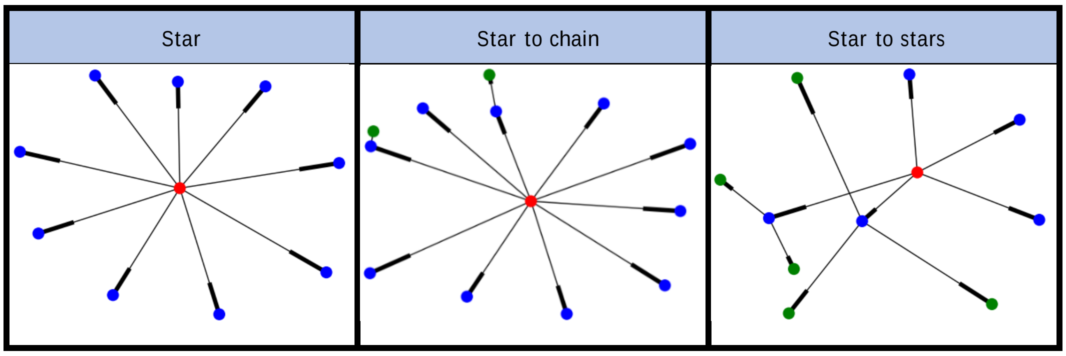


Figure 14: Cascade forms. Nodes represent tweets, edges represent reply connections. Colours in nodes represent tweets' level: red for L0; blue for L1; green for L2; yellow for L3. Edges' bold part represent arrows.

4.2. Tweets collection

This section deals with the description of the collection process followed to extract the reply tweets originated from a dataset containing more than 370 tweets, previously retrieved using a keyword-based approach, belonging to the EMS activation corresponding to the Southern England 2014 floods case study exposed in Section 3.3.1. In order to achieve such objective, some programs have been programmed using Python language and its associated libraries. It must be annotated that all the Python files and functions used in this thesis have been created by its author, namely that no function was externally provided.

The tweets contained in that initial dataset most of the times resulted to be the main parent tweet of a cascade, therefore belonging to L0, although some replies were also found. Since the thesis is focused on the study of reply tweets originated from L0 tweets (cascade's parents), the first task is to gather them all for every cascade developed from the provided dataset.

Hence, the first step is, for every tweet in the original set provided, to obtain the main parent tweet of its corresponding cascade[6], ergo the L0 tweet from which the cascade is build. The programs used for this step are the functions *ex_parents*

[6] Tweets already being main parent tweets are also submitted to the process, for simplicity purposes.

and *prova_par* from the *p.py* file[7]. These functions proceed to open the Excel database where the original set of tweets registered and extract their id. Then, the webpage's source (XML[8] of the page) from every tweet is parsed using the BeautifulSoup library. Note that the webpage from a tweet has always the following structure: 'http://twitter.com/statuses/id', where id is the id number of the tweet. After parsing it, its previous parent is extracted, namely the tweet being replied (if the case) thanks to the attribute 'data-replied-tweet-id' inside the class 'div'. This process is iterative until no parent is found, meaning that the tweet being analysed is already the L0 tweet of that cascade. Once all the tweets in that original set have been parsed, the resulting tweets are saved in a new database. Finally, tweets not available and duplicates in that new set of tweets are eliminated, resulting in a set of 362 tweets. Having a duplicate means that in the original set of tweets, two or more tweets belonged to the same cascade and therefore shared the same first parent.

The next step is the construction of the cascades, namely to gather all the tweets (replies) belonging to every cascade from the set of L0 tweets just collected. The programs used for this step are the functions *ex_cascades*, *tree*, *prova_nova* and *concatenar_ex* from the *p.py* file. These functions proceed to open the Excel database containing the L0 tweets and extract their id. Then, for every tweet, and therefore for every cascade, its webpage's is open, scrolled down until the end for all the tweets to appear in the screen and lastly parsed. The need of scrolling down the webpage is to make a complete parsing, since in a parse the information provided is just the one which appears in the screen. After parsing, thanks to the attribute 'data-tweet-id' inside the class 'div', all the tweets' id corresponding to a cascade are extracted. Then, to sort these tweets by levels, an iterative process is made for each tweet in the cascade. The process consists on parsing every single tweet website, and then consider the attributes 'data-has-parent-tweet' and 'data-mentions' to assign its corresponding level. As mentioned before, this entire task is done by every cascade. Once all the tweets from every cascade are extracted and assigned by its level, they are saved in a new database containing the id of the tweet, its cascade number, the id of its corresponding parent (if the case) and the ids of its corresponding replies (if existing).

[7] All the Python files and functions used in Section 4.2 and 4.3 are enclosed in the Appendix (Section iii.a).

[8] In computing, Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable (Wikipedia).

The final step is to extract all the necessary information from every single tweet within the database just created. To retrieve that information, the API server provided by Twitter is used, and in particular through Tweepy, its corresponding Python library. The programs used for this step is the function *get_tw* inside the *get_tweets.py* file. First of all, a developer account is created since to access the API server it is required to be registered. Subsequently, the functions proceed to open the Excel database containing all the tweets being considered and extract their id. Then, for every tweet, we request its status interacting with the API server and a file containing all the values for every parameter registered in JSON format is returned. Afterwards, for every parameter that is of our interest its corresponding value is extracted, converted to a string format and sorted by an established order. This process is carried out for every tweet in an iterative basis, and once finished is saved in an alternative Excel file.

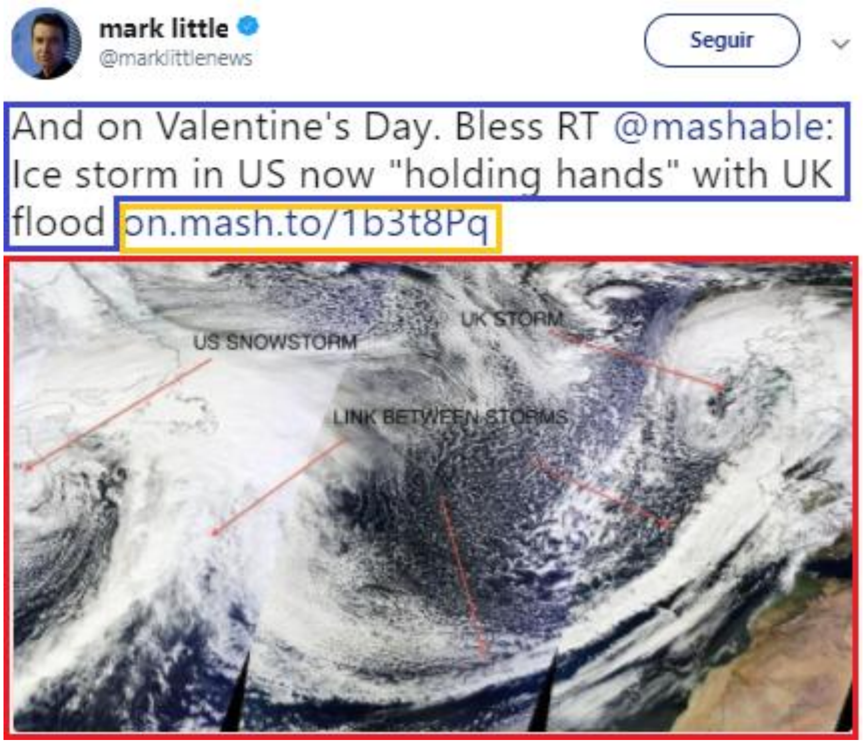
Having finished this last step, since all the cascades are build and the necessary descriptive information from all their tweets is extracted, it is proceeded to the manual annotation of the tweets.

4.3. Annotation of tweets

With the aim of identifying patterns in subsequent analyses that would allow the extraction of, a priori, useful tweets, a manual annotation of the collected tweets is mandatory to be realized. This process is analogous to that of labelling data when creating domain adaptations in supervised classifications to classify tweets (see Section 2.3), although they differ from the fact that this process aims to manually identify patterns while domain adaptation is performed by machine learning.

The importance to perform such a process is that the initial classification of tweets according to their usefulness in natural disaster situations is needed to later be able to identify patterns regarding both useful and non-useful tweets. From now on usefulness will be referred as relevance, since while tweets' usefulness is not known until it is exploited in rescue operations, it is just their relevance in these situations that could be a-priori annotated by us.

Hence, all tweets in our set are analysed sequentially and one by one. For each tweet, it is annotated the content of its text, the media (photo(s), video or gif), and the URL enclosed, which are the only parts in a tweet that can carry information about the natural disaster, using binary values ('0' for irrelevant part and '1' for relevant part). Then, a tweet will be considered relevant if any of its parts is relevant (function OR employed), also using binary values.


mark little
@marklittlenews

And on Valentine's Day. Bless RT @mashable: Ice storm in US now "holding hands" with UK flood on.mash.to/1b3t8Pq

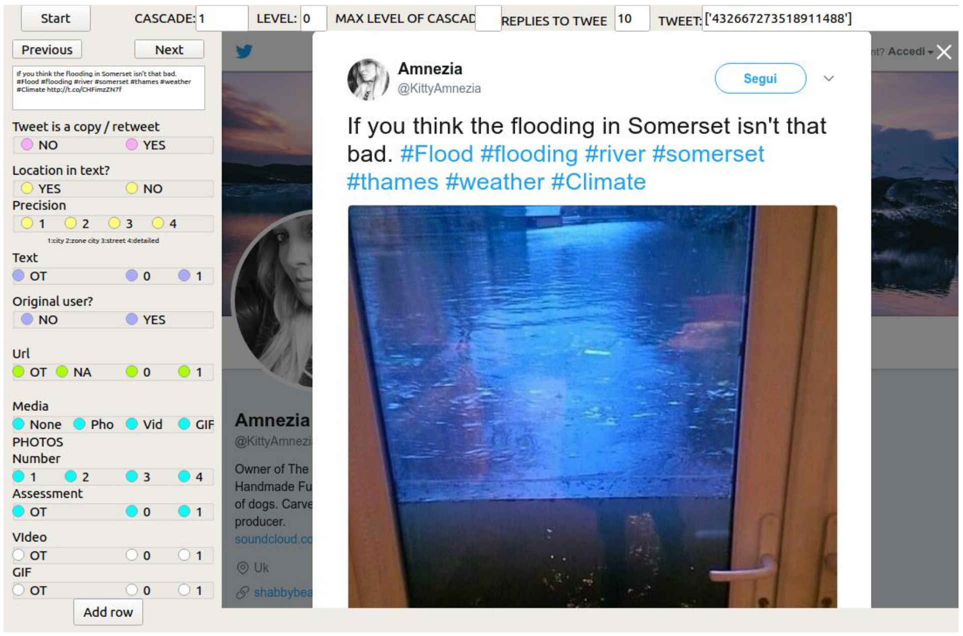
Figure 15: The three elements of annotation in a tweet are highlighted within rectangles: blue, for the text; orange, for the URL; and red, for the media (image). Source: Twitter.

Furthermore, the following fields are also manually registered during the process: if locations are provided within the text (no/yes), if it is a retweet/copy (no/yes), if the creator of the tweet is *original user* [9] (no/yes), and the number of photos provided in the tweet (in the case that photos are provided).

The main issue when performing any kind of manual annotation is the difficulty to maintain the consistency of the appraisements throughout the whole process. To address this problem, an annotation criterion is therefore established for each tweet part to be analysed. The annotation criterion followed is presented in the Appendix (Section i).

[9] Concept introduced in Section 5.1.

Since this process is highly time consuming, fact that in turn could also reduce the aforementioned consistency which has been attempting to reduce through the establishment of criteria, efforts have been made to accelerate the speed of the annotation process. Hence, with the purpose of speeding up the process, a tailor-made interface[10] has been created. This interface allows the immediate visualization of the tweet through the web in its right side, while its left side contains the required buttons to annotate each part of the tweet. When the analysis of a single tweet is completed, the button "Add" located in the bottom of the window automatically write and save the annotation into a row in a specified Excel file. The interface was designed by using the Python library PyQT5.


CASCADE: 1 LEVEL: 0 MAX LEVEL OF CASCAD REPLIES TO TWEE 10 TWEET: ['432667273518911488']

If you think the flooding in Somerset isn't that bad. #Flood #flooding #river #somerset #thames #weather #Climate http://t.co/CHFimzZN7f

Amnezia @KittyAmnezia

If you think the flooding in Somerset isn't that bad. #Flood #flooding #river #somerset #thames #weather #Climate

Figure 16: Annotation interface screenshot.

The data derived from the annotation is recorded in an Excel file, which it is finally combined with the last file created in the previous section with the aim of obtaining a complete database containing all the required data from the tweets in a single document to later perform pattern identification analyses.

[10] This interface is only employed for the manual annotation process of tweets described in this section (one-time use) necessary for performing the subsequent analyses, and therefore it must not be confounded with the tool introduced in Chapter 8 that deals with the automatic classification of reply tweets.

Data regarding tweets within cascades entirely annotated as out of topic (52 cascades), meaning that they are not related in any way to the case study of the Southern England 2014 floods under analysis, are eliminated from the final database. The decision has been taken mainly considering that our study focuses on the analysis of reply tweets corresponding to natural disasters situations, and specifically originated from the case study, and therefore the inclusion of cascades dealing with other topics such as football or TV shows would cause a not desired noise in it, since their behaviour and characteristics are of a complete different nature.

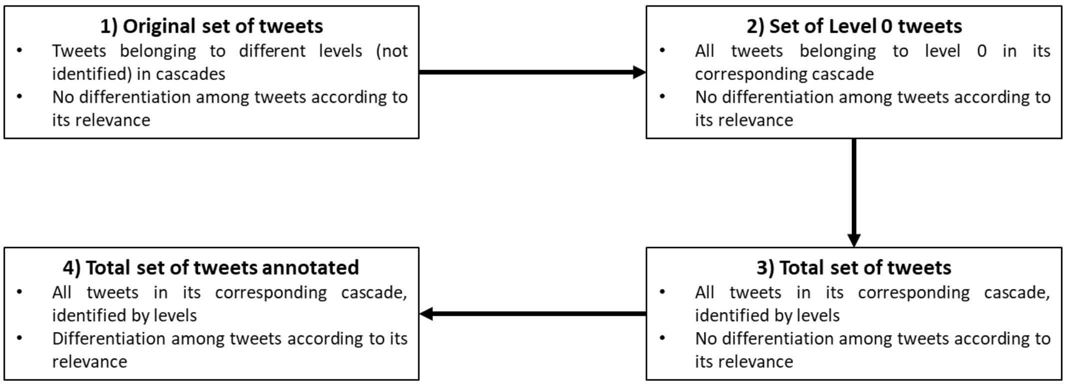
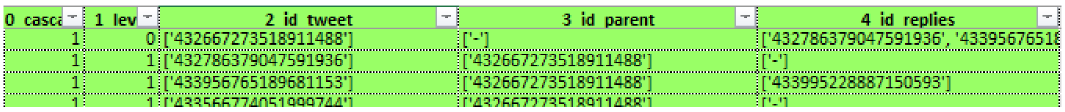

1) Original set of tweets
• Tweets belonging to different levels (not identified) in cascades
• No differentiation among tweets according to its relevance
2) Set of Level 0 tweets
• All tweets belonging to level 0 in its corresponding cascade
• No differentiation among tweets according to its relevance
3) Total set of tweets
• All tweets in its corresponding cascade, identified by levels
• No differentiation among tweets according to its relevance
4) Total set of tweets annotated
• All tweets in its corresponding cascade, identified by levels
• Differentiation among tweets according to its relevance

Figure 17: Process representation (Section 4.2 and 4.3).

4.4. Final database description

The resulting Excel database contains all the information either manually annotated or automatically extracted from the Twitter API server regarding all the tweets related to the case study under analysis.

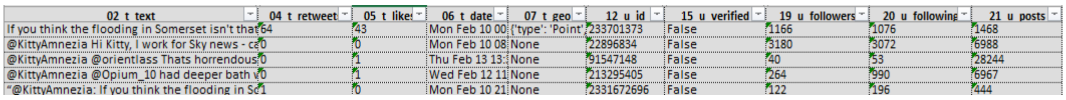
The first 5 columns of the initial sheet deal with the cascade characteristics of the tweets. For each tweet, it is stated the number of the cascade on which it belongs, the cascade level on which it is located, its id, its parent tweet id, and the ids of its reply tweets. Note that all the data regarding to a tweet is recorded in a single row.



0 casca	1 lev	2 id tweet	3 id parent	4 id replies
1	0	['432667273518911488']	['-']	['432786379047591936', '4339567651
1	1	['432786379047591936']	['432667273518911488']	['-']
1	1	['433956765189681153']	['432667273518911488']	['433995228887150593']
1	1	['433566774051999744']	['432667273518911488']	['-']

Figure 18: Columns registering cascade form's features values in the final database.

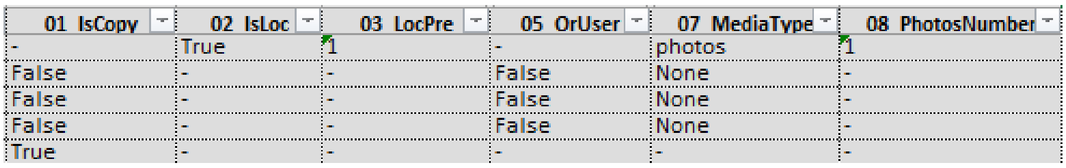
The next 22 columns of the sheet correspond to the information extracted by using the API server offered by Twitter. As instance, fields regarding the text, the number of retweets and likes, the date of publication and geolocation from the tweets along with their user data are registered in these columns.



02 t text	04 t retweet	05 t like	06 t date	07 t geo	12 u id	15 u verified	19 u followers	20 u following	21 u posts
If you think the flooding in Somerset isn't that	64	43	Mon Feb 10 00	{'type': 'Point',	233701373	False	1166	1076	1468
@KittyAmnezia Hi Kitty, I work for Sky news - ca	0	0	Mon Feb 10 08	None	22896834	False	3180	3072	6988
@KittyAmnezia @orientlass Thats horrendous	0	1	Thu Feb 13 13:	None	91547148	False	40	53	28244
@KittyAmnezia @Opium_10 had deeper bath v	0	1	Wed Feb 12 11	None	213295405	False	264	990	6967
"@KittyAmnezia: If you think the flooding in Sc	1	0	Mon Feb 10 21	None	2331672696	False	122	196	444

Figure 19.Columns registering tweet's parameters values from the API server in the final database.

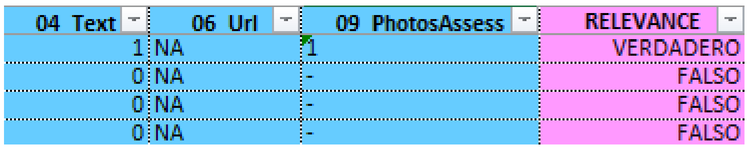
The 6 following columns treat information manually registered during the annotation process regarding locations provided, copies/retweets, and number of photos enclosed in the tweets.



01 IsCopy	02 IsLoc	03 LocPre	05 OrUser	07 MediaType	08 PhotosNumber
-	True	1	-	photos	1
False	-	-	False	None	-
False	-	-	False	None	-
False	-	-	False	None	-
True	-	-	-	-	-

Figure 20: Columns registering tweets' parameters manually labelled values in the final database.

Finally, in the last 4 columns of the sheet are registered all the fields resulting from the annotation of tweets. Annotation results regarding the text, media, and URL from each tweet along with the final relevance outcome are presented in these columns.



04 Text	06 Url	09 PhotosAssess	RELEVANCE
1	NA	1	VERDADERO
0	NA	-	FALSO
0	NA	-	FALSO
0	NA	-	FALSO

Figure 21: Columns registering relevance values in the final database.

Another sheet is subsequently manually created to deal with cascades features. This sheet, instead of representing in a single row all the data regarding a tweet, presents the data resulting from the combination of all the tweets belonging to a cascade, and represents it in a single row for each cascade. This fact allows the execution of analyses regarding cascades features.

Chapter 5

Tweet's descriptive analysis

This chapter makes use of the tweets collected[11] and recorded in the database file described in the previous chapter (Chapter 4) to produce several analyses with the aim of interpreting the results obtained.

Since, as already mentioned before, research studies have never considered the analysis and subsequent understanding of the tweets cascades[12] originated by replies related to natural disasters, it is found relevant to include descriptive analyses regarding the cascades' structures behaviour and their relationship with tweets' characteristics.

The analytical methodology pursued throughout the process is the following: facts and figures are firstly displayed employing tables and graphics, then a depiction aiming to highlight the most relevant outcomes is provided, and finally an explanation of the obtained results is presented. Furthermore, a summary of the main findings in each section is also provided at the end of them.

In the following sections, first it is analysed the structure of cascades, considering their adopted forms and the connections between tweets and levels, among other characteristics, in Section 5.1. On the other hand, in Section 5.2 the relationships between tweets' parameters, such as retweets and followers of the posting users, and cascades' structure are analysed.

[11] Collection process of tweets described in the previous chapter (Section 4.2).

[12] A tweet cascade is the network of all the replies originated from an original tweet.

5.1. Cascade structure characteristics

A descriptive analysis regarding cascade structure characteristics regarding their levels[13] and their adopted forms is performed in this section.

The total number of tweets recorded in the database, originated from the set of tweets provided from the Southern England 2014 floods case study[14], is presented in the next figure (Figure 22) segmented by levels of the cascades.

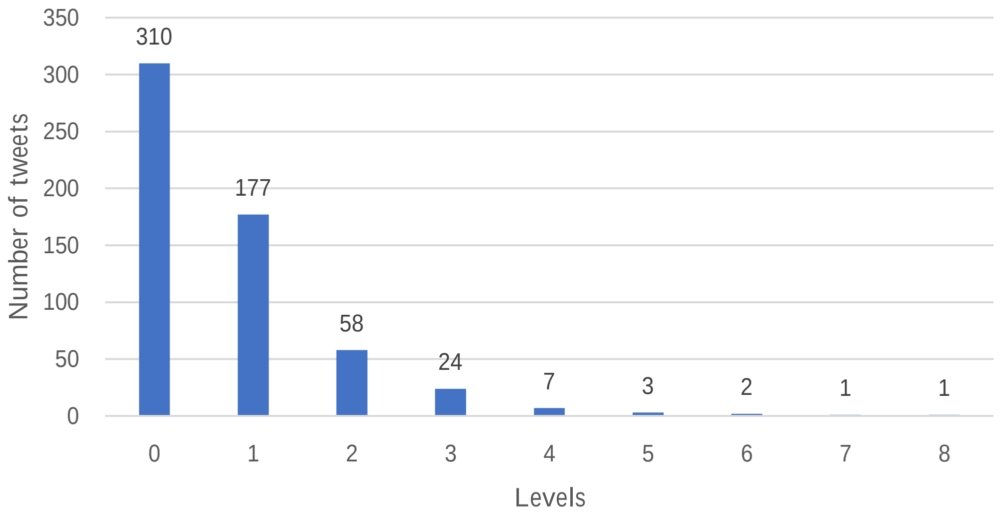


Figure 22: Number of tweets in every level.

As it is shown, the total number of cascades studied is 310, since the number of tweets belonging to L0 per se are equal to the number of cascades. More importantly, the number of tweets in each level decreases the higher the level is. This trend causes that from L4 the number of tweets gathered is lower than 10, meaning that further statistical analyses from this L4 (included) to L8 will not be representative due to their insignificant sample size and therefore results obtained from them will not be entirely conclusive.

[13] A level is a position in the chain of tweets within cascades.

[14] The case study analysed is based on the floods that affect the southern part of United Kingdom during February 2014.

The following figure (Figure 23) displays the maximum level reached [15]by the cascades using percentages. Then, percentages point out how many cascades are able to reach a level.

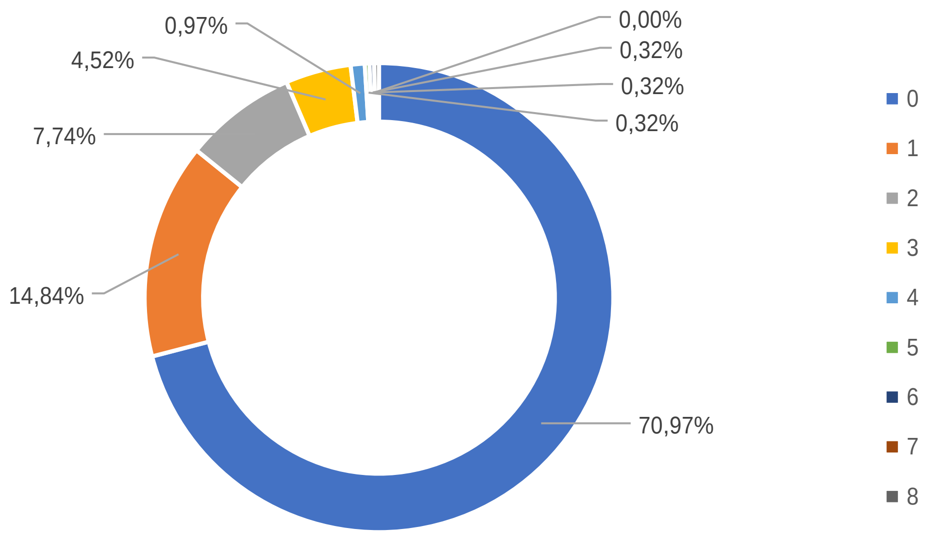


Figure 23: Percentage of cascades that share the maximum level reached, for each level.

The results obtained highlight that cascades most of the times either not develop at all (70,97%) or just contain replies in the first level (14,84%), which means that conversations[16] are not established in an 85% of cascades. This circumstance could be explained by the fact that tweets in L0 either don't have any replying effect on user's followers, or just lead to reactions/personal opinions without any aim of conversational purposes.

The different forms[17] adopted by cascades is analysed in the next figure (Figure 24). It must be annotated that cascades just containing a tweet are not considered in the assay.

[15] The highest number of level that has a tweet belonging to it in a cascade is defined as the maximum level reached of the cascade.

[16] A conversation is established when a reply is in turn replied, originating a chain containing at least 3 tweets, equivalent to say that the cascade has reached at least L2.

[17] Cascades forms are described in Chapter 4.

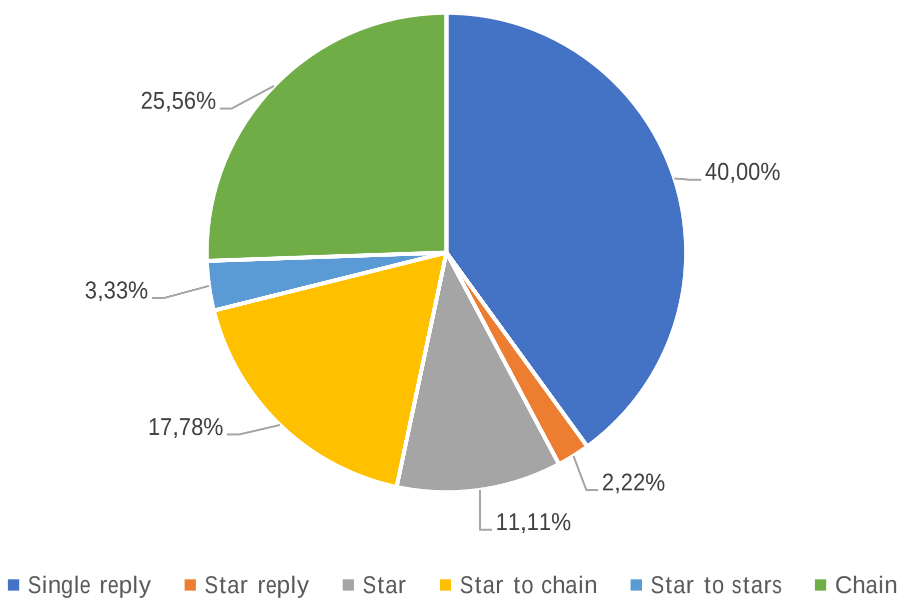


Figure 24: Percentage of cascades that share the same form, for each form.

As it is stressed in the figure, the most adopted form is the *Single reply*, which considers cascades just containing 2 tweets. This fact, as in the case of the maximum level reached, can be also justified considering the exposure of the tweet, direct consequence of its own content and the number of followers of the user (hypothesis confirmed in subsequent analyses). From the set of cascades overcoming L1 (the ones not adopting *Single reply* or *Star* forms), it is observed that they tend to develop either as a chain (*Chain* and *Star to chain* forms). Since chains can be considered as conversational forms, this fact evidences that most of the times cascades develop through conversations.

The next figure (Figure 25) introduces the average number of tweets within a cascade segmented by every of its levels, in the cases that the corresponding cascade reaches these levels[18].

[18] Hence, cascades not reaching a specific level (number of tweets in the level is equal to 0) are not computed in that level.

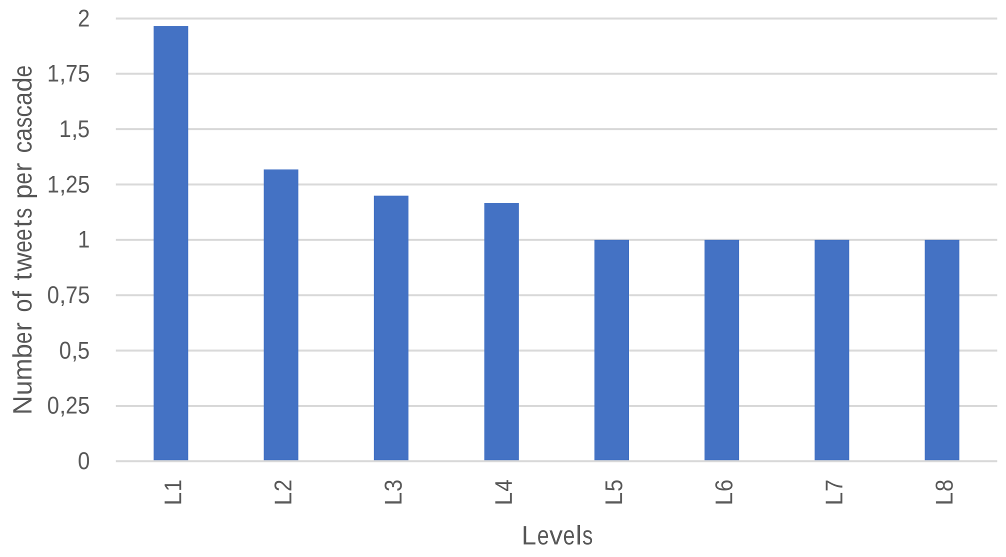


Figure 25: Average number of tweets per cascade, for each level.

The results obtained stand out a downward tendency causing a reduction on the number of tweets belonging to a cascade the higher the level is. The decrease originated between L1 and L2 can be justified considering the form of cascades: the forms that could cause an average number of tweets higher than 1 from L1 (included) per se are *Star to chain*, *Star reply*, *Star* and *Star to stars*[19]. In this sense, the raise in L1 is entailed by the first, third and fourth ones[20] (32,22% of the total cascades). Nevertheless, the increase in L2 can only be explained by the second and fourth form[21] (5,55% of the total cascades), meaning that on average is almost 6 times more likely to have more than 1 tweet in L1 than in L2 within a cascade.

Furthermore, the existent downward tendency also supports the hypothesis that conversations in cascades tend to disappear, entailing a reduction of the tweets in the cascade, the higher the level is.

The following figure (Figure 26) treats the percentage of tweets which receive a reply per cascade segmented by level, in the case that the corresponding cascade reaches that level.

[19] *Single reply* and *Chain* forms just consider cascades owning only one tweet in every level reached.

[20] *Star reply* form consider cascades owning only one tweet in L1.

[21] *Star to chain* form just consider cascades owning only one tweet in every level reached from level 2 (included). *Star* form just consider cascades owning only one tweet in L1.

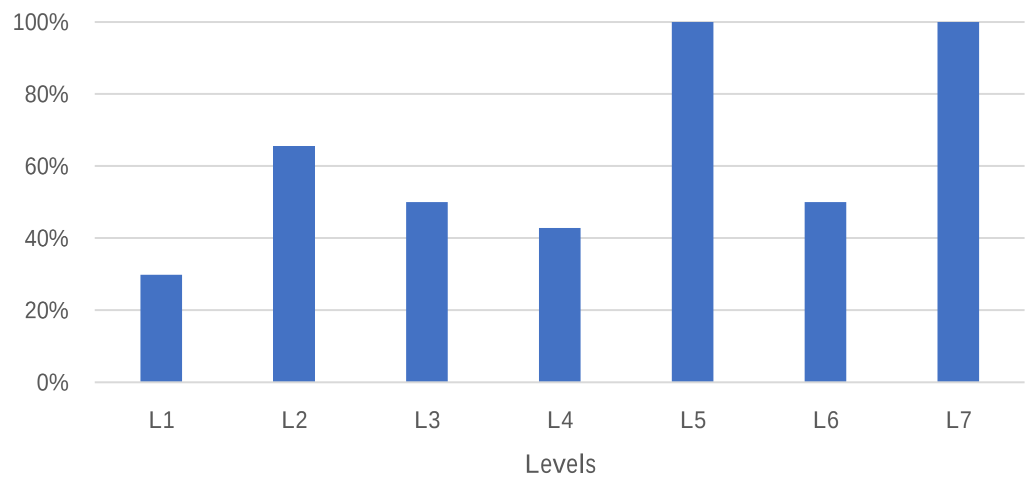


Figure 26: Percentage of replied tweets per cascade, for each level, in the case the cascade has reached the corresponding levels.

Tweets in L1 are usually not replied (70% of the cases), fact that could be explained due two reasons: content and exposure. The content of these tweets most of the times encompass reactions and personal opinions without the aim of being answered. On the other hand, the lower a tweet is exposed, the lower chances of being replied has. Nonetheless, when conversations are established (from L2) the tweets are more likely to be replied again (65% of the cases).

Furthermore, it is seen a reduction on the percentage of tweets belonging to levels 3 and 4 receiving replies, fact that supports the already-mentioned hypothesis that conversations tend to disappear the higher the level is, as explained in the previous analysis.

To better understand the relationship between the number of replies received by tweets in L0 and the maximum level reached by their corresponding cascades the following figure (Figure 27) is introduced.

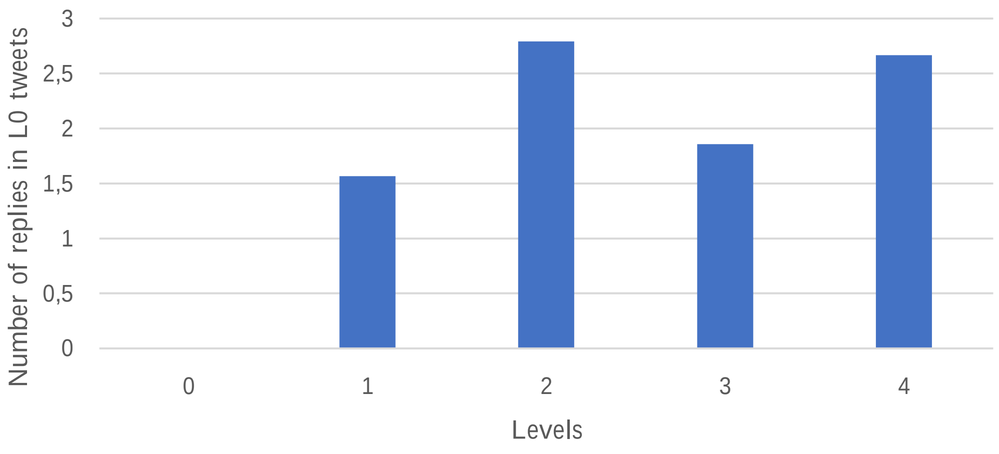


Figure 27: Average number of replies received by L0 tweets of, for each maximum level reached by their corresponding cascades (levels 0 - 4).

In the figure it is demonstrated that receiving more replies in the parent tweet of cascades allows them to reach higher levels. Even if this observation was considered logic, since to reach levels a cascade is needed of replies, had to be proved.

In a cascade, a tweet posted by an *original user* is defined as a tweet posted by the same user that posted the tweet corresponding to the L0 of that cascade, also known as cascade originator. In the next figure (Figure 28) is presented the number of tweets posted by *original users*, segmented by level.

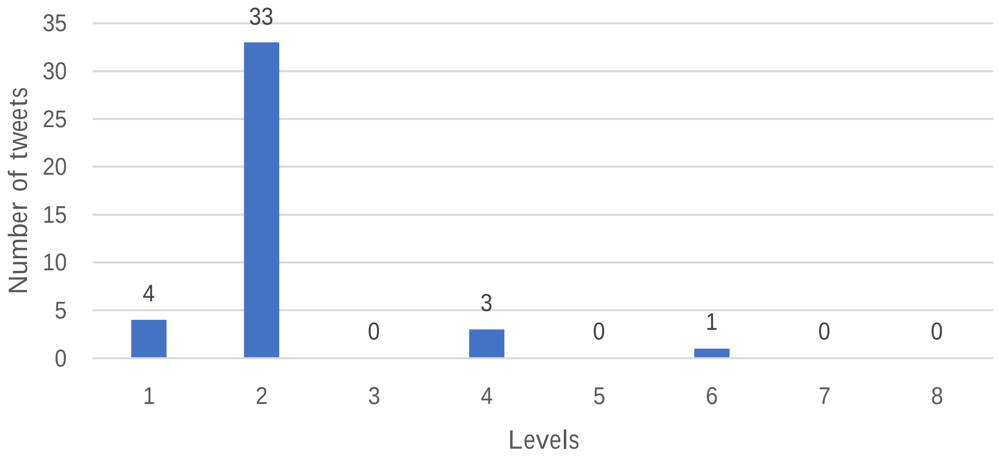
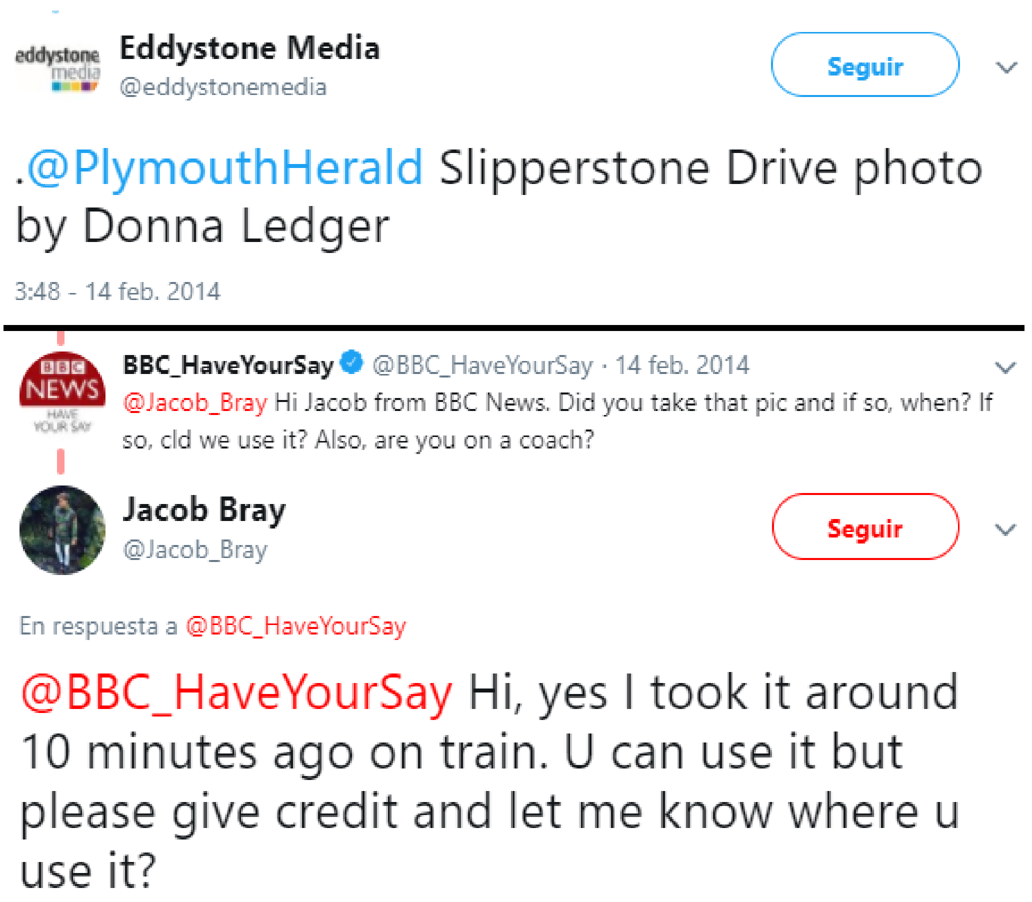


Figure 28: Number of tweets posted by *original users*, for each level.

L1 tweets are used by these users to add information to the L0 tweet, as it has been checked in the obtained 4 samples. However, the level containing more tweets posted by *original user* is L2. The reason behind this fact is that in this level *original users* reply to tweets in L1 carrying questions and messages aiming to be replied.

Furthermore, although the numbers are very low, it can be noted that in further levels posts just occur in even number levels (fourth and sixth) as a consequence of typical conversations structure.

Two examples of tweets posted by *original users*, one belonging to L1 and another to L2, are provided in the next figure (Figure 29). In them it can observed the content characteristics of tweets just described.



Eddystone Media
@eddystonemedia
Seguir

.@PlymouthHerald Slipperstone Drive photo by Donna Ledger

3:48 - 14 feb. 2014

BBC_HaveYourSay @BBC_HaveYourSay · 14 feb. 2014
@Jacob_Bray Hi Jacob from BBC News. Did you take that pic and if so, when? If so, cld we use it? Also, are you on a coach?

Jacob Bray
@Jacob_Bray
Seguir

En respuesta a @BBC_HaveYourSay

@BBC_HaveYourSay Hi, yes I took it around 10 minutes ago on train. U can use it but please give credit and let me know where u use it?

Figure 29: Two examples of tweets posted by *original users*. The tweet on the top (posted by Eddystone Media) belongs to L1. The bottom tweet (posted by Jacob Bray) belongs to L2, which reply to a question, also included, formulated in L1 (posted by BBC_HaveYourSay). Source: Twitter.

In the below figure (Figure 30), the percentage of tweets posted by *original users*, segmented by level, is displayed. Results stress that these tweets most of the times are posted in L2 (57,9%). The explanation to this fact is that tweets in L1 aiming to establish conversations usually tend to involve the user originator of the cascade.

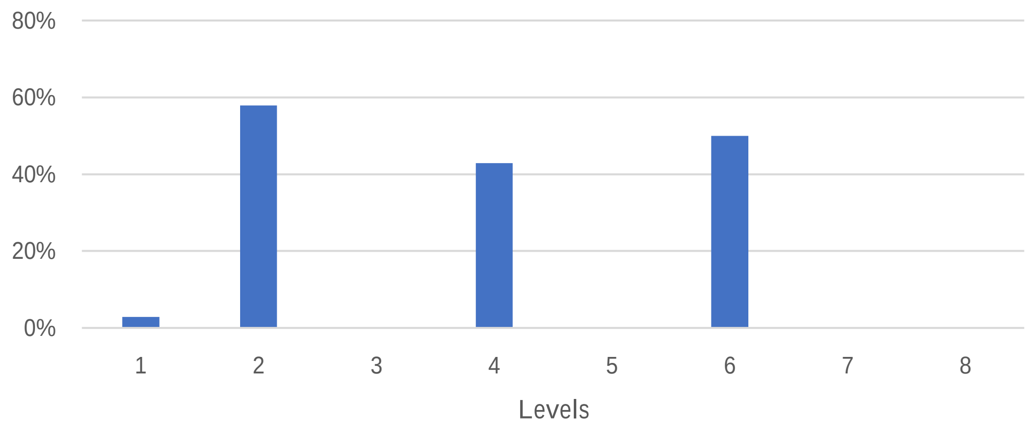


Figure 30: Percentage of tweets posted by *original users*, for each level

The following figure (Figure 31) introduces the average number of replies per tweet, considering just L1, and making a comparison between the values corresponding to tweets posted by all users and the just tweets posted by *original users*.

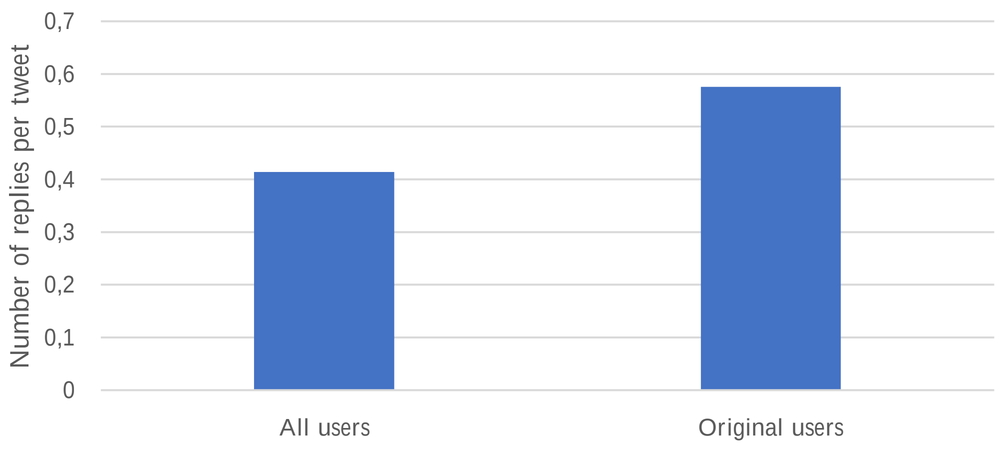


Figure 31: Average number of replies per tweet, differentiating these tweets in the case they are posted by *original users.*

As it is shown in the figure, the average number of tweets replying to the tweets posted by *original users* is higher in L1 (28% higher). This circumstance could be explained by the fact that tweets posted by *original users* causes a major continuation of the established conversation. Besides, replies to these tweets also could tend to contain thank you messages. To check these hypotheses, a content analysis of the replies has been performed. The results confirm them, since conversational replies constitute 68% of the total ones, while the resulting 32% belongs to thank you messages.

Figure 32 presents the probability for a cascade of reaching the following level (level i+1), in the case that the level under study (level i) is reached, for each of the levels.

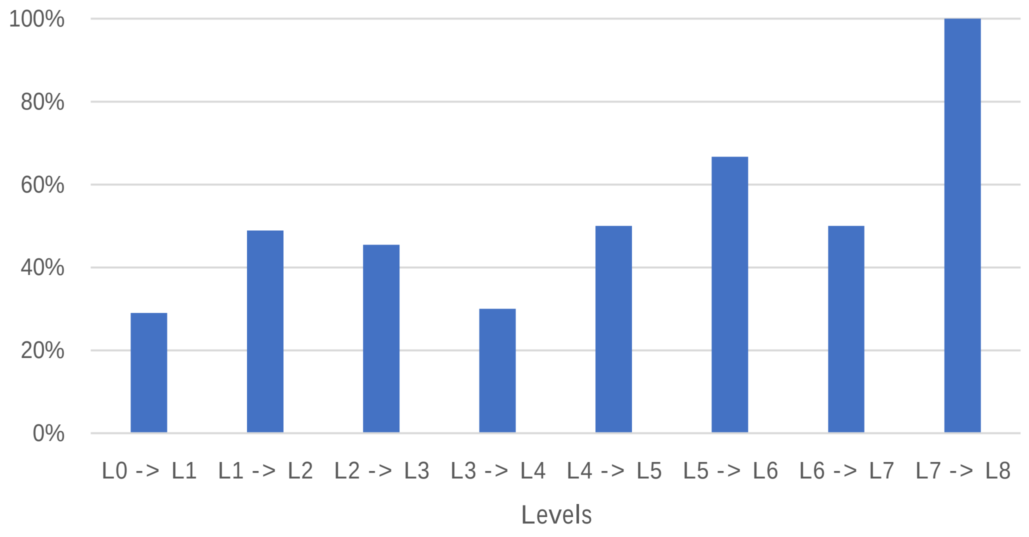


Figure 32: Percentage of cascades that once reached a level, reach the following one, for each level.

Here, the aspect already indicated of the low rate of development of cascades from L0 (30%) is highlighted again. Furthermore, it can be observed that it is more probable to reach L2 once the cascade has reached L1, than reaching L1. This circumstance could be explained by the fact that the content of tweets belonging to L1 lead more to engage users to participate into conversations causing cascades to reach L2 compared to L0 tweets.

Besides, cascades owning tweets in L1 are more exposed than the ones just owning the L0 tweet, not only by the very fact of having received a reply (meaning that already possessed some degree of exposure), but also because they will be diffused from that moment also to the followers of the replying users.

Also, results show that when conversations start (from L2), the probability to reach next levels (until L4) is inversely proportional to the rate of conversations' disappearance.

To sum up, the main findings of this section have been the following:

- The total number of tweets belonging to each level decreases the higher the level is.

- Conversations are usually not established in cascades (85% of the cascades not contain any conversation).

- The most adopted form of cascades (excluding those that don't develop) is *Single reply,* which encompass those owning just two tweets. Nonetheless, cascades overcoming L1 tend to develop as a chain (*Chain* and *Star to chain* forms).

- L1 tweets posted by *original users* are used to add information. Nonetheless, L2 tweets posted by them are used to answer questions, which have varied purposes, formulated by other users in L1.

- Tweets posted by *original users* receive a higher number of replies in the first level (L1).

5.2. Tweet's parameters values in cascades

Descriptive analyses taking into consideration the values of several parameters of tweets extracted from the API Twitter server and their relationship with the structure of cascades are carried out in this section.

The tweets' parameters to be analysed are the number of retweets, number of likes, date creation and media. Also, parameters provided by the API of Twitter regarding the users posting the tweets to be analysed are the number of followers, number of followed users, number of account posts and account verification.

It must be annotated that the set of tweets under analysis were emerged from a natural disaster that took place four years ago. Hence, results originated from parameters related to the account of users are not fully representative, since they could have enormously changed since then. This fact also concerns the analyses of next chapter (Chapter 6).

5.2.1. Retweets

A retweet is defined as a reposted or forwarded message on Twitter.

The following figure (Figure 33) displays the average number of retweets per tweet, segmented by level.

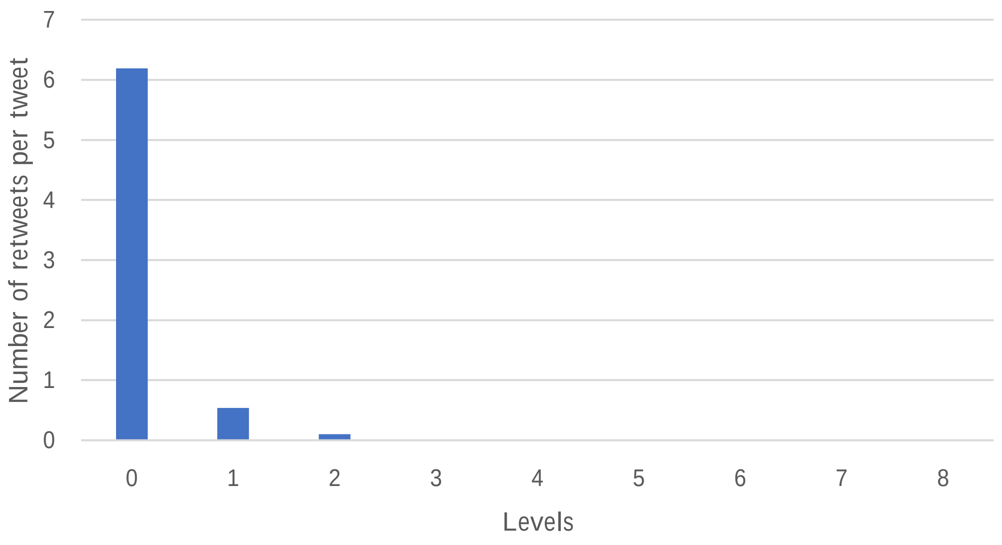


Figure 33: Average number of retweets per tweet, for each level.

The results obtained accentuate that retweets tend to mainly occur in messages belonging to lower levels, being L0 the one which causes the higher number by a huge difference. An interpretation of them is that tweets that carry a more general content, meaning that it is not addressed to a specific user and therefore not personalised, are more probable to convey relevant information and consequently more likely to be diffused.

The average number of replies per tweet, segmented by its corresponding number of retweets, is introduced in the following figure (Figure 34). The retweet's ranges have been selected with the aim of obtaining similar sample size for each range. Results stress an increasing trend on the number of replies when the higher the number of retweets is, direct consequence of the tweets exposure.

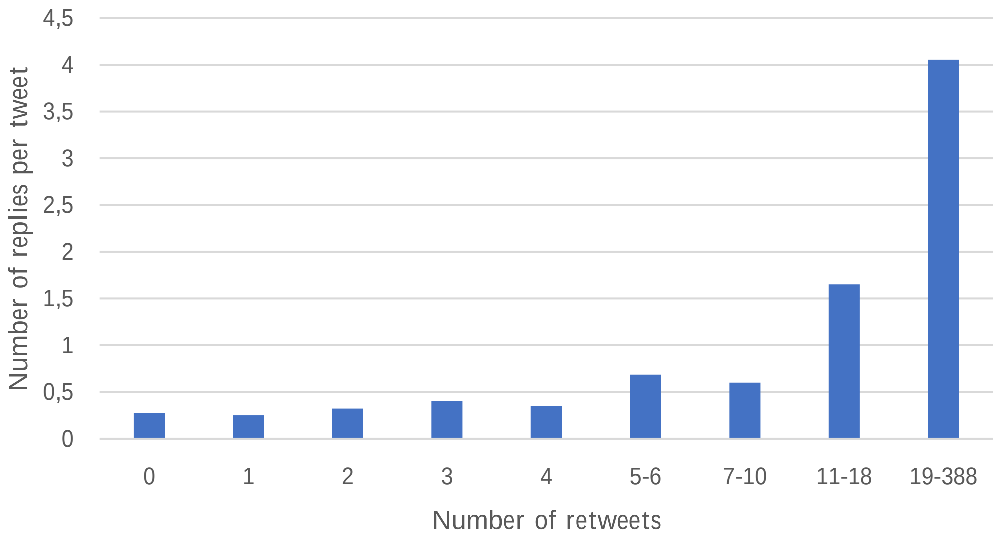


Figure 34: Average number of replies per tweet, segmented by the number of retweets of these tweets (with ranges considered).

The maximum level reached by cascades is analysed taking into account the average number of retweets in their L0 tweet in Figure 35. In the graphic is highlighted that the fact of gathering more retweets in the first tweet of the cascade helps to reach further levels than L1. This circumstance is direct consequence of the fact that higher retweet numbers tend to originate more replies, as described in the analysis resulting from Figure 27.

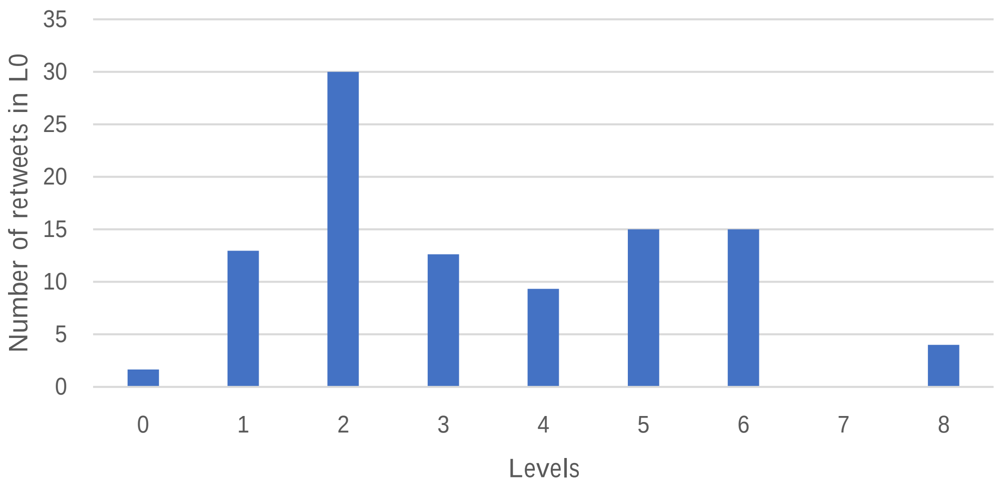


Figure 35: Average number of retweets per tweet considering just the set of tweets belonging to L0, for each maximum level reached by their corresponding cascades.

The following table (Table 1) displays the percentages of cascades' adopted forms, segmented by the number of retweets in their L0 tweet employing ranges.

Number of RTs in L0	Not developed	Single reply	Star reply	Star	Star to chain	Star to stars	Chain
0	88%	7%	1%	0%	2%	0%	3%
1	78%	13%	0%	3%	0%	0%	8%
2	78%	4%	0%	0%	7%	0%	11%
3	64%	29%	0%	0%	0%	0%	7%
4	67%	17%	0%	0%	6%	0%	11%
5 – 6	71%	6%	0%	6%	12%	0%	6%
7 – 10	45%	25%	0%	5%	0%	0%	25%
11 -18	16%	32%	5%	21%	16%	5%	5%
>18	12%	12%	0%	18%	29%	12%	18%

Table 1: Percentage of cascades that share the same form, for each form, and segmented by the number of retweets of their corresponding L0 tweets (with ranges considered).

The outcome obtained stresses the fact that when a lower number of retweets is achieved in L0, the corresponding cascades tend to just contain 1 or 2 tweets (*Not developed* and *Single reply*). Nonetheless, when the number of retweets starts to be significant (from 7 retweets) cascades adopting chain forms emerge. Finally, cascades owning more than 18 retweets in its L0 tweet tend to develop as stars. This predominance from star forms could be explained considering that the higher exposure obtained by high retweets numbers causes more number of replies.

5.2.2. Likes

Likes are used by users to point out that tweets are of its interest.

The following figure (Figure 36) displays the average number of likes per tweet, segmented by level.

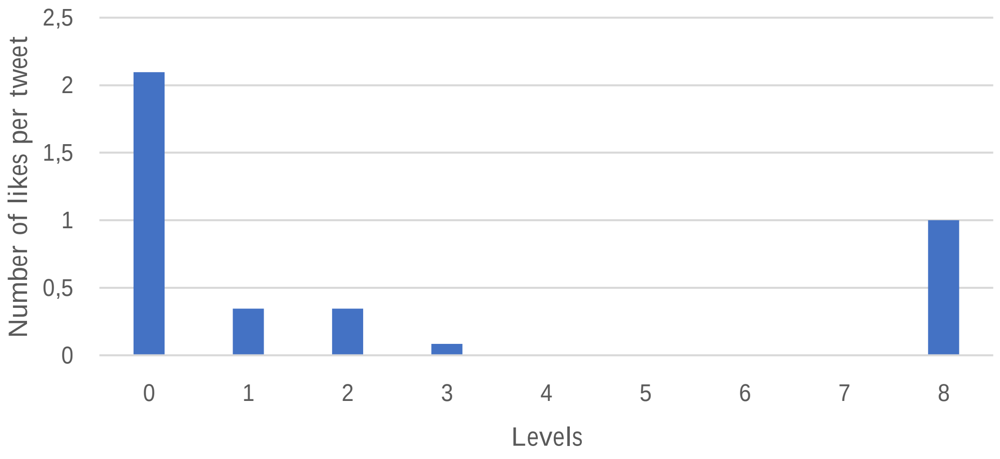


Figure 36: Average number of likes per tweet, for each level.

The results obtained can be analogously interpreted as the retweet case, since it is the content of tweets in lower levels, which is less conversational approached and usually contain more informative essence, that could cause higher number of likes.

The average number of replies per tweet, segmented by its corresponding number of likes, is introduced in the following figure (Figure 37). As in the retweet case, the likes' ranges have been selected with the aim of obtaining similar sample size for each range. Results stress an increasing trend on the number of replies the higher the number of likes is, hypothetically as consequence of the particular content of their corresponding tweets, already commented in previous analysis.

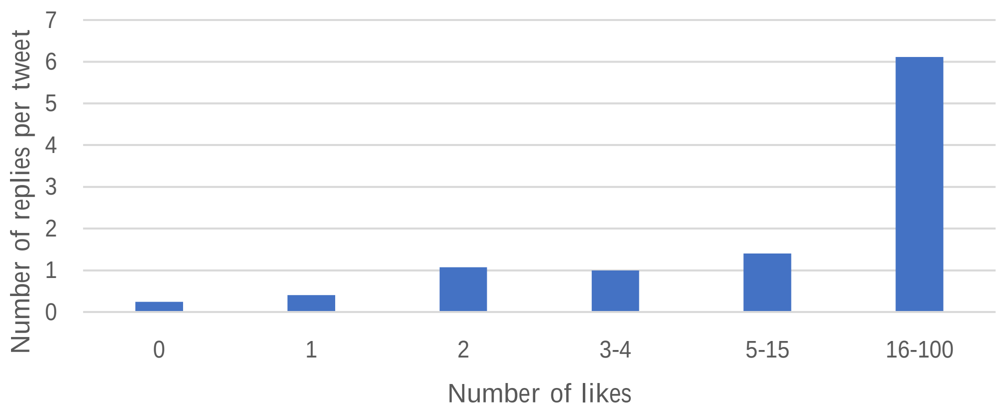


Figure 37: Average number of replies per tweet, segmented by the number of likes of these tweets (with ranges considered).

The maximum level reached by cascades is analysed taking into account the average number of likes in their L0 tweet in the figure (Figure 38) below. The explanation considered is once again analogous to the retweet case, since it is also highlighted that the fact of gathering more likes in the first tweet of the cascade helps to reach further levels than L1 (included) since the number of replies originated from them is higher.

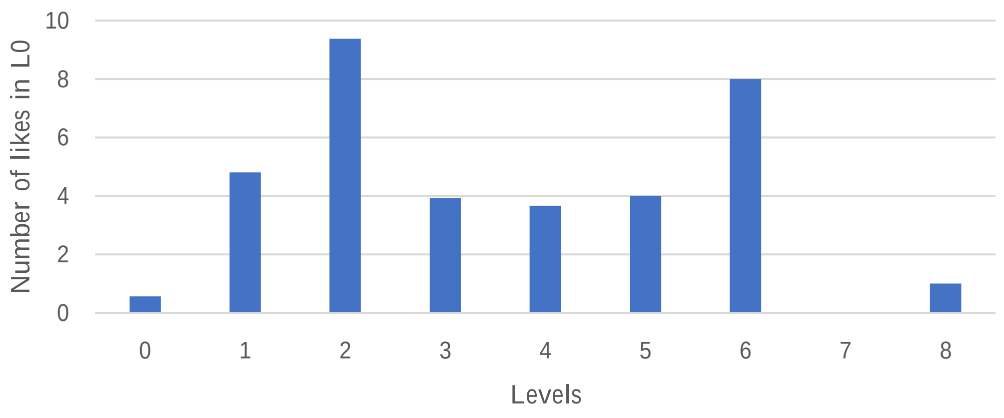


Figure 38: Average number of likes per tweet considering just the set of tweets belonging to L0, for each maximum level reached by their corresponding cascades

The following table (Table 2) displays the percentages of cascades' adopted forms, segmented by the number of likes in their L0 tweet employing ranges.

Number of likes in L0	Not developed	Single reply	Star reply	Star	Star to chain	Star to stars	Chain
0	88%	6%	1%	0%	1%	0%	4%
1	58%	22%	0%	3%	3%	0%	14%
2	50%	15%	0%	5%	20%	0%	10%
3 – 4	42%	11%	5%	16%	11%	5%	11%
5 – 15	28%	33%	0%	11%	17%	0%	11%
> 15	0%	0%	0%	25%	38%	25%	13%

Table 2: Percentage of cascades that share the same form, for each form, and segmented by the number of likes of their corresponding L0 tweets (with ranges considered).

The outcome obtained stresses the fact that when no likes are gathered in L0 tweets, their corresponding cascades tend to not further develop. However, cascades whose L0 tweets achieved likes are more likely to develop and adopt different forms, without a specific predominance of any of them.

5.2.3. Creation date

The following two figures (Figure 39 and Figure 40) are related to the creation date of the cascades, equivalent to say the creation date of their L0 tweet. Dates are segmented by days, which encompass from the 10th February 2014, the day that the activation alert started, to the 15th February 2014.

The first figure presents the average number of tweets per cascade segmented by the creation date of the cascades, while in the second figure is exposed the average maximum level reached of the corresponding cascades.

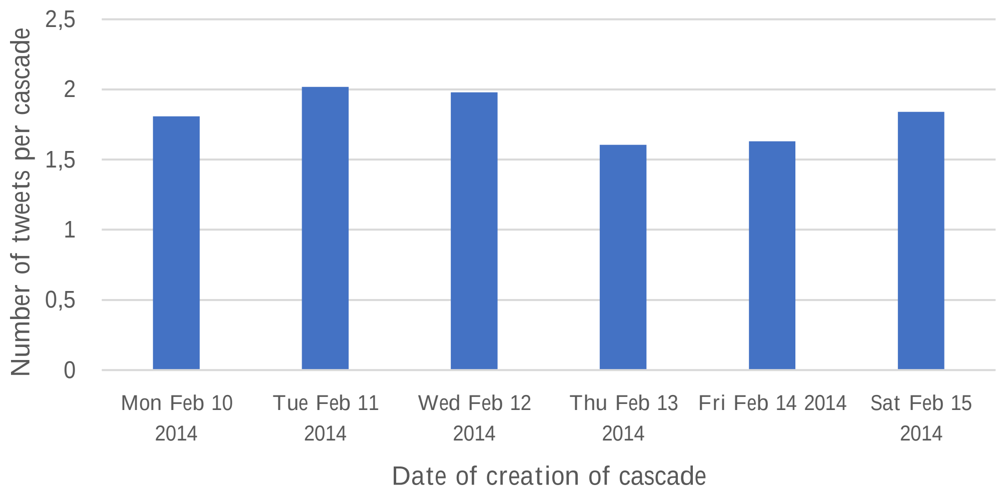


Figure 39: Average number of tweets per cascade, segmenting by the creation date of these cascades.

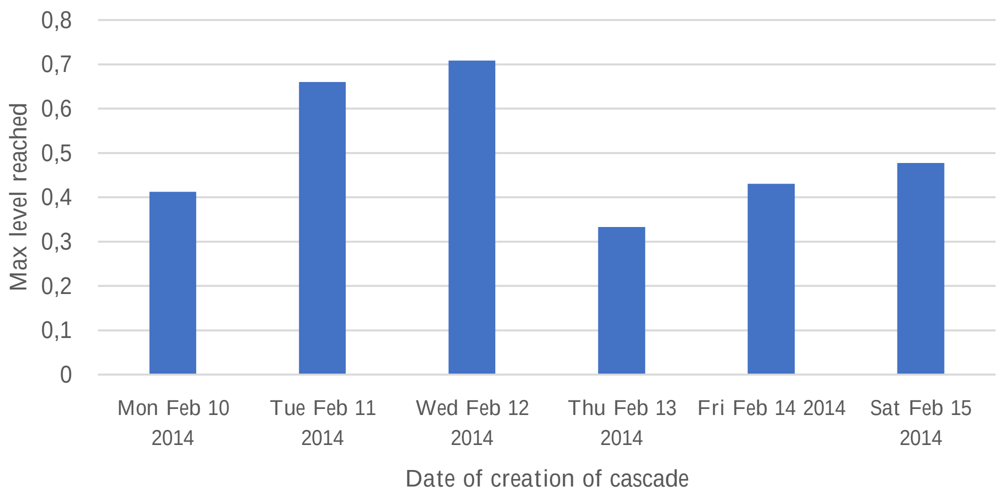


Figure 40: Average maximum level reached of cascades, segmenting by the creation date of these cascades.

The results show that no significant differences regarding the number of tweets owned by cascades and the level reached by them exist depending on the day they are created. Hence, it can be asserted that there is no correspondence between the behaviour of the cascades and the flood situation, which stabilized throughout the week.

5.2.4. Users' verified account

Verified accounts are those accounts from users considered of public interest, and therefore carry a blue badge on Twitter to lets people know their authenticity.

The following figure (Figure 41) introduces the average number of replies per tweet, segmented by level, making a comparison between reply values corresponding to tweets posted by all users and tweets posted just by users with verified accounts.

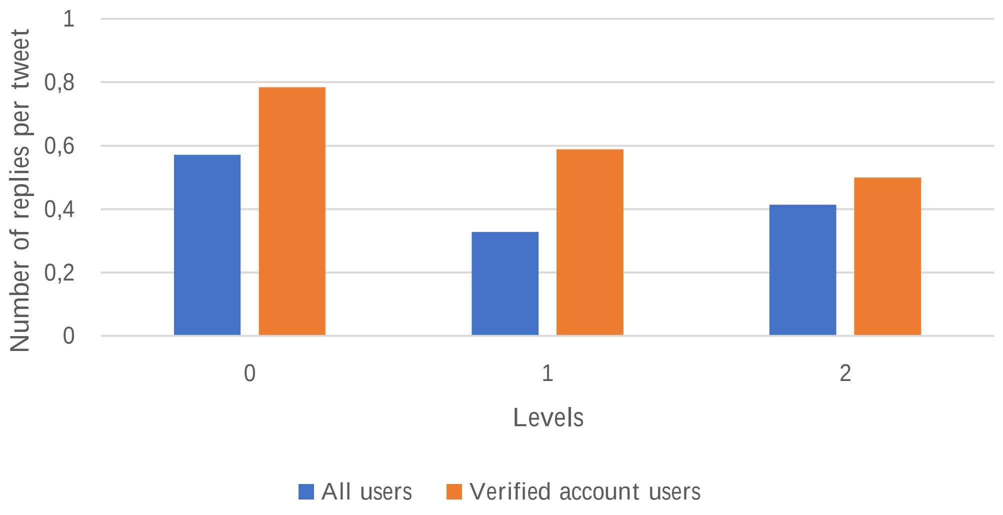


Figure 41: Average number of replies per tweet, segmenting by levels, and differentiating these tweets in the case they are posted by users with verified accounts.

As it is shown in the figure, the average number of tweets replying to the tweets posted by users with verified accounts is higher in every level. This circumstance could be explained by the fact that users with verified accounts have more followers and influence than other users, which lead to a higher exposure of their tweets.

In Figure 42, the percentage of tweets posted by users with verified accounts, segmented by level, is displayed. Results stress that these users most of the times post in L0 and scarcely they tend to reply. This fact is supported by an observation provided in (Feng et al., 2015), which reveal that highly connected users (like users with verified accounts) are exposed to an overload of messages that in turn provoke a lower probability to view or diffuse any of them. Hence, verified users can be considered as not conversational, meaning that they not look for establishing conversations and hence they feel more comfortable just originating cascades.

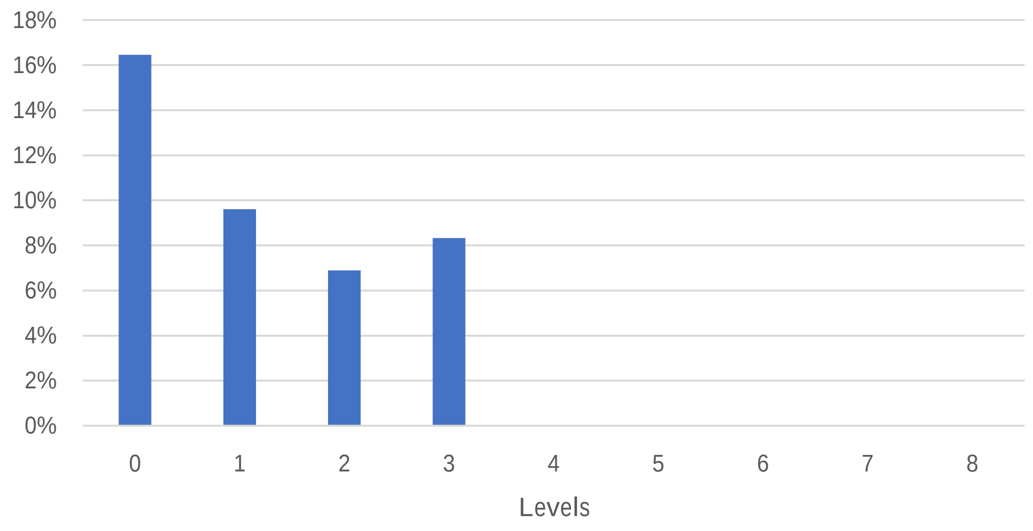


Figure 42: Percentage of tweets posted by users with verified accounts, segmenting by levels.

5.2.5. Media

Users have the possibility to include media in their tweets in form of images, videos and GIFs. Nevertheless, in the time the tweets under study were posted, Twitter just let images (and only one per tweet) to be included.

Furthermore, since this thesis is framed on a case study whose objective is rapid mapping, media becomes more important because of its primordial usage in such contexts, as already in previous chapters (Chapter 1 and 3).

The following figure (Figure 43) introduces the average number of replies per tweet, considering just L1, and making a comparison between the values corresponding to the total set of tweets and just the tweets containing media (images). It must be annotated that tweets carrying media copied from other tweets belonging to previous levels in the same cascade are evaluated as if they were not carrying it.

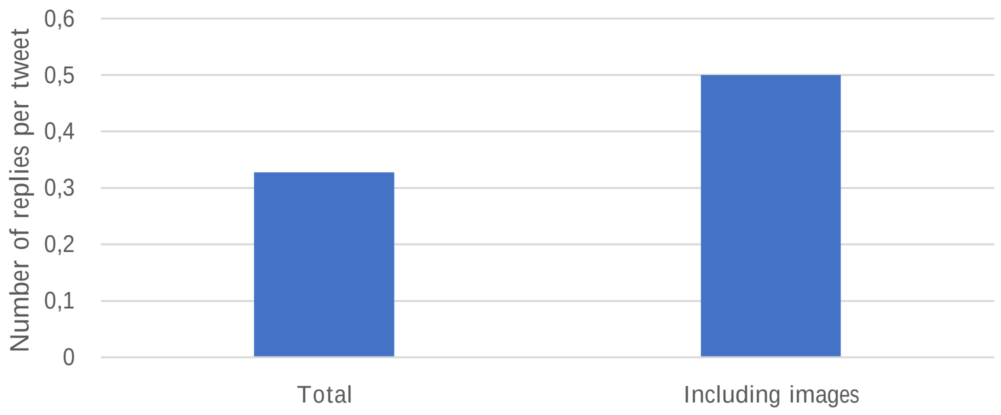


Figure 43: Average number of replies per tweet, differentiating these tweets in the case they contain images.

As it is shown in the figure, the average number of replies to the tweets containing images is higher in L1 (35% higher). This circumstance could be explained by the fact that tweets containing images tend to create a bigger effect on users, translated in higher number of replies.

In the below figure (Figure 44), the percentage of tweets containing media, segmented by level, is displayed. Results stress that tweets not belonging to L0 tend to not include media. Besides, even if a comparison with L0 is not possible to be done since the provided dataset has been created to contain only tweets geolocated and including media, it is evidenced the fact that tweets including media tend to be more posted in L0. A possible explanation to this fact is that when replying and conversating, usually users don't feel the necessity to support their messages by enclosing images into them.

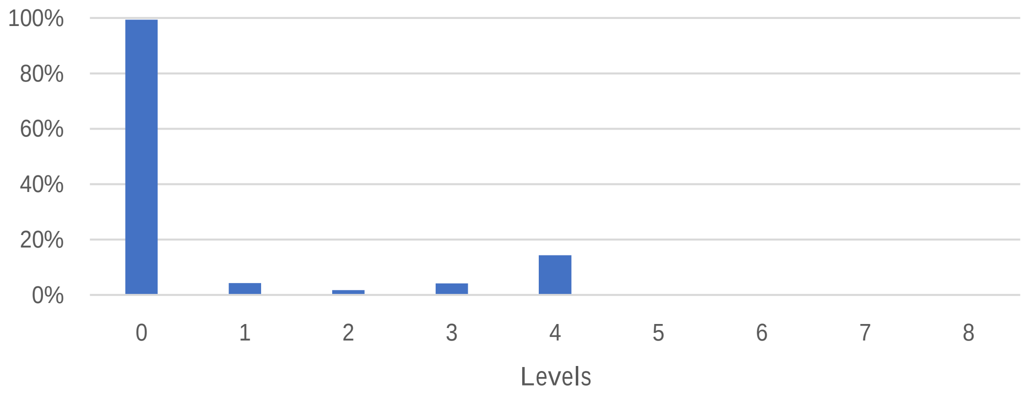


Figure 44: Percentage of tweets containing images, for each level.

Finally, the percentage of times that an *original user* post including media, is displayed in the figure (Figure 45) below. It stands out that these users don't tend to incorporate media in tweets after L0. However, an exception is shown in a tweet in L1, which is the level that these users use to add information to their L0 tweet. Hence, the only likely level that *original users* could provide posts with media, even if not usual, would be L1.

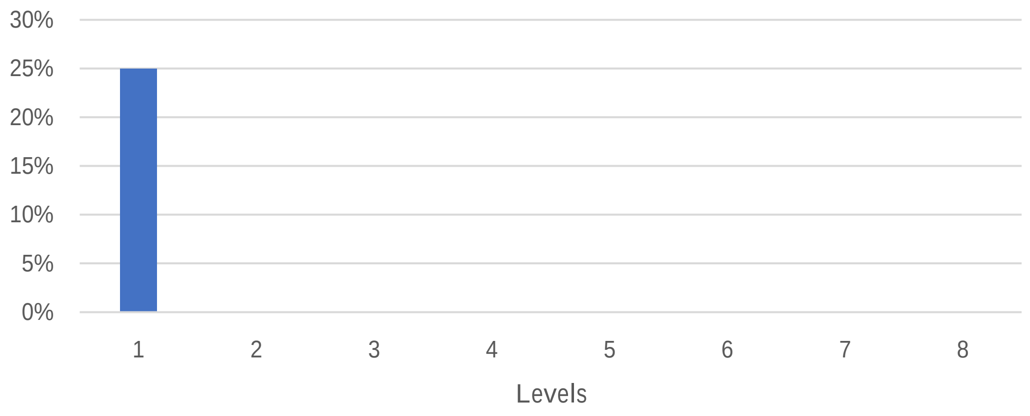


Figure 45: Percentage of tweets posted by *original users* and containing images, for each level.

5.2.6. Followers

People who follow a user, meaning these Twitter users who subscribe to the updates of that user, are called his/her followers.

The following figure (Figure 46) introduces the percentage of tweets that users just post in L0 and L1, and segmented by their number of followers using ranges. The results evidenced a weak upward trend (with the exception of the first range), meaning that users with higher followers' number tend to post more in L0 and L1 than users with fewer number. The explanation to this fact could be analogous to that of verified accounted users, meaning that users with higher followers' number feel more comfortable to tweet aiming in originating cascades than in a replying context.

In Figure 47, the average number of replies per tweet, segmented by users followers' number using ranges, is presented. As in previous cases, the followers' ranges have been selected with the aim of obtaining similar sample size for each range, although in the last ones it has been not possible to achieve.

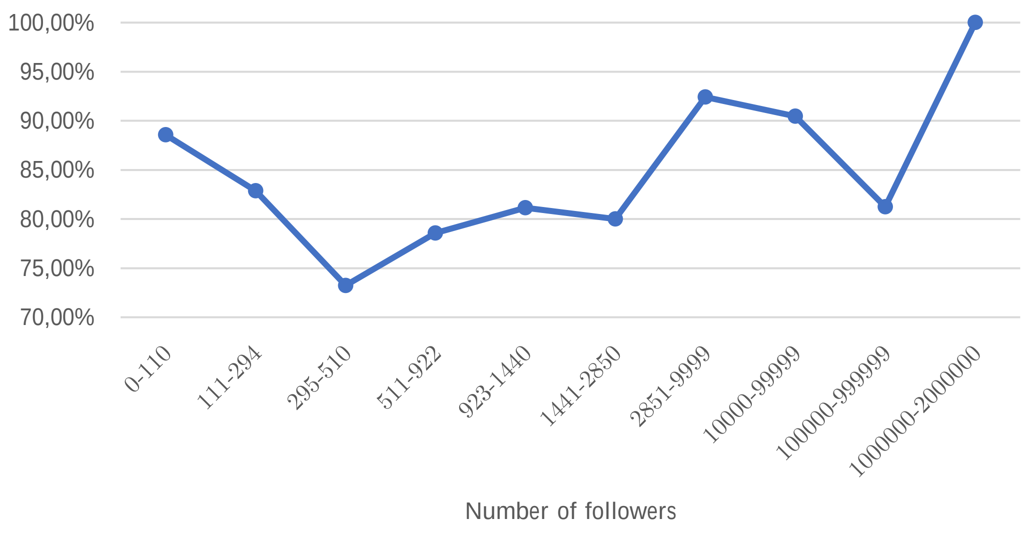


Figure 46: Percentage of tweets posted just in L0 and L1 (considering all levels), segmenting by the number of followers of the users which posted these tweets (with ranges considered).

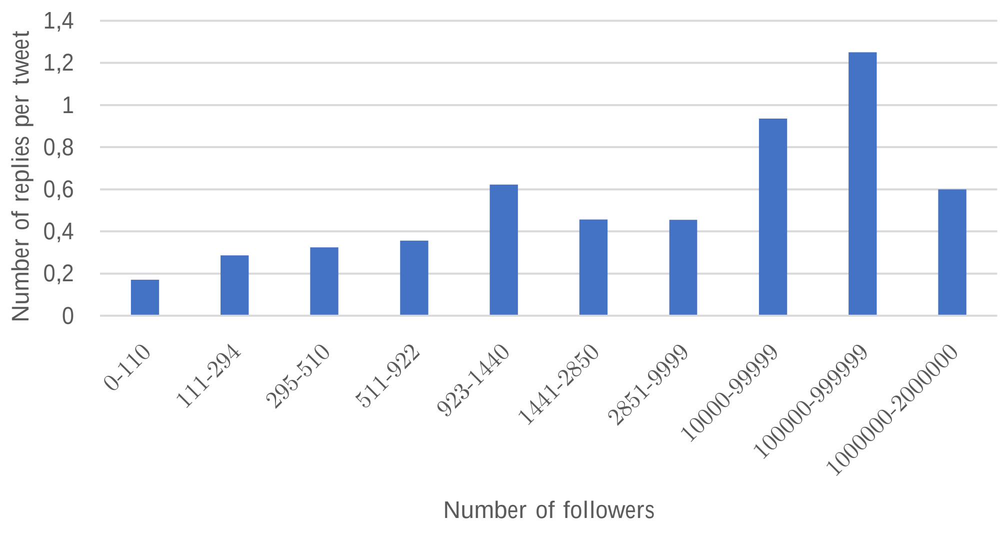


Figure 47: Average number of replies per tweet, segmenting by the number of followers of the users which posted these tweets (with ranges considered).

From the figure, it is shown an increasing tendency in the number of replies the higher the number of followers of the posting user is. This fact could be interpreted as a higher exposure of tweets lead to higher number of their replies, analogous to the retweet and verified accounted users cases.

Another figure (Figure 48) that reinforces the argument provided in the first analysis of this Section 5.2.6 is the one below. In it, the average followers' number of users that post, segmented by level, is displayed. As it is observed, users with higher followers' number tend to post more in lower levels than otherwise.

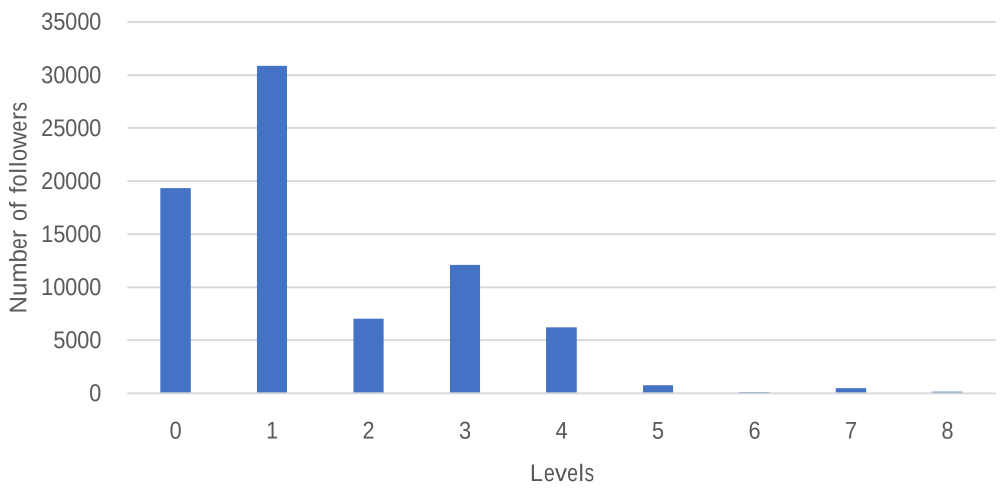


Figure 48: Average number of followers of the posting users, segmenting by the level whose posts belong to.

5.2.7. Followed users

A user following people, meaning that that user is subscribed to the updates of these users, are called the followed users of him/her.

The following figure (Figure 49) introduces the percentage of tweets that users just post in L0 and L1, and segmented by the number of users that they follow using ranges. The results don't show any relationship between the number of users followed and the number of posted tweets in the first two levels. Hence, it is asserted that the number of users followed is not a significant factor in this aspect.

In Figure 50, the average number of replies per tweet, segmented by the number of users followed by the users posting using ranges, is presented. As in previous cases, the ranges have been selected with the aim of obtaining similar sample size for each range. From the figure, it is shown that there is not a correlation between the number of replies and followed users.

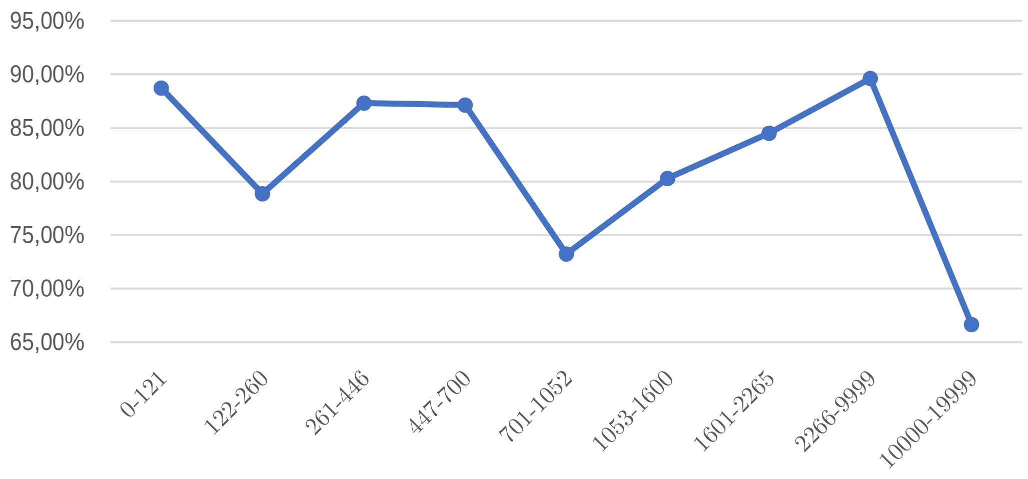


Figure 49: Percentage of tweets posted just in L0 and L1 (considering all levels), segmenting by the number of followed users of the users which posted these tweets (with ranges considered).

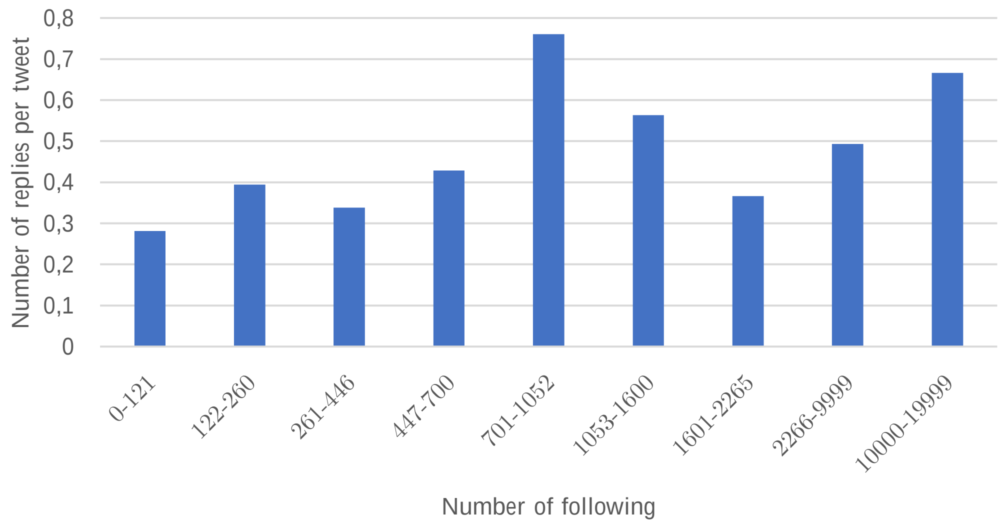


Figure 50: Average number of replies per tweet, segmenting by the number of followed users of the users which posted these tweets (with ranges considered).

Another figure (Figure 51) that supports the argument provided in the first analysis of this Section 5.2.7 is the one below. In it, the average followers' number of users that post, segmented by level, is displayed. As it is observed, a correlation between the number of users followed and the level posted is not existent.

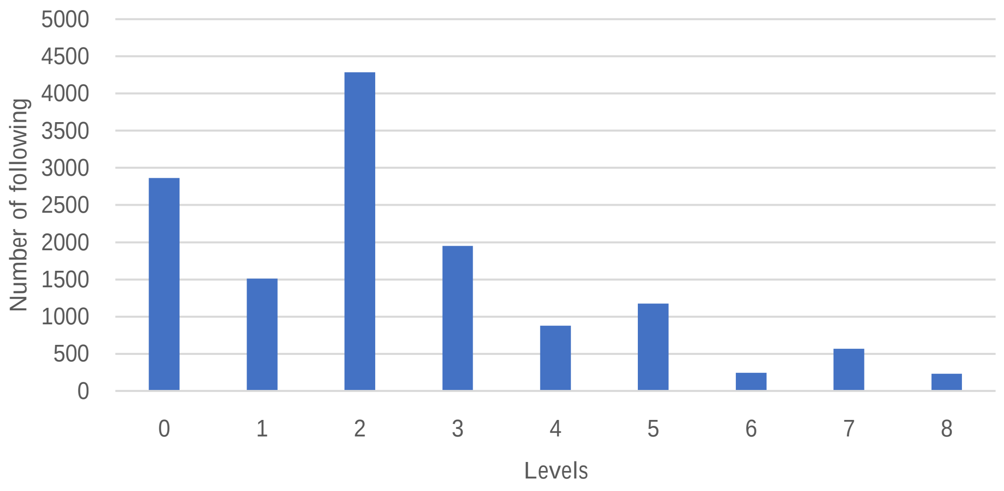


Figure 51: Average number of followed users of the posting users, segmenting by the level whose posts belong to.

5.2.8. Users' account posts

The total number of posts registered in an account of a user is called the user's number of account posts.

The following figure (Figure 52) introduces the percentage of tweets that users just post in L0 and L1, and segmented by the number of user account's posts using ranges.The results don't show any relationship between the number of users' account posts and the number of posted tweets in the first two levels. Hence, it is asserted that the number of users followed is not a significant factor in this aspect.

In Figure 53, the average number of replies per tweet, segmented by the number of account's posts by posting users using ranges, is presented. As in previous cases, the ranges have been selected with the aim of obtaining similar sample size for each range. From the figure, it is shown that doesn't exists a clear tendency on the number of replies regarding the number of account posts of the users posting.

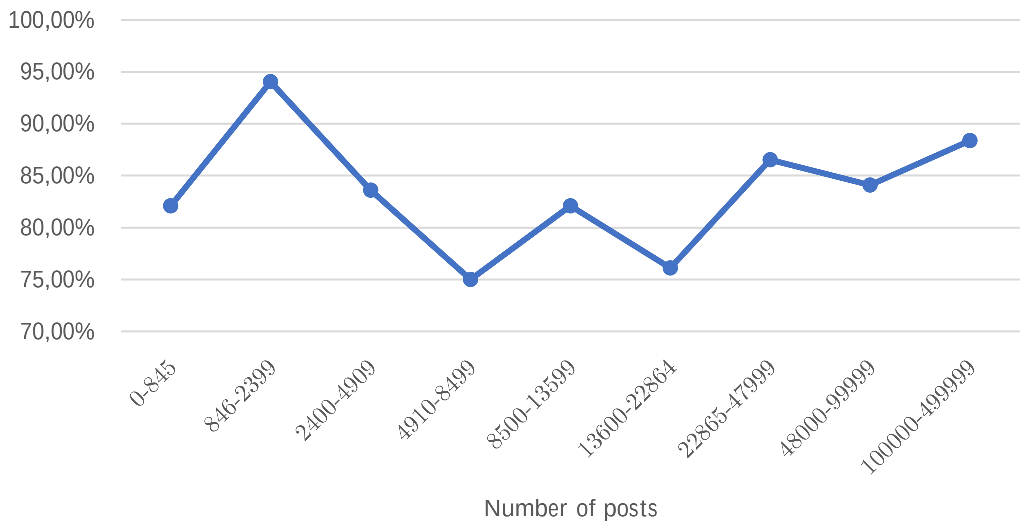


Figure 52: Percentage of tweets posted just in L0 and L1 (considering all levels), segmenting by the number of account posts of the users which posted these tweets (with ranges considered).

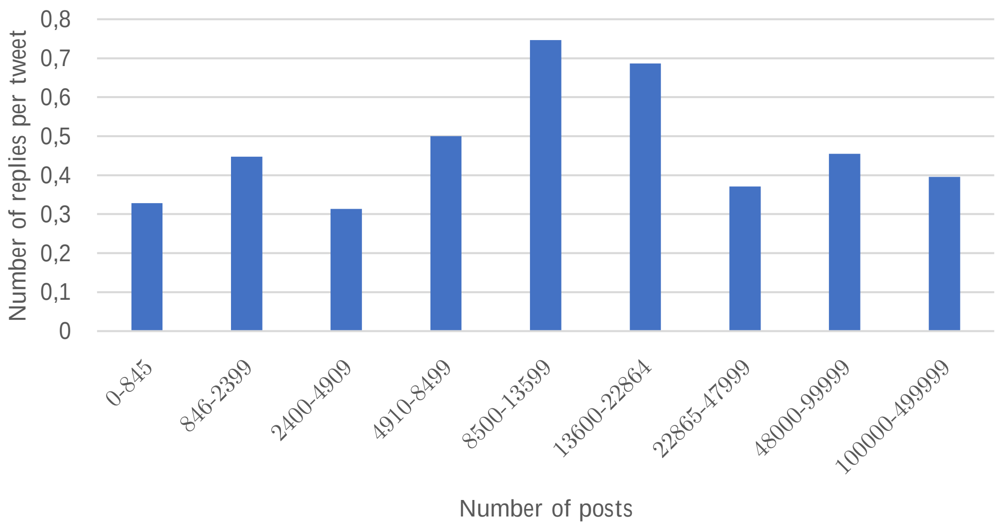


Figure 53: Average number of replies per tweet, segmenting by the number of account posts of the users which posted these tweets (with ranges considered).

Another figure that reinforces the argument provided in the first analysis of this Section 5.2.8 is Figure 54. In it, the average number of account's posts by users that post, segmented by level, is displayed. As it is shown, a correlation between the number of posts and the level posted is not existent, even if a weak downward tendency is observed.

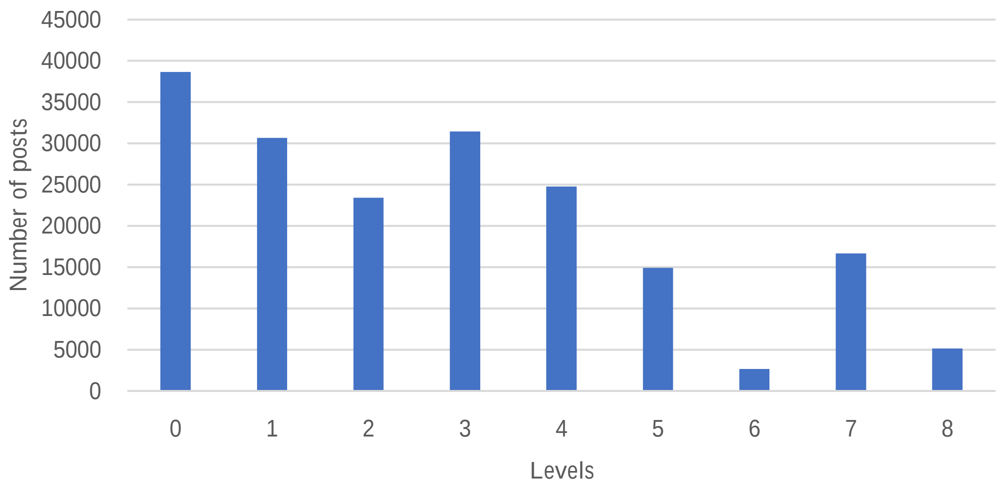


Figure 54: Average number of account posts of the posting users, segmenting by the level whose posts belong to.

To sum up, the main findings of this section have been the following:

- Retweets and likes from users tend to occur in lower levels (mainly in L0). Furthermore, tweets owning higher number of retweets and likes tend to receive more replies. Also, the fact of gathering more retweets and likes in L0 tweets helps their corresponding cascades to reach further levels than L1.

- Cascades with higher number of retweets in their L0 tweet tend to develop adopting a star-related form.

- Tweets posted by users with verified accounts receive a higher number of replies. Furthermore, these users most of the times post in L0, meaning that scarcely tend to reply.

- Tweets including media (images) receive a higher number of replies in the first level (L1). Furthermore, tweets not belonging to L0 tend to not include media.

- Users with higher number of followers tend to post more in lower levels (L0 and L1). Furthermore, tweets posted by these users receive a higher number of replies.

Chapter 6

Reply's tweets relevance analysis

This chapter makes use of the data collected to produce several analyses considering the annotated relevance of tweets. Consequently, the purpose of this chapter is twofold:

- To gain insight on the relationship between the relevance of reply tweets in natural disaster contexts and the structure of cascades, as well as with the values of the different tweet parameters such as number of retweets and likes.

- To identify patterns in reply tweets (from L1 in cascades) to be able to classify them according to their relevance in the case of rescue operations.

Regarding the second objective, a section dealing with patterns recognition based on tweets' texts analyses aiming to cluster replies according to six established types is included (Section 6.3).

In the following sections, first it is analysed the structure of cascades considering tweets' relevance, including predicative approaches, in Section 6.1. On the other hand, in Section 6.2 the possible relationships between tweets' parameters and cascades' structure according to the relevance of the tweets are put in manifest. Finally, in Section 6.3 an analysis with regards to tweets' texts aiming to extract patterns is performed.

Figure 55 presents the number of tweets in L0 collected in the database, and therefore the number of existent cascades, and segment them into relevant and non-relevant ones. It is useful to highlight the precision of the original dataset provided (66,13 %) considering the annotation criterion followed and included in the Appendix (Section i).

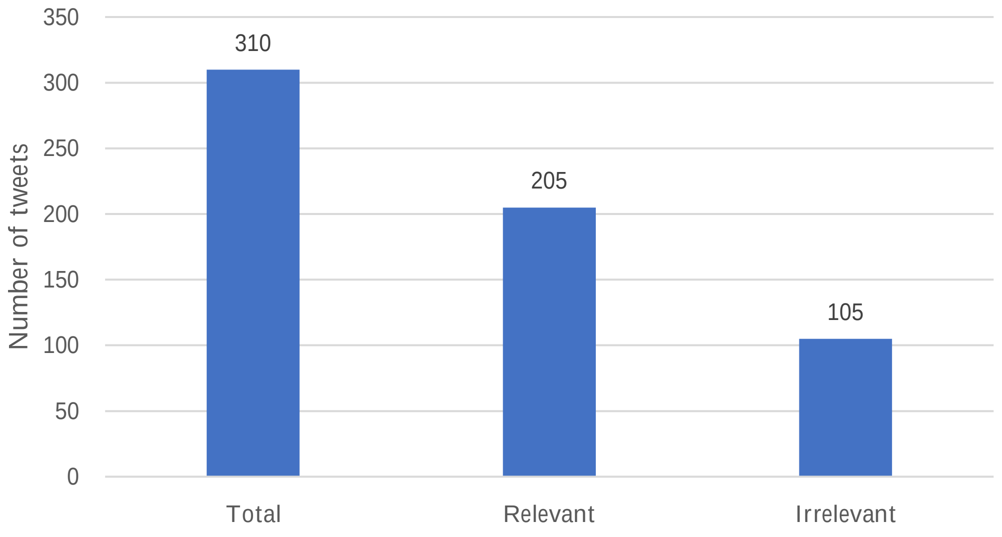


Figure 55: Number of L0 tweets according to their relevance.

Furthermore, the number of cascades considered relevant from the total number of cascades is displayed in the following figure (Figure 56). It must be annotated that a cascade is considered relevant if at least one tweet belonging to it is relevant.

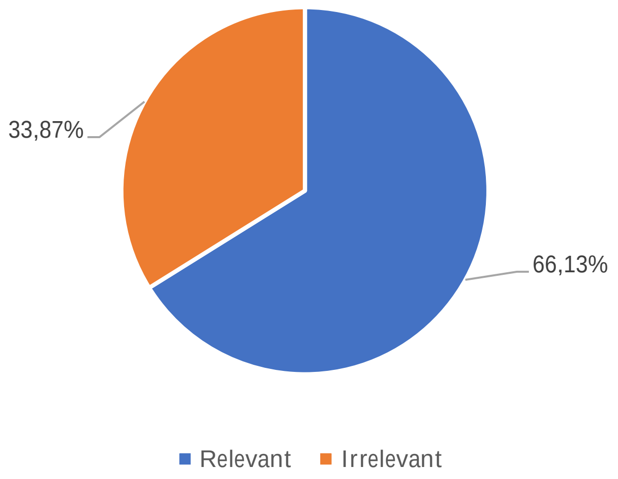


Figure 56: Percentage of cascades according to their relevance.

Results show that the percentage of relevant cascades (66,13 %) is equal to the one of relevant tweets in L0, meaning that no relevant tweets are found from L1 (included) in cascades whose L0 tweet is irrelevant. Hence, the usefulness in future natural disasters of examining tweets in cascades whose L0 is non-relevant is considered as null. Consequently, their consideration when extracting relevant reply messages is not convenient, since they would just lead to a reduction in terms of precision. Nevertheless, the analyses presented in this chapter most of the times introduce results taking into account both cases: the total of cascades and the set of cascades annotated as relevant.

Furthermore, a relevance analysis considering the different parts of tweets is exposed in the table below (Table 3). As described in the annotation criterion used in Chapter 4 and included in the Appendix (Section i), a tweet will be annotated as relevant if at least one of its three parts (text, media and URL) is annotated as relevant.

It could be seen that tweet's texts are the main source of relevance since they are relevant in 217 of the 227 relevant tweets (95,59 %) in the dataset (considering all the levels). Hence, it is verified that text is an excellent *recaller* of tweets in classifying contexts. This fact is explained considering that texts most of the times carry information about relevant images.

Only text	Only URL	Only media	Text and URL	Text and media	URL and media	Text, URL and media	TOTAL
21	1	9	0	192	0	4	227

Table 3: Number of relevant tweets, differentiated by the part/s (text, URL, media) within the tweet that make them relevant. For example, tweets corresponding to the "Text and media" category are relevant because their text and media parts are relevant.

Apart from texts, it is also highlighted that in a majority of relevant tweets their enclosed image is relevant (90,3 %). However, using images as a classifier parameter is not as simple as in the text case for IT technical complexities, which not allow to distinguish them according to their relevance. Furthermore, special attention to tweets carrying images must be placed, since they are important to be extracted for rapid mapping purposes.

Finally, URL's from tweets are not considered useful to be examined due to their low rate of relevance (just in 5 tweets, of which 4 are combined with text and images relevance).

6.1. Cascade structure characteristics

Tweet's relevance taking into consideration cascade structure characteristics such as levels and adopted forms are analysed in this section.

The following figure (Figure 57) presents the total number of relevant tweets in the dataset from L1 (included), segmented by level.

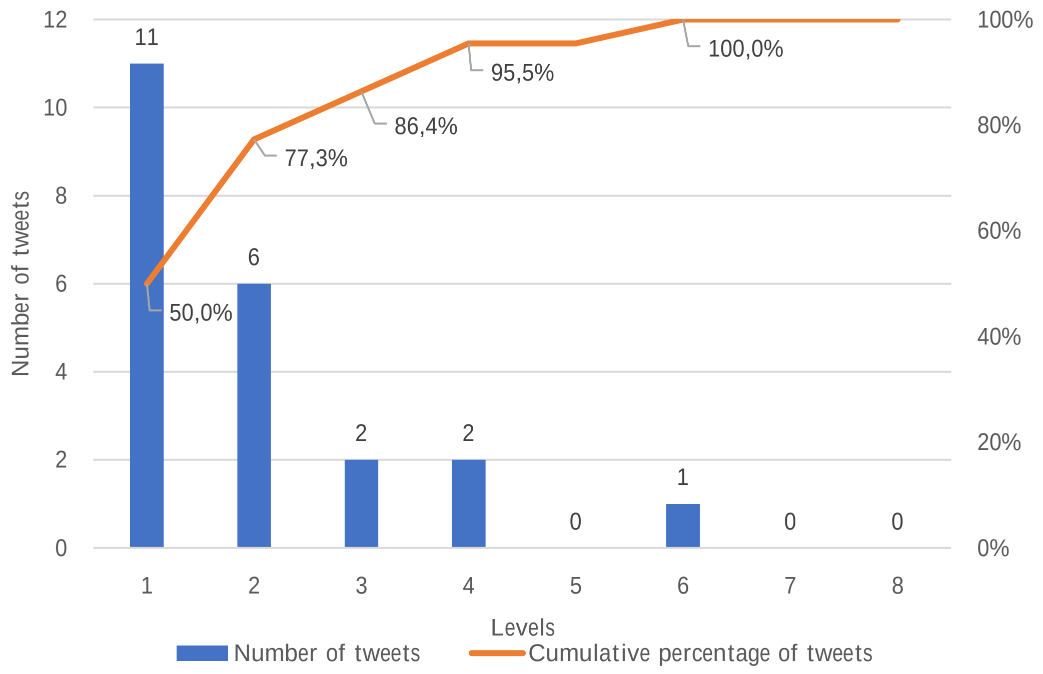


Figure 57: Columns represent the number of relevant tweets for each level. Line represents the accumulated percentage of relevant tweets considering the set of relevant tweets (from L1) for each level.

As it is shown, there is a decreasing trend on the number of relevant tweets the higher the level is. This fact can be almost completely explained by the decreasing trend of the total number of tweets (relevant and irrelevant), but also by the reduction of informative content the higher the level is. Furthermore, that after L3 just a 13,64% of relevant tweets are found, being that value of 4,55% after L4.

The following two figures (Figure 58 and Figure 59) aim to accentuate the just-mentioned aspect of the observed descent in terms of relevant content from L3 (non-included). On the one hand, the first figure (blue line) shows the inversed accumulated percentage of relevant tweets considering the overall relevant tweets in the dataset.

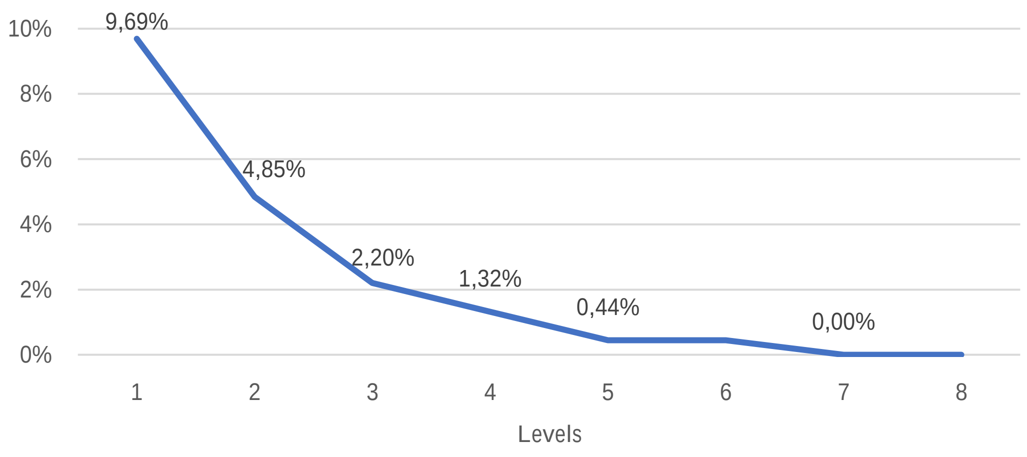


Figure 58: Inversed cumulative percentage of relevant tweets considering the overall set of relevant tweets (from L0), for each level.

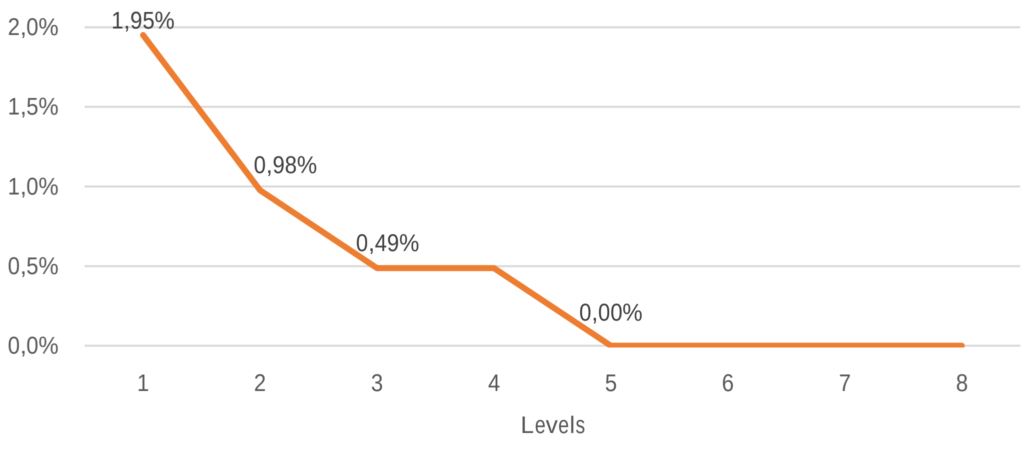


Figure 59: Inversed cumulative percentage of tweets containing relevant images considering the overall set of relevant tweets (from L0), for each level.

On the other hand, the second figure (orange line) shows the inversed accumulated percentage of relevant images considering the overall relevant images in the dataset. Results accentuate the low rate of relevant content (1,32% and 0,49%) found after L3.

In the next table (Table 4) is displayed the probability of obtaining a relevant reply tweet per cascade. The probability is computed in four different scenarios: considering all the cascades contained in the dataset or only considering relevant cascades; and considering reply tweets belonging to all the levels (L1 to L8) or only considering reply tweets belonging to the first three (L1 to L3).

	Between L1 and L8 (included)	Between L1 and L3 (included)
Number of relevant tweets	22	19
Probability considering all cascades (310 cascades)	7,10%	6,13%
Probability considering only relevant cascades (205 cascades)	10,73%	9,27%

Table 4: Number of tweets and percentage of relevant tweets considering different set of tweets. The first column considers the relevant tweets belonging to levels between L1 and L8 (both included). The second column just considers the relevant tweets belonging to levels between L1 and L3 (both included).

Results registered in the second row considers all the cascades whether if they are relevant or not (310 cascades), presenting a probability of finding relevant tweets of a 7,10%. Nevertheless, if it just considered cascades that are relevant, assuming an idealistic situation in which the initial retrieval of L0 tweets has been done with complete precision, it increases to 10,73%. This result can be interpreted alike as: it is needed the examination of more than 10 cascades to find 1 relevant reply tweet.

As in the previous analysis has been stressed the convenience of finishing the examination of tweets until L3 (included), the percentages taking into consideration this possibility are computed. The results obtained (6,13% and 9,27%) are extremely close to those which consider the replies belonging to all the levels, and therefore it can be concluded that the convenience exposed is definitely true.

The following table (Table 5) presents, for each number of observed replies (0, 1, 2 and 3), the percentage of relevant tweets replied. The analysis takes into account three different levels (from L1 to L3), and just the tweets within relevant cascades.

Number of replies	Level 1	Level 2	Level 3
0	13,56%	7,14%	9,09%
1	6,90%	33,33%	16,67%
2	50,00%	-	-
3	-	0%	-

Table 5: Percentage of relevant tweets replied considering the overall set of tweets in levels 1-3, segmented by the number of replies of these tweets.

The obtained outcomes stress the non-existent correlation between the number of replies originated from a tweet and its relevance, since no cell contains a value higher than 50%. This circumstance just can be interpreted as that relevant tweets don't cause a distinct effect on the replying motivation by other users compared to those irrelevant.

In the next table (Table 6), percentages of relevant tweets considering the posting user are displayed. Hence, in the case that the user that post the tweet in a cascade is the same that posted the first tweet in L0, meaning that he/she is the originator of the cascade, it is considered that the user is an *original user* (True row). The analysis just takes into account the first two levels (L1 and L2) as in L3 there are no tweets posted by *original users*, and just the tweets within relevant cascades.

Original User	Level 1	Level 2
False	9,30%	5,88%
True	75,00%	20,83%

Table 6: Percentage of relevant tweets either posted by *original users* ("True" row) or not ("False" row) considering the overall set of tweets in levels 1-2.

Results show that tweets posted by *original users* tend to be more relevant than those posted by other users. Considering L1, the probability of a tweet to be relevant is more than 8 times higher when posted by them, although just 4 tweets were obtained to be posted by this kind of users, relatively small sample to extract conclusions. Nevertheless, the probability concerning L2, even if lower than in L1, is 3,54 times higher when posted by them, which continues being a remarkable high value. Besides, in this level the sample size belonging to *original users* increases to 24 tweets and therefore results can be considered as representative. Hence, the consideration of this users' condition (*original user*) in identification patterns processes is convenient.

In order to stress the usefulness of texts to classify tweets already mentioned in previous analyses (see Table 3), the next figure (Figure 60) is displayed. In it, the percentage of relevant tweets containing in turn relevant texts is shown to be higher than 80% for every level unless L4 (which just have 2 relevant tweets). If the percentage considering tweets belonging to all levels together from L1 (included) is computed, it results to be 86,36% (19 of 22 tweets), which is deemed as a very high value. Hence, the convenience of considering texts to classify tweets is finally considered as certain.

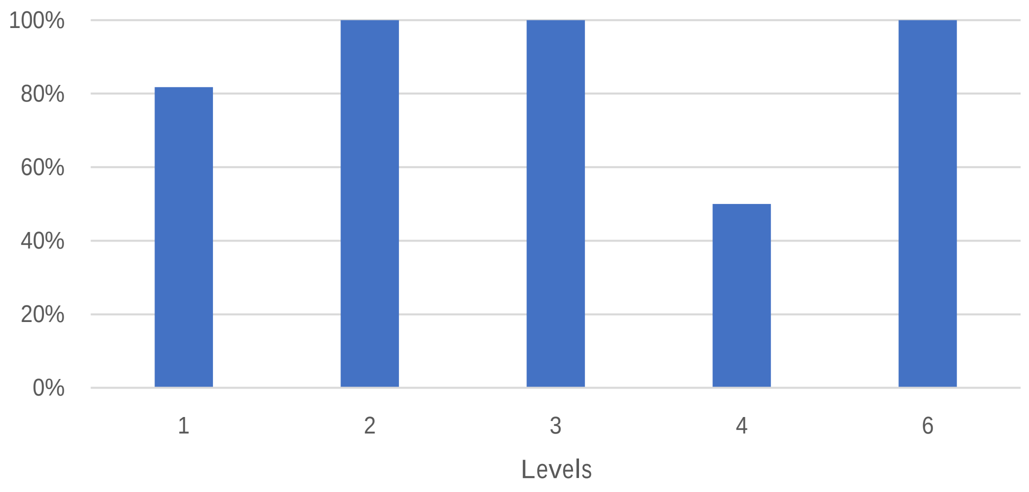


Figure 60: Percentage of relevant tweets whose text is relevant, for each level.

Finally, a study replicating five analyses provided in Section 5.1 has been performed segmenting the cascades according to their relevance (relevant and irrelevant). Nevertheless, no significant differences between both cases have been found for any of the analyses, and therefore the 5 figures supporting the study have not been included in this section but in the Appendix (Section ii).

6.1.1. Predictive analysis

Predictive analyses have been performed with the objective of analysing the possibility that the characteristics of either single or a set of tweets in one level can determine the characteristics of the tweets belonging to subsequent levels. In our case, the tweet characteristic under evaluation is its relevance. Such analyses aim to find patterns so that in future cases of study the extraction of the tweets in a level could be focussed on certain behaviours of cascades considering a set of tweets already annotated in another level.

The following figure (Figure 61) deals with the probability of finding relevant tweets in subsequent levels in the case that relevant tweets are found in a specific level. More precisely, it presents the probability of finding at least 1 relevant tweet in subsequent levels (from level i+1 to the last one) in a cascade, when in a particular level (level i) is found at least 1 relevant tweet[22]. This analysis is helpful to understand whether it is valuable to continue the examination of a cascade depending if in at any level is found a relevant tweet or not.

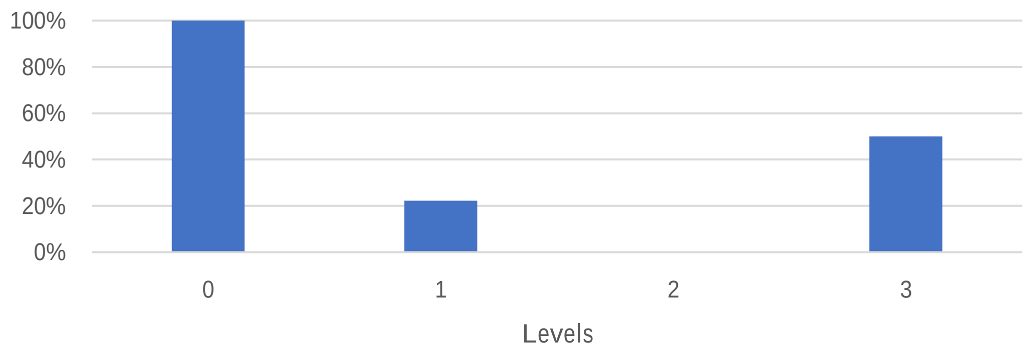


Figure 61: Probability for a cascade of finding relevant tweets in subsequent levels if in that level there are relevant tweets too, for each level. It is considered that in a level there are relevant tweets if at least 1 tweet belonging to that level is relevant for each cascade. Only the set of cascades reaching subsequent levels are considered, for each level.

[22] A relevant level in a cascade is defined as a level that owns at least 1 relevant tweet.

Results show that L0 behaves differently from the other ones studied (1,2 and 3). In L0, as already mentioned, all the cascades containing in that level a relevant tweet are 100% guarantee to find existent relevant tweets in subsequent levels, if reached. However, for levels 1, 2 and 3 this analysis is not considered appropriate to use for the purpose that has been devised because of the low probability values obtained.

To understand the probability of obtaining a relevant tweet with just considering if the previous one has been relevant or not, it is performed the study presented in the following tables (Table 7, Table 8 and Table 9), which comprises from L0 to L3. Therefore, the analysis doesn't take into consideration the set of tweets in a specific level within cascades but is based on the study of single tweets.

In it, the percentage of relevant tweets in each level derived by every possible path[23] (or chain), which consider the relevance of the tweets in previous levels, is displayed. It must be annotated that tweets whose relevance has not been determinate since they were considered copied[24] tweets cannot be clustered to any path, and therefore do not take part of the analysis.

		POSSIBLE PATHS		
	Level 0	Irrelevant	Relevant	TOTAL
Level 1	Tweets relevant (number of tweets)	0	11	11
	Percentage of relevance (Path / TOTAL)	0%	100%	

Table 7: Percentage of relevant tweets found in each of the possible paths (or chains). In this table there are considered levels 0-1.

[23] A path (or chain) is defined as the sequence of relevant-irrelevant tweets in each of the levels considered (number of paths = 2^k,where k=levels considered in the chain – 1).

[24] Tweets that are a copy of other tweets in a cascade were not annotated during the annotation process described in Section 4.2, as indicated in the annotation criterion included in the Appendix (Section i).

		POSSIBLE PATHS				
	Level 0	Irrelevant		Relevant		TOTAL
	Level 1	Irrelevant	Relevant	Irrelevant	Relevant	
Level 2	Tweets relevant (number of tweets)	0	0	5	0	5
	Percentage of relevance (Path / TOTAL)	0%	0%	100%	0%	

Table 8: Percentage of relevant tweets found in each of the possible paths (or chains). In this table there are considered levels 0-2.

		POSSIBLE PATHS								
	Level 0	Irrelevant				Relevant				TOTAL
	Level 1	Irrelevant		Relevant		Irrelevant		Relevant		
	Level 2	Irr	Rel	Irr	Rel	Irr	Rel	Irr	Rel	
Level 3	Tweets relevant (number of tweets)	0	0	0	0	1	0	1	0	2
	Percentage of relevance (Path / TOTAL)	0%	0%	0%	0%	50%	0%	50%	0%	

Table 9: Percentage of relevant tweets found in each of the possible paths (or chains). In this table there are considered levels 0-3.

The results obtained stress again the convenience of just focusing on cascades developed from relevant tweets in L0 to extract relevant tweets in subsequent levels, since all the relevant replies in every level originate from relevant messages in L0.

Also, that relevant tweets in L2 only come from non-relevant tweets in the previous level as observed in Table 8. This circumstance is explained by the fact that the 5 relevant tweets found in L2 are all consequence of asked questions in L1, which are considered irrelevant. Nevertheless, since 93,78% of the tweets belonging to L1 are irrelevant (90,43% considering just relevant cascades) this finding is not useful to be considered as a pattern to identify relevant tweets belonging to L2.

Finally, the only two relevant tweets in L3 come from non-relevant tweets in L2. The reason is analogous to the just mentioned: they are consequence of questions in L2. A conclusion to this fact is that relevant tweets belonging to levels 2 and 3 are never found in the same chain of a cascade. A figure exposing the structure of the two unique cascades in the dataset containing relevant tweets in L3 is presented below (Figure 62).

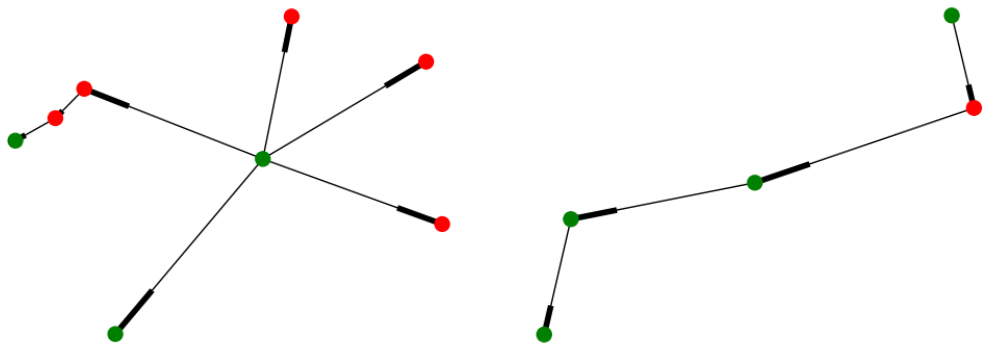


Figure 62: Graphs of the two only cascades containing a relevant tweet in L3. Nodes represent tweets, edges represent reply connections. Colours in nodes represent tweets' relevance: green for relevant tweets; red for irrelevant tweets. Edges' bold part represent arrows.

6.2. Tweet's parameters values in cascades

Tweet's relevance taking into consideration their parameters' values are analysed in this section. The obtained values from the API server considering tweet's parameters can be divided into two types: infinite integer and binary values. This segmentation is therefore used in this section, since each type requires of different analyses to be performed.

Furthermore, all studies regarding the section only consider tweets belonging to levels 1 and 2 within relevant cascades, and tweets not being a copy of tweets from another level in a same cascade. The first condition (levels) have been taken considering that sample sizes (number of tweets) from subsequent levels from L2 are too small to take their results as representative, since segmentation by levels is performed. The second condition is based on the fact that a study considering replies belonging to irrelevant cascades, those which don't contain any relevant tweet, just would have increased the noise on the obtained results because of the non-existence of relevant tweets. Nonetheless, previous analyses have indirectly demonstrated the inconvenience of studying tweets in non-relevant cascades (see Table 4 and Figure 56) for pattern identification purposes. The third condition is taken since copied tweets lack of annotation (neither relevant nor irrelevant).

6.2.1. Infinite integer parameters values

Parameters whose values are of an infinite integer type are treated in this section. Hence, parameters regarding retweets' number and likes' number of tweets, as well as followers' number, number of users followed and account's posts number by the users posting tweets are analysed.

Analyses are performed considering the percentage of relevant tweets obtained for each possible parameter value (or range of values) and the sample size for each corresponding parameter value (or range of values).

Hence, analyses are represented in graphs which share the same format. On the one hand, the data represented in blue circles regarding each parameter value correspond to their percentage of relevant tweets. Hence, this percentage could be seen as the precision that the parameter value has to obtain relevant tweets. Consequently, the aim will be focused on identifying parameter values whose percentage takes extreme digits (values very close to 0% and 100%) in order to be considered as potential classification patterns. In this sense, the average precision corresponding to each parameter has to be firstly evaluated to be able to assess the extremity of each obtained precision. On the other hand, the data highlighted in black numbers above the blue circles represent the sample size regarding each parameter value. Hence, this size could be seen as the reliability (or representativeness) of the obtained precision, meaning that precisions of parameters' values owning large sample sizes are much more probable to be obtained in other datasets than those owning smaller samples, whose associated precision can entail huge variations depending on the dataset examined. Then, the larger the sample size, the better the reliability and vice versa. Consequently, not only the objective is focused on evaluating precisions, but also have to take into account that its corresponding sample size is large enough.

In the case concerning the number of retweets and likes, the higher parameter value obtained is of 16. Hence, being 16 a low value, sample sizes have been found for most of the values and therefore there is no need to use ranges. However, if no samples are obtained regarding a specific value, its blue circle has not been, obviously, represented. Furthermore, it will be observed in the following figures that the values of these two parameters in L1 and L2 obey an exponential distribution in terms of sample sizes, meaning that most of the tweets take lower values, circumstance that negatively affect the effectiveness of the analyses aiming to identify patterns to be used subsequently as classifiers.

Nevertheless, in the case concerning the followers' number, the number of users followed and the account's posts number by the users posting tweets, parameters values obtained are highly varied and very few times repeated. Hence, there is need to use ranges to be able to gather an enough sample size to perform subsequent analyses. As in previous studies, the ranges have been established aiming to obtain similar sample sizes in each one so, unlike the previous case, exponential distributions will not be obtained.

6.2.1.1. Level 1

6.2.1.1.1. Retweets

In the next figure (Figure 63) is represented the precision and sample size for every obtained value regarding the number of retweets of tweets within relevant cascades belonging to L1. Average precision in this level is 12,22% (11 relevant tweets from a set of 90 tweets).

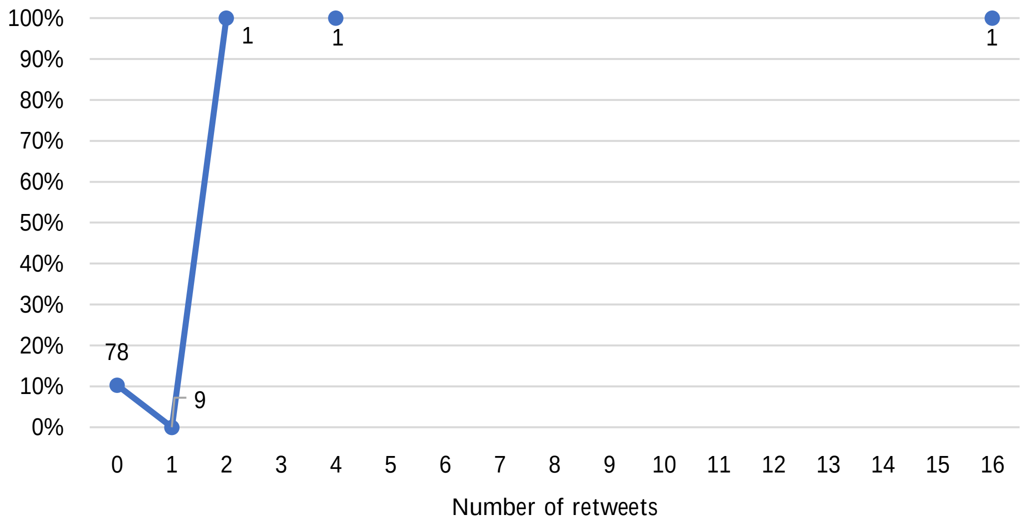


Figure 63: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of retweets of tweets belonging to L1.

The results show that 0 and 1 retweets' numbers entail a low precision (combined is equal to 9,1 %) and have a large sample size (87 samples). However, this precision is not low enough to be considered as a pattern, since it is very close to the average precision, consequence of not having the tweets more dispersed through the values. Retweets' numbers from 2 entail a high extreme precision, although their small sample size (3 samples) doesn't allow them to be considered as pattern too.

6.2.1.1.2. Likes

In the next figure (Figure 64) is displayed the precision and sample size for every obtained value regarding the number of likes of tweets within relevant cascades belonging to L1.

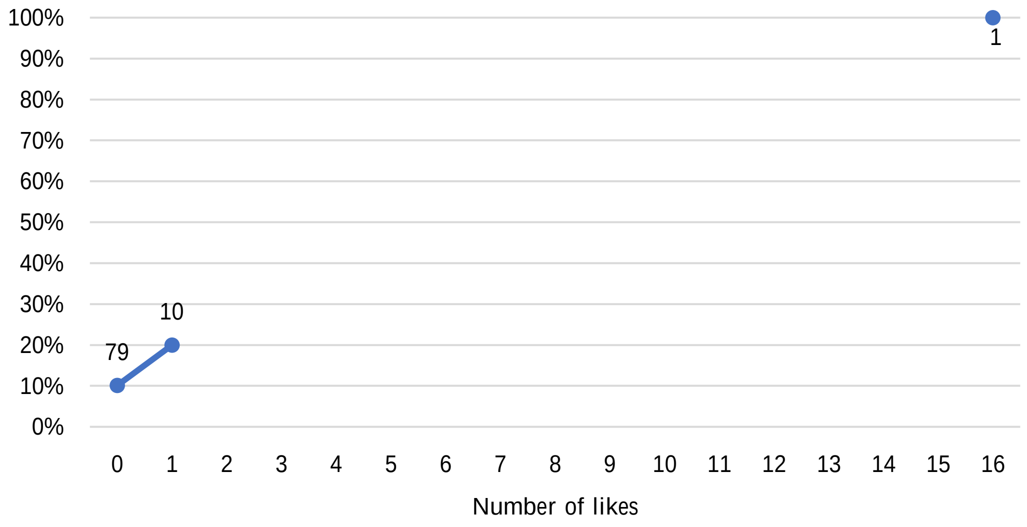


Figure 64: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of likes of tweets belonging to L1.

The results show that 0 likes' number entail a precision similar to the average, consequence of having collected the majority of the tweets. Likes' number equal to 1 entail a higher precision compared to the average, although it is not extreme enough to be considered as pattern. Finally, it is obtained a relevant tweet corresponding to a value of 16, impossible to consider since it represents a single sample.

6.2.1.1.3. Followers

In the next figure (Figure 65) is displayed the precision and sample size for every obtained value's range regarding the posting user's number of followers within relevant cascades belonging to L1.

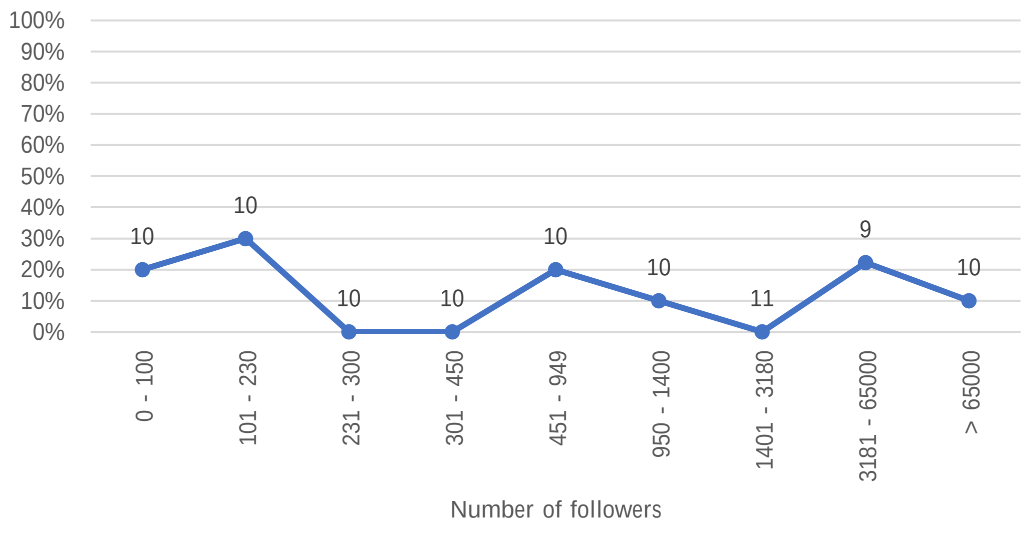


Figure 65: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of followers of the users that posted tweets belonging to L1 (with ranges considered).

The results show that there are ranges on which their corresponding precision has taken 0% (extreme), and others on which their precision is much higher than the average one (even if not extreme). Nonetheless, the blue line in the graphic highlights that a trend between ranges doesn't exist (sinusoidal form), meaning that the precisions obtained highly depend on the established ranges and a variation of their setting would carry an enormous change on the results. Hence, this parameter cannot be considered to identify patterns.

6.2.1.1.4. Followed users

In the next figure (Figure 66) is represented the precision and sample size for every obtained value's range regarding the posting user's number of followed users within relevant cascades belonging to L1.

The results obtained can be analogously interpreted as the previous analysis (number of followers). Even if values lower than 240 take higher precisions than higher values (but not extreme), it is highlighted the lack of existence of a trend regarding the ranges since the blue line take a sinusoidal form, and consequently precisions largely depend on the ranges set. Hence, also this parameter cannot be considered to identify patterns.

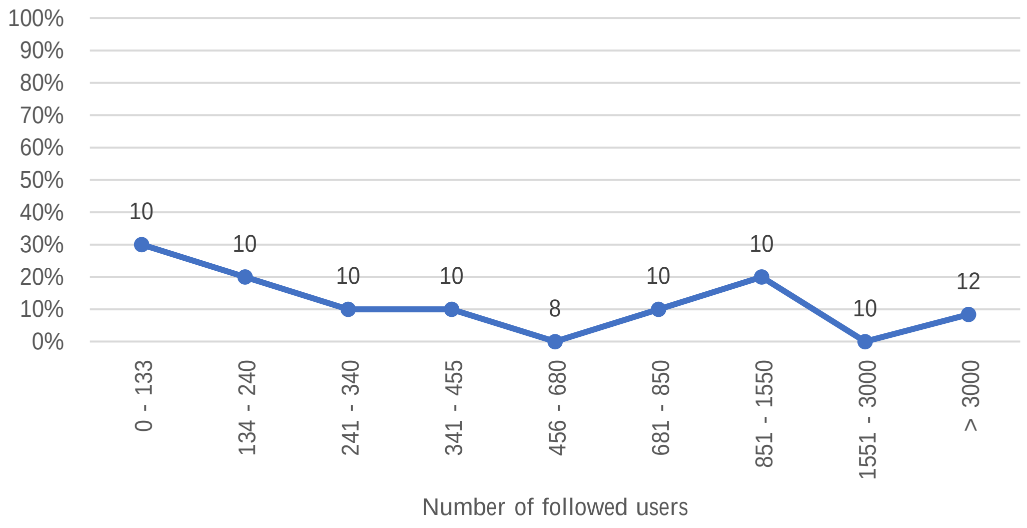


Figure 66: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of followed users of the users that posted tweets belonging to L1 (with ranges considered).

6.2.1.1.5. Users' account posts

In the next figure (Figure 67) is represented the precision and sample size for every obtained value's range regarding the posting user account's posts number within relevant cascades belonging to L1.

The results obtained are very similar to the two-previous analysis. Even if values contained in the first range (lower than 600) take a remarkable higher precision than in other ranges, it cannot be considered as extreme (40 %). Furthermore, it is highlighted the lack of existence of a trend throughout the ranges. Hence, also this parameter cannot be considered to identify patterns.

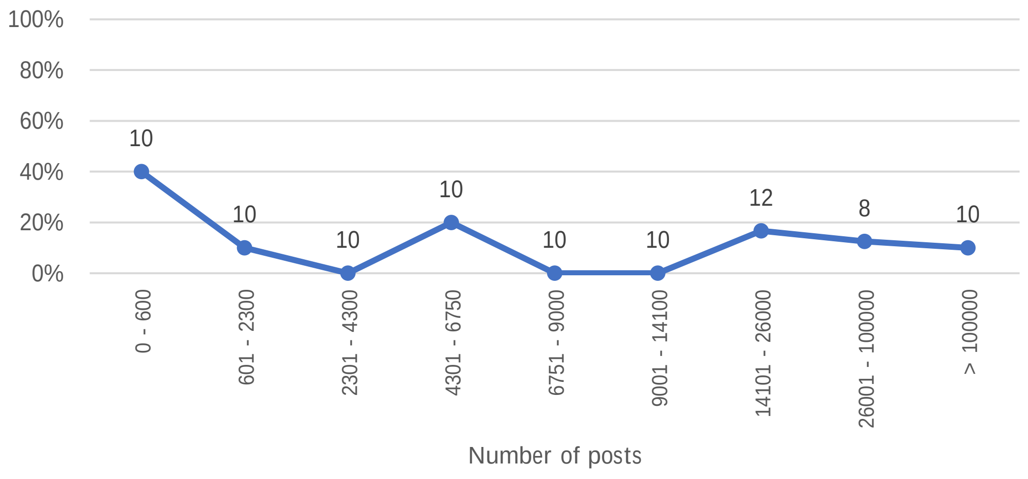


Figure 67: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of account posts of the users that posted tweets belonging to L1 (with ranges considered).

6.2.1.2. Level 2

6.2.1.2.1. Retweets

In the next figure (Figure 68) is represented the precision and sample size for every obtained value regarding the number of retweets of tweets within relevant cascades belonging to L2. Average precision in this level is 14,63 % (6 relevant tweets from a set of 41 tweets).

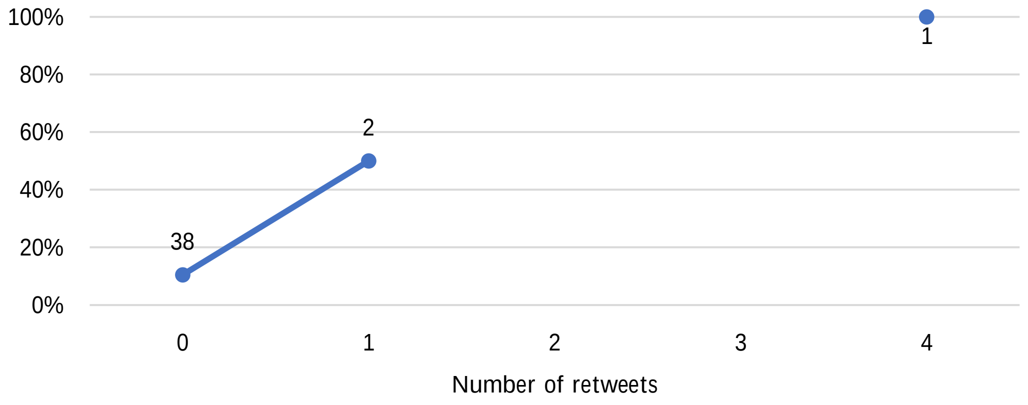


Figure 68: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of retweets of tweets belonging to L2.

The results show that 0 retweets' number entail a precision similar to the average, consequence of having collected the majority of the tweets. Retweets' numbers from 1 entail a remarkable higher precision compared to the average (66,7 %). However, the fact that only 3 samples take retweets' numbers higher than 0 doesn't allow to consider them as a possible pattern.

6.2.1.2.2. Likes

In the next figure (Figure 69) is displayed the precision and sample size for every obtained value regarding the number of likes of tweets within relevant cascades belonging to L2.

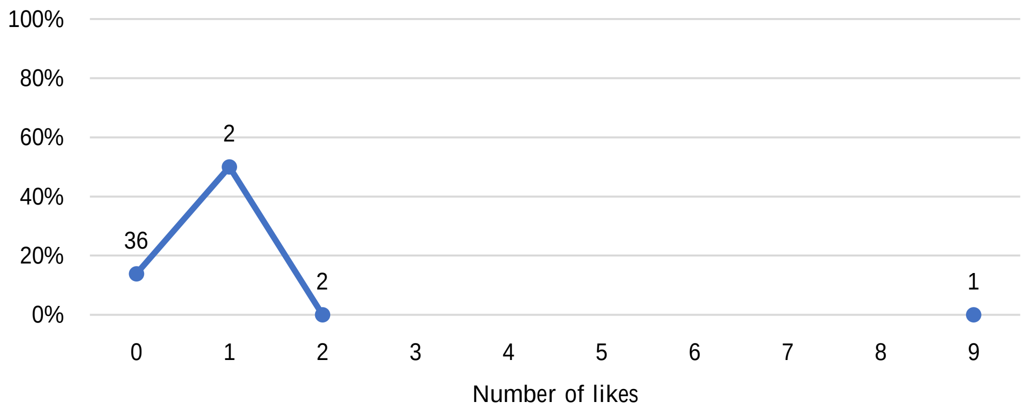


Figure 69: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of likes of tweets belonging to L2.

The results show that 0 likes' number entail a precision similar to the average consequence of having collected the majority of the tweets. Likes' number equal to 1 entail a higher precision compared to the average, although higher likes' numbers reveal zero precision, highlighting the lack of a possible trend. Hence, also this parameter cannot be considered to identify patterns.

6.2.1.2.3. Followers

In the next figure (Figure 70) is displayed the precision and sample size for every obtained value's range regarding the posting user's number of followers within relevant cascades belonging to L2.

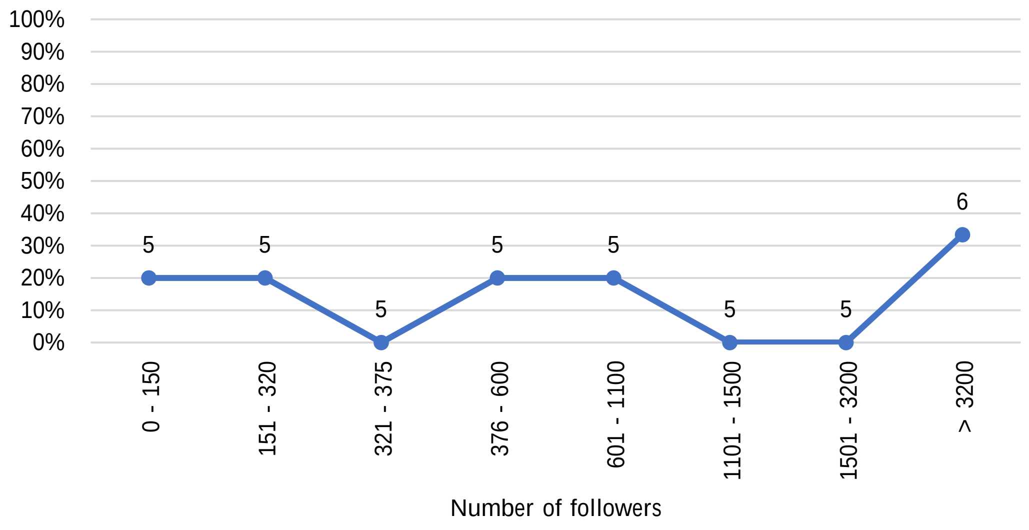


Figure 70: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of followers of the users that posted tweets belonging to L2 (with ranges considered).

Analogously to the results obtained in L1 for this same parameter, they show that there are ranges on which their corresponding precision has taken 0% and others on which their precision is similar to the average one. Nevertheless, there is not any range precision able to exceed a 35%. Furthermore, a hypothetical trend is not observed. These two facts carry that this parameter could not be considered to identify patterns too.

6.2.1.2.4. Followed users

In the next figure (Figure 71) is represented the precision and sample size for every obtained value's range regarding the posting user's number of followed users within relevant cascades belonging to L2.

The results show that there are ranges on which their corresponding precision has taken 0% (extreme), and others on which their precision is much higher than the average one (even if not extreme). Nonetheless, the blue line in the graphic highlights that a trend between ranges doesn't exist (sinusoidal form), meaning that the precisions obtained highly depend on the established ranges and a variation of their setting would carry an enormous change on the results. Hence, this parameter cannot be considered to identify patterns.

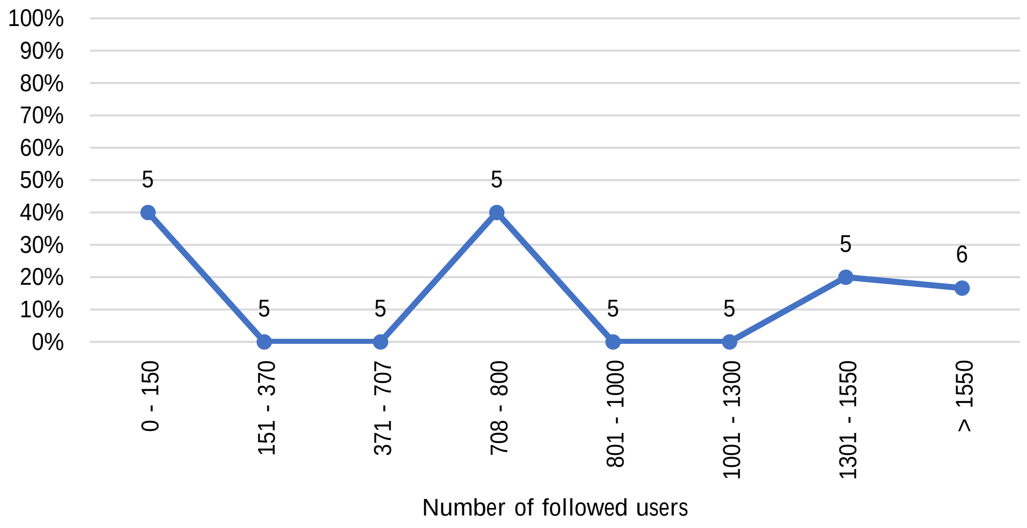


Figure 71: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of followed users of the users that posted tweets belonging to L2 (with ranges considered).

6.2.1.2.5. Users' account posts

In the next figure (Figure 72) is represented the precision and sample size for every obtained value's range regarding the posting user account's posts number within relevant cascades belonging to L2.

The results show that precisions for each range are very similar to the average precision, with the exception occurring in the range of posts' numbers between 3000 and 8000 posts, whose tweets are irrelevant. Nonetheless, a linear trend to be used to identify patterns doesn't exist, since these ranges carrying zero precision are found in the middle of the overall sample of tweets. Hence, this parameter cannot be considered to identify patterns.

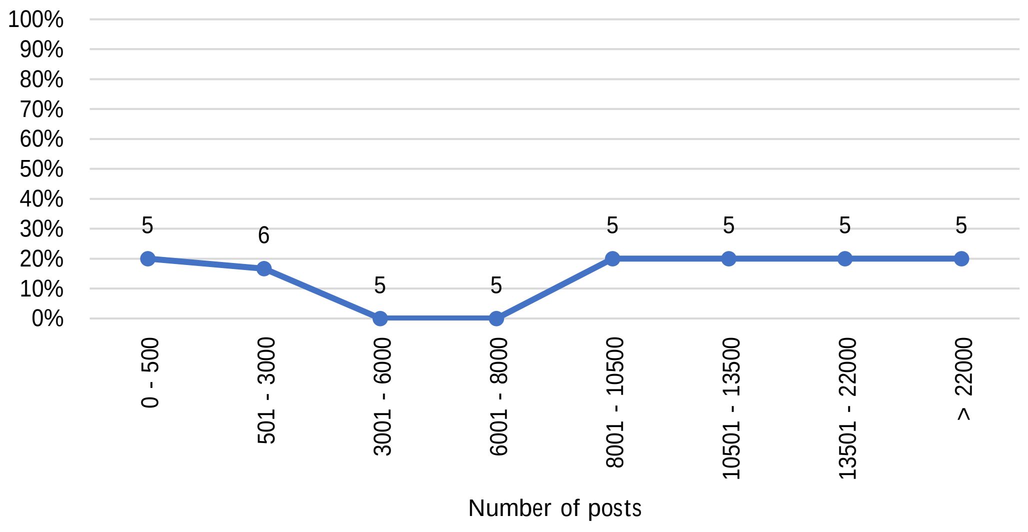


Figure 72: Percentage of precision (line) and number of relevant tweets (sample size) corresponding to the number of account posts of the users that posted tweets belonging to L2 (with ranges considered).

6.2.2. Binary parameters values

Parameters whose values are of a binary type are treated in this section. Hence, parameters regarding users' verified accounts, the inclusion of media in tweets, the use of the GPS posting location, and the type of device used to post are analysed.

Analyses are performed considering the percentage of relevant tweets obtained for the two parameter values, the sample size for each corresponding parameter value, and the significance of parameters.

As in the previous section (Section 6.2.1), the percentage could be seen as the precision that the parameter value owns to obtain relevant tweets. Consequently, the aim will be focused on identifying parameters values whose percentage takes extreme digits. In this sense, the average precision corresponding to each parameter has to be firstly evaluated to be able to assess the extremity of each obtained precision. Again, sample size could be seen as the reliability (or representativeness) of the obtained precision. Consequently, not only the objective is focused on evaluating precisions, but also have to take into account that its corresponding sample size is large enough. Therefore, the smallest sample size corresponding to a value in each parameter will be displayed. Finally, significance compares both precisions, by dividing them, in a parameter to highlight the difference between values, and therefore to analyse the parameter convenience to be used as classifier.

Consequently, the aim will be focused on detecting parameters with higher significance values.

In the case of the parameter regarding the type of device used when posting, the distinction is made considering computers and mobile phones. Computer device type encompasses the following values extracted from the API server by Twitter: *Twitter for websites, Twitter for Windows, Twitter for Mac* and *Twitter web client*. Mobile phone device type encompasses the following values: *Mobile web, Twitter lite, Twitter for Windows phone, Tweetcaster for Android, Echofon, Echofon Android, OS X, photos on Ios, Ios, camera on Ios, Twitter for iPhone, Twitter for Android, Twitter for iPad* and *Twitter for Blackberry*. Tweets whose value extracted from the API server is not included in these two types are not considered in this kind of analyses.

6.2.2.1. Level 1

6.2.2.1.1. Users' verified account

In the next table (Table 10) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the users' account verification within relevant cascades belonging to L1.

Verified account	Relevant tweets	Irrelevant tweets	Precision
False	67	9	11,84%
True	12	2	14,28%

Significance	1,206
Reliability	14

Table 10: Percentage of precisions of tweets posted by users without ("False" row) and with ("True" row) verified account, along with the significance and the reliability of the parameter, considering the set of tweets belonging to L1.

Results show that both precisions are very close to the average precision (12,22 %) and similar between them, fact that lead to obtaining a very low parameter significance. Hence, this parameter cannot be considered to classify tweets considering their relevance in this level.

6.2.2.1.2. Media

In the next table (Table 11) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the inclusion of media in tweets within relevant cascades belonging to L1.

Media	Relevant tweets	Irrelevant tweets	Precision
None	79	8	9,19%
Images	0	3	100%

Significance	10,875
Reliability	3

Table 11: Percentage of precisions of tweets non-containing ("None" row) and containing ("Images" row) images in them, along with the significance and the reliability of the parameter, considering the set of tweets belonging to L1.

Results show that the precision obtained when media (images) are enclosed in tweets is complete (100 %). This circumstance, added to the fact that the precision concerning tweets that don't include images is very low (9,1 %), causes that the resulting parameter significance to be very high. Otherwise, there are very few tweets containing images, and therefore results are not really reliable. Nevertheless, since images in the context of natural disasters is considered the main source of information, this parameter is found convenient to be used as classifier in this level.

6.2.2.1.3. GPS posting location

In the next table (Table 12) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the use of the GPS posting location in tweets within relevant cascades belonging to L1.

GPS posting location	Relevant tweets	Irrelevant tweets	Precision
False	77	10	11,49%
True	2	1	33,33%

Significance	2,900
Reliability	3

Table 12: Percentage of precisions of tweets posted non-providing ("False" row) and providing ("True" row) the GPS posting location, along with the significance and the reliability of the parameter, considering the set of tweets belonging to L1.

Results show that the precision in the case of making use of posting location in tweets is higher than the average precision, which is a positive fact. Nevertheless, considering that the sample size in this case is very low (3 tweets), along with a not very highly remarkable value regarding the parameter significance, leads to the inconvenience of using this parameter to classify tweets considering their relevance in this level.

6.2.2.1.4. Type of device

In the next table (Table 13) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the type of device used to post tweets within relevant cascades belonging to L1.

Type of device	**Relevant tweets**	**Irrelevant tweets**	**Precision**
Computer	42	5	10,63%
Mobile phone	29	5	14,70%

Significance	1,382
Reliability	34

Table 13: Percentage of precisions of tweets posted using computers ("Computer" row) and mobile phones ("Mobile phone" row), along with the significance and the reliability of the parameter, considering the set of tweets belonging to L1.

Results show that both precisions are very close to the average precision and similar between them, fact that lead to obtaining a parameter significance very close to 1. Hence, even if having a large sample size, this parameter cannot be considered to classify tweets according to their relevance in this level.

6.2.2.2. Level 2

6.2.2.2.1. Users' verified account

In the next table (Table 14) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the users' account verification within relevant cascades belonging to L2.

Verified account	Relevant tweets	Irrelevant tweets	Precision
False	33	4	10,81%
True	2	2	50,00%

Significance	4,625
Reliability	4

Table 14: Percentage of precisions of tweets posted by users without ("False" row) and with ("True" row) verified account, along with the significance and the reliability of the parameter, considering the set of tweets belonging to L2.

Results show that the precision in the case that tweets are posted by users possessing a verified account is much higher, accounting a remarkable 50%, than in the opposite case in this level. Nevertheless, the sample size obtained is too low (4 tweets) to consider the use of this parameter as convenient during the classification method design.

6.2.2.2.2. Media

In the next table (Table 15) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the inclusion of media in tweets within relevant cascades belonging to L2.

Media	Relevant tweets	Irrelevant tweets	Precision
None	35	5	12,50%
Images	0	1	100%

Significance	8,000
Reliability	1

Table 15: Percentage of precisions of tweets non-containing ("None" row) and containing ("Images" row) images in them, along with the significance and the reliability of the parameter, considering the set of tweets belonging to L2.

Results got in this level are very similar to these obtained in L1 (Section 6.2.2.1.2), and therefore their analysis is analogous to the one already provided in the previous level. Hence, the parameter is found convenient to be used as classifier in this level too.

6.2.2.2.3. GPS posting location

In the next table (Table 16) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the use of the GPS posting location in tweets within relevant cascades belonging to L2.

GPS posting location	**Relevant tweets**	**Irrelevant tweets**	**Precision**
False	32	4	11,11%
True	3	2	40,00%

Significance	3,600
Reliability	5

Table 16: Percentage of precisions of tweets posted non-providing ("False" row) and providing ("True" row) the GPS posting location, along with the significance and the reliability of the parameter, considering the set of tweets belonging to L2.

Once again, results obtained in this parameter don't significantly differ to these derived from L1 (Section 6.2.2.1.3), since precision is much higher than the average one (about 3 times). However, their sample size is too low for the precision to be reliable enough, and therefore to be able to include the parameter in the classification process.

6.2.2.2.4. Type of device

In the next table (Table 17) is presented the precision for both values, the significance of the parameter and the smallest sample size corresponding to a value regarding the type of device used to post tweets within relevant cascades belonging to L2.

Type of device	Relevant tweets	Irrelevant tweets	Precision
Computer	12	2	14,28%
Mobile phone	20	4	16,66%

Significance	1,166
Reliability	24

Table 17: Percentage of precisions of tweets posted using computers ("Computer" row) and mobile phones ("Mobile phone" row), along with the significance and the reliability of the parameter, considering the set of tweets belonging to L2.

Results highlight the fact that precisions in both cases are very similar and reach a low value, very close to the average precision of the level. Therefore, the parameter will not be considered in subsequent steps regarding the classifications of tweets according to their relevance.

6.3. Tweets' texts analysis

Previous analyses presented throughout this chapter have demonstrated that texts in tweets are considered relevant in a 95,59% (see Table 3) of the tweets considered relevant concerning the overall dataset. Hence, the convenience of analysing tweet's texts with the aim of classifying tweets according to their relevance is therefore clear.

This section deals with the identification of patterns in tweet's texts to classify them according to their relevance. Patterns can be used to select relevant tweets or to discard irrelevant ones. Each identified pattern has to be evaluated according to its effectiveness, meaning the ability to precisely separate tweets.

With the objective of identifying these patterns, a classification has been carried out considering all the replies excluding copied tweets (26 tweets), from L1 to L8, within relevant cascades. This fact means that 161 tweets have been qualitatively manually annotated according to the following types:

- Informative (description): tweet's texts which add information by making a description of a situation. As informative type it is priority, meaning that tweets are classified to this type even if they share characteristics of other types in the case that include descriptive information.

- Informative (others): tweet's texts which add information without making descriptions of any situation.

- Question descriptive information: tweet's texts that ask for descriptive information to other users.

- Question personal opinion: tweet's texts that ask for personal opinions to other users.

- Personal opinion: tweet's texts that provide a subjective and therefore personal opinion of a situation.

- Reaction: tweet's texts that include reactions to another parent tweet.

Tweets classified into the first two types are the ones whose text is considered relevant. On the contrary, irrelevant texts are classified to the last four types.

The following figure (Figure 73) presents the percentage of tweets classified to each one of these six types. Here, it is just considered tweets belonging to levels 1, 2 and 3 since they are the ones with a sample size bigger than 10 tweets per level and therefore conclusions can be reached. The results show that informative tweets, considering the two types, are balanced found in the three levels, that questions tend to be asked most of the times in L1 referring to the L0 tweet, and that personal opinions are usually provided in L2.

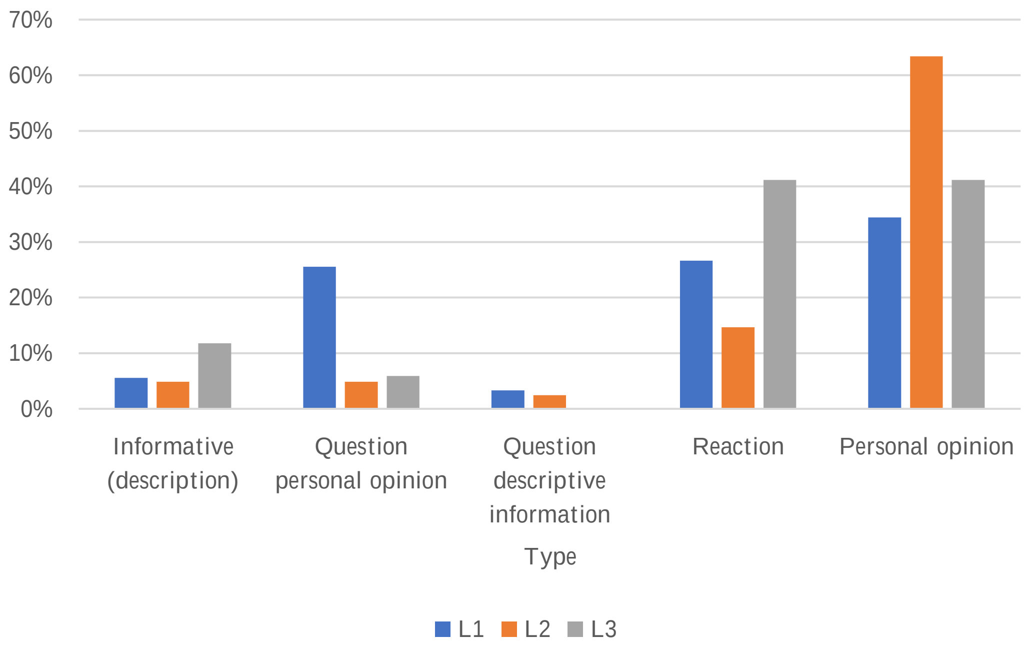


Figure 73: Percentage of tweets classified in each text category, segmenting by the levels 1-3.

6.3.1. Pattern recognition

With the aim of recognizing patterns considering the six types just presented, a qualitative analysis has been performed regarding the set of 161 tweets. Hence, every tweet from the set is analysed according to its type. Then, characteristics a priori shared between tweets of the same type, and not shared between types, are annotated as candidates to be patterns. In most of the cases, these characteristics make use of unigrams, that is to say specific characters or/and words to be found in texts.

After performing several computations, some of the annotated characteristics have been then demonstrated of not being convenient to be used due to two main reasons: they are shared more than expected between the different types or they don't appear in tweets as much as previously considered. An example regarding the first reason is the use of locations prepositions such as "in", "at" or "on".

Examples supporting the second reason are the use time prepositions and adverbs, and other expressions of time.

Finally, three characteristics are found convenient to be further evaluated[25] in order to become patterns. These characteristics are the unigram "?", location and a set of unigrams.

The unigram "?" aims to identify tweets corresponding to questions, since they are all irrelevant, to discard them. The following table (Table 18) presents the number of times that the unigram "?" is found on tweets for each type. As results show, just 1 tweet of the 31 tweets corresponding to questions doesn't use the unigram. Nevertheless, there is 1 relevant tweet that make use of the unigram, even if its text is not related to a question. Recall and precision values have been calculated taking into account the relevance of tweets. Therefore, as the unigram is used to discard questions, it is not considered for those tweets corresponding to reactions and opinions. Hence, the set encompass 50 tweets, of which 31 are discarded and 19 selected. On the one hand, from the 31 discarded tweets, there it is found 1 relevant tweet. On the other hand, from the 19 selected tweets, there is found 1 irrelevant tweet corresponding to questions. Since it is focused on the discarded ones, the recall and precision obtained from this characteristic is therefore of a 96,77% in both cases.

	Is "?" in the text?	
TYPE	NO	YES (to discard)
Informative (description)	10	0
Informative (other)	8	1
Question personal opinion	1	26
Question descriptive information	0	4
TOTAL	19	31

Table 18: Number of tweets classified in each text category differentiating whether they contain the unigram "?" in the text. The colour of cells indicates the relevance: green cells indicate relevant tweets; red cells indicate irrelevant tweets, and therefore the ones that must be discarded.

[25] Evaluation in this section only considers the relevance of tweets' texts (one of the three parts of a tweet), and then not the tweet one resulting from the combination of its three parts (text, media and URL).

The consideration of locations in texts aims to identify relevant tweets, separating them from those tweets corresponding to reactions and opinions. Locations considered can be provided either by referring an exact place, i.e. “Victoria St”, or by using relative terms, i.e. “in the town centre”. Regarding the first case, locations not provided with a minimum of precision (established in city names) or not referring to a place within the area of study (i.e. “Kiev”) are not considered. The second case has been included because relative terms are usually referred to an exact place mentioned in previous tweets within the same chain. The following table (Table 19) presents the number of times that locations are found on tweets for each type. It is seen that in none reaction and personal opinion tweets location is provided. Furthermore, that in a 12 of the 19 relevant tweets location has been provided. Therefore, as locations are used to select relevant tweets, the set encompass 130 tweets of which 12 are selected and 118 are discarded. On the one hand, from the 12 selected tweets all of them are relevant. On the other hand, from the 118 discarded tweets, there is found 7 being relevant. Since it is focused on the selected ones, the recall and precision obtained from this characteristic is therefore of a 63,15% and 100%, respectively.

	Is the location provided in the text?	
TYPE	NO	YES (to select)
Informative (description)	2	8
Informative (other)	5	4
Reaction	41	0
Personal opinion	70	0
TOTAL	118	12

Table 19: Number of tweets classified in each text category differentiating whether they provide the location in the text. The colour of cells indicates the relevance: green cells indicate relevant tweets, and therefore the ones that must be selected; red cells indicate irrelevant tweets.

The set of unigrams aims to identify tweets corresponding to reactions and personal opinions to discard them. The set is composed of the 5 following unigrams: “your”, “thank”, “!”, “(“, “)”. Hence, the characteristic works using an OR function: if any of the unigrams is found in a text, the outcome is positive, and negative otherwise. Nevertheless, in this analysis only tweets that don’t provide locations and belonging to levels 1, 2 and 3 are considered, resulting in 97 tweets. The following table (Table 20) presents the number of times that the set of unigrams is found on tweets for each type. As results show, no relevant tweets are found carrying any of the unigrams, while approximately half of reaction and personal opinion tweets carry them. Therefore the set encompass 96 tweets of which 41 are discarded and 56 selected. On the one hand, from the 41 discarded tweets none of them is relevant. On the other hand, from the 55 selected tweets, 4 are relevant while the resting 51 are irrelevant. Since it is focused on selecting and discarding, the recall and precision obtained from this characteristic is therefore of a 100% and 7,27% respectively for the selecting process, and of 44,56% and 100% respectively for the discarding process.

	Is any of the unigrams of the established set in the text?	
TYPE	NO (to select)	YES (to discard)
Informative (description)	1	0
Informative (other)	3	0
Reaction	19	14
Personal opinion	32	27
TOTAL	55	41

Table 20: Number of tweets classified in each text category differentiating whether they contain any of the unigrams included in the established set in the text. The colour of cells indicates the relevance: green cells indicate relevant tweets, and therefore the ones that must be selected; red cells indicate irrelevant tweets, and therefore the ones that must be discarded.

Finally, the table below (Table 21) summarises the effectiveness of these three characteristics, highlighting their recall, precision and F-value values.

	Unigram "?"	Location	Set of unigrams (to select)	Set of unigrams (to discard)
Recall	96,77 %	63,15 %	100 %	44,56 %
Precision	96,77 %	100 %	7,27 %	100 %
F-value	96,77 %	77,41 %	13,56 %	61,65 %

Table 21: Recall, precision, and F-value (performance) values for each of the identified tweet's text patterns.

Chapter 7

Classification method

This brief chapter deals with the presentation of the classification method of reply tweets, which aims to select those ones considered relevant for enhancing situational awareness of humanitarian organizations in natural disaster situations and discard the irrelevant ones, originated from a provided set of tweets previously extracted through queries to the Twitter API server. A theoretical performance evaluation of the proposed method is also subsequently presented.

The patterns identified, in other words, those characteristics or values regarding the structure of cascades, tweets' parameters and tweets' texts resulting useful for sorting out reply tweets according to their relevance throughout the previous chapter (Chapter 6), are employed for the design of the classification method.

In the following sections, first all the identified patterns in Chapter 6, along with the features deemed useless for classifying reply tweets, are presented in Section 7.1. Finally, in Section 7.2 the proposed classification method for reply tweets according to their relevance is introduced.

7.1. Identified patterns

After performing all kinds of analyses in the previous chapter regarding cascades' structure characteristics, tweet's parameters considering cascades' structure, and finally tweet's texts, according to the tweets' annotated relevance aiming to identify patterns to be used as classifiers, this section deals with the explanation and justification of the characteristics selected as patterns.

Analyses regarding cascades' structure characteristics proved the following points:

- It is more effective to consider for the retrieval of relevant replies just cascades deemed to be relevant, that is to say tweets in L0 annotated as relevant, since they guarantee a better precision of the selected set of tweets while conserving the same recall level.

- It is found convenient to only consider tweets belonging to levels from 1 to 3, since the recall obtained by considering just them is of an 86,36%.

- It is found convenient to use tweets' texts to classify them according to their relevance, since texts are considered relevant in an 95,59% of the overall relevant tweets in the dataset, and in an 89,47% of the set of relevant tweets belonging to levels from 1 to 3.

- Tweets' relevance only due to media (images) just account for a 3,96% considering the overall relevant tweets in the dataset. Nevertheless, given the nature of the entire case study (beyond my thesis), which is focused on finding relevant images, it is found convenient to retrieve all the tweets including images in order to increase the recall level of the classifier, even if precision levels would be negatively affected.

- It has been demonstrated that when the users that post any tweet in a cascade are the same who posted the first tweet in L0 (*original users*) entails a higher probability, around 8 times higher in L1 and 3,5 in L2, for tweets to be relevant. The recall and precision obtained by considering this *original user* condition is of an 42,10% and 28,57%, respectively.

- Predictive analyses have been highlighted of not being convenient to be used as patterns.

- The number of replies in tweets have been highlighted of not being convenient to be considered as patterns.

Analyses regarding tweet's parameters proved the following points:

- All the parameters whose values are of an infinite integer type (retweets' number and likes' number of tweets; followers' number, number of users followed and account's posts number by the users posting tweets) are found not convenient to be used as patterns in none of the two levels studied (L1 and L2).

- Users' verified accounts, the use of the GPS posting location, and the type of device used to post parameters, whose values are of a binary type, are also not found convenient to be used as patterns in none of the two levels studied (L1 and L2) based on the results obtained.

- The only parameter that has demonstrated to be useful to be used as pattern is inclusion of media in tweets, since it entails a 100% precision in both levels.

Analyses regarding tweets' texts found convenient to use as patterns the following characteristics:

- The use of the unigram "?" by users in the text of their tweets.

- The mention of locations by users in the text of their tweets.

- The use of at least one of the unigrams contained in the established set ("your", "thank", "(", ")" and "!") by users in the text of their tweets.

The following table (Table 22) summarises the characteristics and parameters to be used in the classification process of tweets considering each studied feature, highlighting the section within Chapter 6 on which were analysed.

	To be used	Not to be used
Cascade structure characteristics		
Relevant cascades	X	
Tweets belonging to L1, L2 and L3	X	
Tweets' texts usefulness to classify	X	
Original user condition	X	
Number of replies		X
Predictive analyses		X
Tweets' parameters values in cascades		
Number of retweets		X
Number of likes		X
Number of followers		X
Number of followed users		X
Number of posts		X
User verified account		X
Tweets containing images	X	
Tweets providing GPS posting location		X
Posting device type		X
Text analysis		
Unigram "?"	X	
Location	X	
Set of unigrams	X	

Table 22: Characteristics (of cascades and tweets' texts) and tweets' parameters to be used in the classification process.

7.2. Classification process

A classification method considering the identified patterns found in the previous chapter and presented in the previous table (Table 22) is described and subsequently evaluated in this section.

The proposed process is divided into three sequential steps:

1. Filtering: Firstly, tweets will be filtered according to the relevance of the L0 tweet in their respective cascades, the level on which they belong to, and the treatment of tweets whose text is copied from other tweets within a same cascade. The first filter (optional) consists of just considering tweets within cascades whose L0 tweet is relevant, which means that a previous manual annotation of the tweets registered in the starting dataset is convenient to be performed[26]. The second filter consists of just automatically retrieving tweets belonging to levels 1,2 and 3 in cascades for subsequent analyses. The third filter consists of automatically discarding copied tweets.

2. Text analysis: Secondly, filtered tweets are analysed according to their texts. Hence, a sequential filtering process taking into account the condition of *original user* introduced in Section 5.1, as well as the three patterns (unigram "?", location and set of unigrams) found in the text analysis section (Section 6.3) is used. The process develop as follows:

    i. The unigram "?" will be firstly used to discard tweets a priori corresponding to questions, which obviously are the ones that include the unigram.

    ii. The selected tweets will be separated according to if they provide a location or not: the ones providing location will directly be included in the final dataset, while the others will be sent to perform next steps.

    iii. Tweets sent to next steps will be analysed in terms of the posting user according to the *original user* condition. Therefore, tweets

[26] See discussion in Section 7.2.1.

posted by *original users* will continue to be analysed, while the others will be discarded.

iv. Tweets continuing to be analysed will be separated according to if they include any of the unigrams in the established set (and presented in Section 6.3), or not. Therefore, the ones including them will be discarded, while the others will be included in the final dataset.

3. Images selection: Finally, all the tweets that overpass the filtering step (first), include images, and have not been already included in the dataset resulting from the text analysis step (second), will be included the final dataset.

Figure 74 provides a graphical representation of the proposed classification process of tweets is. Furthermore, the flow in terms of the quantity of tweets for each possible path taken is portrayed, segmenting by their relevance.

In the table below (Table 23) is presented the number of tweets studied and the quantity obtained in each of the sets, selected (whose objective is to retrieve all the relevant tweets) and discarded (whose objective is to retrieve all the irrelevant tweets), and segmented by their relevance. This computation will allow to calculate the performance of the classification method in the subsequent table. The set of tweets studied is the one resulting after the filtering step, since it just aims the discard of those tweets not interested to be considered in the analyses.

	Total tweets	Relevant	Irrelevant
Set of tweets studied (after filtering step)	148	19	129
Selected set	23	15	8
Discarded set	125	4	121

Table 23: Number of tweets (total, relevant and irrelevant) in each of the studied sets.

Finally, the following table (Table 24) presents the performance of the classification method, highlighting the obtained recall, precision and F-value values for each of the sets, and the accuracy obtained considering both.

	Selected set	Discarded set
Recall	78,94 %	93,79 %
Precision	65,21 %	96,80 %
F-value	71,42 %	95,27 %
Accuracy	91,89 %	

Table 24: Recall, precision, and F-value (performance) values of both selected and discarded sets, along with the accuracy value of the classification method.

Even if reported the results for both sets, the objective of the classification method focuses on the selected one, which reaches a performance higher than 70%. Furthermore, the resulting accuracy is very high (91,89 %), which confirms the effectivity of the proposed classification method.

Besides, it is noticed that the introduction of the image selection step after the text analysis process has helped to increase the performance of the classification method. Nevertheless, this change could have been negative in the case that the images selected in that step resulted to be irrelevant since precision would have decreased (recall values could have either been maintained or increased). As it has been already mentioned that in terms of images it is preferred to attempt to increase recall even if precision can be negatively affected because of rapid mapping purposes, the inclusion of the images selection step is conveniently justified.

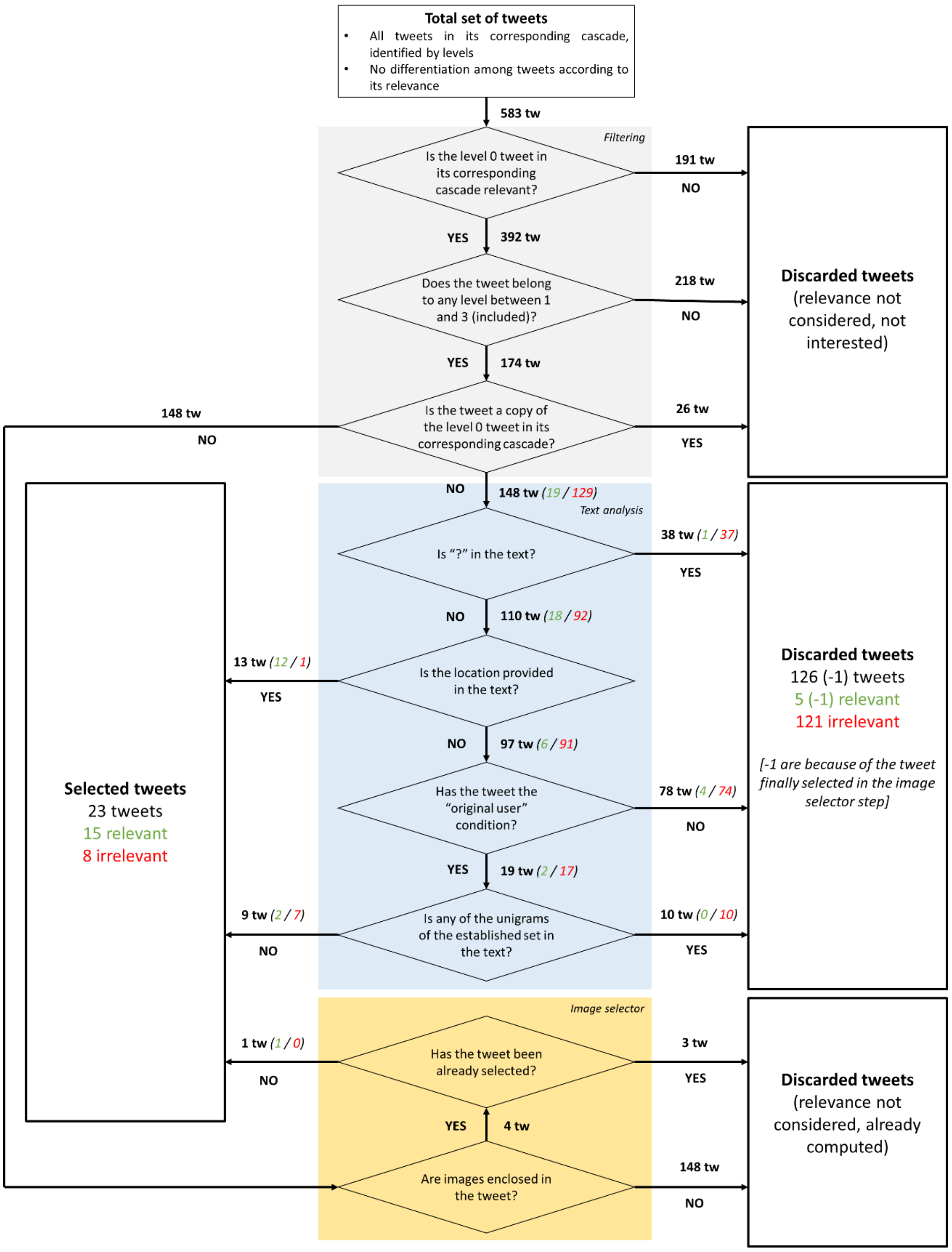

Total set of tweets
All tweets in its corresponding cascade, identified by levels
No differentiation among tweets according to its relevance
583 tw
Filtering
Is the level 0 tweet in its corresponding cascade relevant?
191 tw
NO
YES
392 tw
Does the tweet belong to any level between 1 and 3 (included)?
218 tw
NO
YES
174 tw
Is the tweet a copy of the level 0 tweet in its corresponding cascade?
148 tw
NO
26 tw
YES
Discarded tweets (relevance not considered, not interested)
NO
148 tw (19 / 129)
Text analysis
Is "?" in the text?
38 tw (1 / 37)
YES
NO
110 tw (18 / 92)
Is the location provided in the text?
13 tw (12 / 1)
YES
NO
97 tw (6 / 91)
Has the tweet the "original user" condition?
78 tw (4 / 74)
NO
YES
19 tw (2 / 17)
Is any of the unigrams of the established set in the text?
9 tw (2 / 7)
NO
10 tw (0 / 10)
YES
Discarded tweets
126 (-1) tweets
5 (-1) relevant
121 irrelevant
[-1 are because of the tweet finally selected in the image selector step]
Selected tweets
23 tweets
15 relevant
8 irrelevant
Image selector
Has the tweet been already selected?
1 tw (1 / 0)
NO
3 tw
YES
YES
4 tw
Are images enclosed in the tweet?
148 tw
NO
Discarded tweets (relevance not considered, already computed)

Figure 74: Representation of the process that every tweet undergoes. Numbers represent the number of total/relevant/irrelevant tweets that undergo each step (black colour for total tweets / green colour for relevant tweets / red colour for irrelevant tweets).

7.2.1. A discussion about the first filter in the filtering step

The first filter in the filtering step of the proposed classification method aims to discard cascades owning a L0 tweet considered useless to enhance situation awareness (thus irrelevant), since in the analysis corresponding to Figure 56 has been demonstrated that no relevant reply tweet has its origin from irrelevant L0 tweets. Hence, patterns throughout the analyses have been identified considering only those cascades originated from relevant L0 tweets (those helping to enhance situational awareness), and therefore considered as relevant.

Classification patterns found and finally implemented in the method focus on selecting those reply tweets with informative purposes: the unigram "?" is used to discard a priori tweets corresponding to questions; locations provided in texts are used to select a priori tweets corresponding to descriptions; the *original user* condition is used to discard those tweets not posted by these users because of the low relevance probability that they entail; and finally the set of unigrams is used to discard tweets a priori corresponding to reactions and personal opinions and select the informative ones.

Consequently, it can be understood (from the above explanation) that reply tweets originated from irrelevant L0 tweets, and therefore irrelevant, could perfectly satisfy all these steps and be included in the resulting selected set. This fact is feasible since the patterns used just take into account the format of tweets' text (*original user* step apart) aiming to only extract informative ones, and reply tweets originated from irrelevant L0 tweets can also be of an informative type although not contributing to enhance situational awareness (thus irrelevant). An example of an irrelevant reply tweet that would have been selected if the L0 of its corresponding cascade had not been discarded in the first filter of the filtering step is provided in Figure 75.

Hence, the skip of this optional first filter causes a decrease on the precision of the selected set, since informative reply tweets not focused on enhancing situational awareness (thus irrelevant) would be included, denoting its convenience to be used.

Furthermore, another point to be considered is that from a L0 tweet are originated on average 4,03 [27]reply tweets (in the case that the L0 tweet is replied). This fact accentuates the convenience to introduce to the classification process only

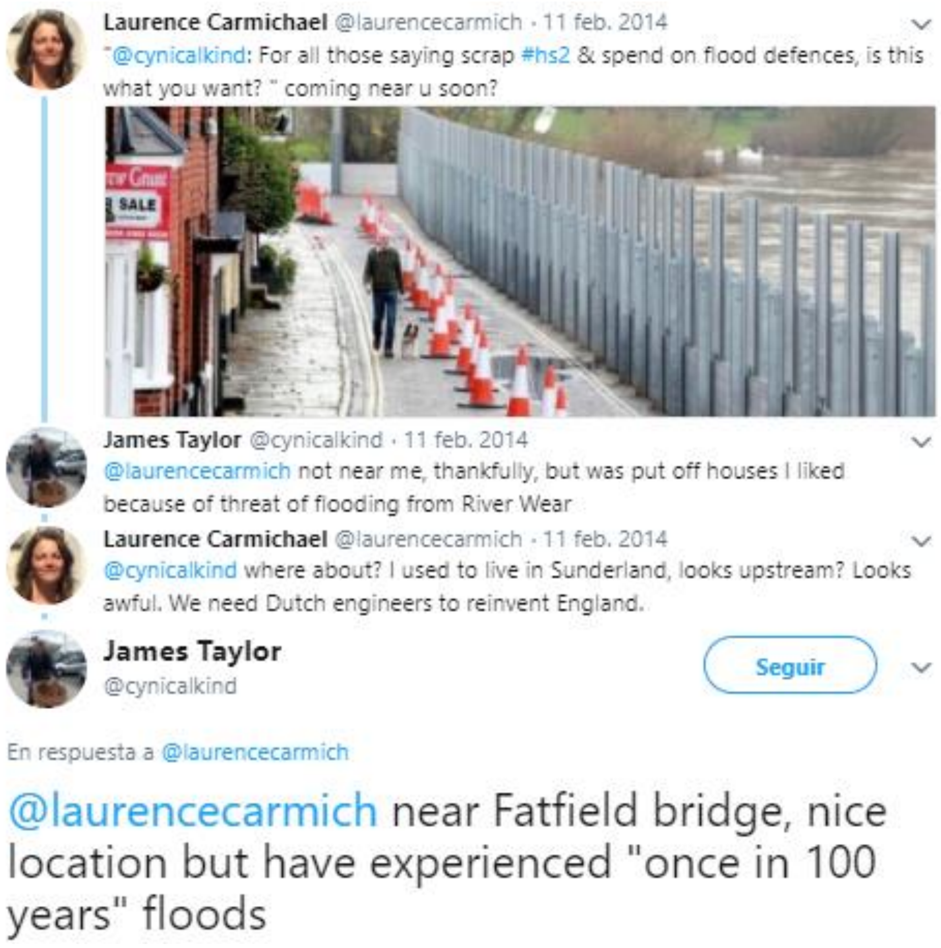
[27] Statistic that concerns L0 tweets related to the case study. Nevertheless, it was seen that L0 tweets dealing with other topics (sports or TV shows) behave differently.

tweets able to enhance situational awareness (thus relevant), since for every irrelevant tweet introduced, the classifier must process four additional irrelevant reply tweets to finally discard them.

Nonetheless, there are two cases in which the skip of the filter is recommended:

- Dataset containing only relevant L0 tweets: In this idealistic case, the execution of the filter would be useless. However, currently no algorithm is able to extract L0 tweets with a 100% precision.

- Large volume datasets: In this common case, it is unfeasible to perform manually the filter. Nevertheless, the work should be focused on extracting L0 tweets with high precision levels in order to improve the performance of the classification method.


Laurence Carmichael @laurencecarmich · 11 feb. 2014
"@cynicalkind: For all those saying scrap #hs2 & spend on flood defences, is this what you want? " coming near u soon?

James Taylor @cynicalkind · 11 feb. 2014
@laurencecarmich not near me, thankfully, but was put off houses I liked because of threat of flooding from River Wear

Laurence Carmichael @laurencecarmich · 11 feb. 2014
@cynicalkind where about? I used to live in Sunderland, looks upstream? Looks awful. We need Dutch engineers to reinvent England.

James Taylor
@cynicalkind

En respuesta a @laurencecarmich

@laurencecarmich near Fatfield bridge, nice location but have experienced "once in 100 years" floods

Figure 75: L3 reply tweet (bottom).

Accordingly, it has been demonstrated that the precision obtained in the selected set (output dataset) is the resulting from the precision of two main processes:

- Extraction of L0 tweets: The extraction of tweets through queries to the Twitter API server. Algorithms employed focus on the retrieval of tweets including media and providing locations in the text. The generated dataset is the one employed as input for the next process. This process is out of the scope of this thesis.

- Classification of reply tweets: This is the process in which the thesis is based on. As already explained, it deals with the extraction of a priori relevant reply tweets from a provided input dataset (generated in the previous process). The reply tweets selected are included in the output dataset.

Next figure (Figure 76) represents the flow of the two processes (or phases), highlighting their corresponding precisions.

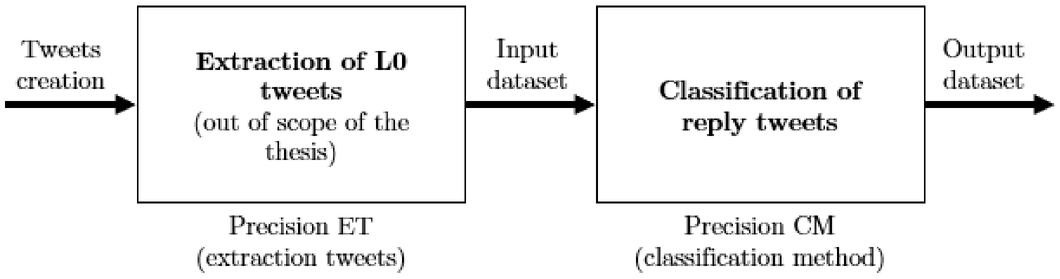

Tweets creation
Extraction of L0 tweets
(out of scope of the thesis)
Input dataset
Classification of reply tweets
Output dataset
Precision ET (extraction tweets)
Precision CM (classification method)

Figure 76: Representation of the overall process.

Consequently, the precision obtained in the output dataset (overall precision) could be calculated as:

$$Overall\ precision = Precision\ ET \cdot Precision\ CM$$

However, since the relation is not one-to-one, meaning that for each L0 tweet multiple (or zero) reply tweets are generated, the overall precision must be calculated as:

$$Overall\ precision = Factor\ ET \cdot Precision\ CM$$

$$Factor\ ET = f(Precision\ ET) \qquad Factor: [0,1]$$

The Factor ET is a value function of the precision of the first process, namely the percentage of relevant L0 tweets in the input dataset of the second process. Obviously, the higher the precision is the higher the Factor ET is, although there is not linear correspondence since it depends on the number of reply tweets originated from L0 tweets, value that varies according to the topic that every L0 tweet deals with.

Therefore, it is evident that in order to only evaluate the classification method devised in the thesis (second process) is necessary to provide to the process an input dataset with a 100% of precision (Factor ET equal to 1). This is done by using the first filter in the filtering step in both the theoretical (Section 7.2) and practical, namely the tool, (Section 8.3.1.1) evaluations.

Notwithstanding, in Section 8.3.1.2.1 and 8.3.1.2.2 the input dataset considered will contain irrelevant tweets with the aim of assessing in which grade the overall precision is affected by the first process. The resulting Factor ET value will also be calculated and interpreted.

Chapter 8

ReTREC tool for reply's tweets classification

This chapter deals with the presentation of the created ReTREC (Relevant Tweets' Replies Extraction in Cascades) tool.

The objective of the tool is the extraction of relevant reply tweets originated from a provided set of tweets. Consequently, the extracted replies from ReTREC would help to enhance the level of situational awareness obtained from that original set of tweets. The relevance criteria followed for the classification of tweets is the same used during the annotation process described in Section 4.3 and included in the Appendix (Section i).

The tool consists of two main processes: data collection process and classification process. On the one hand, the data collection process embraces all those steps necessary for the construction of the tweets' cascades. On the other hand, the classification process considers the execution of the procedure proposed in Section 7.2. Both processes are detailly explained in Section 8.1.

The tool provides an interface to be run in Linux with the aim of removing utilisation barriers due to its complexity, and therefore allow unexperienced people in terms of computer science to use it. This interface automatically executes all the required functions, programmed using Python language. A description of the interface is also provided in this chapter.

Finally, an evaluation of the classification performance and accuracy obtained by the tool taking into account two different case studies is included.

In the following sections, first it is described the main processes executed by the tool to classify the reply tweets according to their relevance, illustrating the different Python libraries used, in Section 8.1. In Section 8.2 some instructions are provided to implement the tool to any system, along with a description of the its interface. Finally, in Section 8.3 an evaluation of the tool through the two case studies introduced in Chapter 3 is performed.

8.1. Process description

8.1.1. Data collection process

The tool uses an Excel file containing tweets' ids as input source. These ids must be registered in a specific column, as later indicated in Section 8.2.

The column containing the tweets' ids is retrieved by the *ex_parents* function within the *parent.py* file[28] with the objective of finding the parent tweet (or L0 tweet) of the cascades corresponding to these original tweets. The process that is carried out is very similar to that explained in Chapter 4: for each provided tweet's id, its webpage's source (XML of the page) is parsed making use of the BeautifulSoup library (*prova_par* function within *parent.py* file). The parsing aims to obtain the id of the L0 of the cascade thanks to the attribute 'data-conversation-id' inside the class 'div'. In the case that throughout that process an obtained tweet either is not available (has been eliminated) or belongs to a user with private account, the cascade is no longer considered. Once this process is completed for all the tweets provided in that original set, elimination of duplicates is performed. The fact of having L0 tweets duplicated in the final dataset means that two or more tweets in the original set belong to the same cascade.

The next step deals with the manual annotation of the L0 tweets obtained, replicating the first filter within the filtering procedure described in the classification process of the previous chapter (Chapter 7). The objective of the step is to keep those tweets considered relevant for subsequent phases and discard those irrelevant ones, since it was previously found in Chapter 6 that relevant replies just originate from relevant L0 tweets. The rows containing ids whose tweets were annotated as irrelevant are deleted from the database.

[28] All the Python files and functions used in Section 8.1 are enclosed in the Appendix (Section iii.b).

The following stage is focused on the construction of tweets' cascades until L3. Two main steps encompass this stage: replies' ids search and tweets' level classification.

- **Replies' ids search:** For each L0 tweet, its webpage's is opened using the Selenium library and scrolled down until the end to make all the possible replies to appear in the window. Then, the webpage source is parsed obtaining a set of ids corresponding to replies thanks to the attributes 'data-tweet-id' and 'data-has-parent-tweet' within the 'div' class. Nevertheless, the parsing just guarantees the obtaining of replies' ids of the immediate subsequent level, and therefore the mentioned step must be performed for every id obtained in previous iterations (just one time for id). This fact means that the number of parsings to be done is equal to the total number of tweets within the cascade. Functions *ex_cascades, id_parsing* and *prova_nova* within *cascades.py* file are used for executing this step.

- **Tweet's level classification:** Once the ids corresponding to all the replies within a cascade are obtained, they must be clustered into levels. To do so, the API Twitter server is called for each id to get its status. Consequently, the parameters 'in_reply_to_status_id_str', 'text', 'entities' and 'user' are extracted from the JSON file provided. Even if just the first one is used in this specific step, the others are also extracted to minimise the number of calls to the API server consequence of the limitation of calls imposed by Twitter. Therefore, the immediate parent of each tweet is obtained through the 'in_reply_to_status_id_str' parameter. Then, an iteration is done for all the tweets in the cascade to find the one whose parent is *None*, corresponding to the L0 tweet. Three successive iterations are done (one per each level to be considered, levels 1,2 and 3) to find and cluster those tweets whose parent has been clustered in the immediate previous level. Tweets still not clustered after these iterations not belong to levels 0-3, and therefore are discarded. The fact of discarding these tweets replicate the second filter within the filtering procedure in the classification process of the previous chapter (Chapter 7). Functions *ordenar* and *get_tw* within *cascades.py* file are used for executing this step.

8.1.2. Classification process

The objective of this step is to classify the reply tweets still present in the database according to their relevance and select those ones considered relevant. The process is composed of a function employing several simple conditionals that replicate the stages proposed in the classification process of the previous chapter (Chapter 7). Furthermore, it is used the algorithm provided by the API Rosette server which is based on entity linking [29]techniques to identify locations in tweets' texts. Nevertheless, three considerations regarding the classification process executed by the tool have to be made:

- Copied tweets are difficult to identify automatically, since there are times in which they adopt unusual text structures, fact that could cause a decrease of precision.

- The algorithm provided by Rosette is not able to identify relative locations. Therefore. relative locations in texts such as "town centre" are not considered by the tool, fact that could cause a decrease of recall.

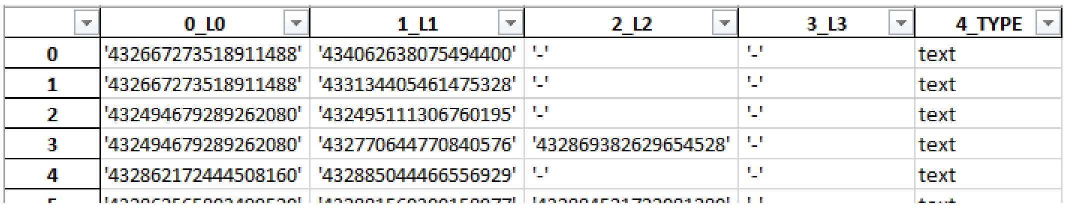
- The image selection step is substituted by a media selection step, meaning that replies containing all type of media (images, videos or GIFs) are always extracted.

The output of the entire process is an Excel file containing the tweets' ids of every chain (in rows) containing a relevant tweet in the four first columns (one for each level). The id from the relevant tweet of the chain will always be the last one provided (in each row). For instance (corresponding to row 3 in Figure 77), if a relevant tweet is found in L2 of a cascade, in the first column the id corresponds to the L0 of the cascade, in the second column the id represents the L1 tweet of the corresponding chain, in the third column it is found the id of the relevant tweet, and finally in the fourth column no id is provided ('-' instead).

Furthermore, the fifth column 'TYPE' registers the part of the tweet responsible of its selection. Consequently, three different types could arise: text, text and photos, and photos (the API server does not differentiate between media types; thus, videos and GIFs are presented as photos).

The function *classifier* within *cascades.py* file is used for executing this step.

[29] Concept introduced in Section 2.2.



	0_L0	1_L1	2_L2	3_L3	4_TYPE
0	'432667273518911488'	'434062638075494400'	'-'	'-'	text
1	'432667273518911488'	'433134405461475328'	'-'	'-'	text
2	'432494679289262080'	'432495111306760195'	'-'	'-'	text
3	'432494679289262080'	'432770644770840576'	'432869382629654528'	'-'	text
4	'432862172444508160'	'432885044466556929'	'-'	'-'	text

Figure 77: Example of the generated Excel file by the tool, containing the ids of the relevant reply tweets.

8.2. Interface description and guide

In order to be able to launch the ReTREC tool, firstly all the necessary libraries, modules and bindings must be installed in the system (Python3). They are gathered in the following list:

- BeautifulSoup: It is a Python library for pulling data out of HTML and XML files, and provides idiomatic ways of navigating, searching, and modifying the parse tree.

- Selenium: They are Python bindings that provide a convenient API to access Selenium WebDrivers such as Firefox, Ie and Chrome.

- Pandas: It is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

- Urllib: It is a module that provides a high-level interface for fetching data across the World Wide Web.

- Tweepy: It is open-sourced, hosted on GitHub and enables Python to communicate with Twitter platform and use its API.

- PyQT5: It is one of the two most popular Python bindings for the Qt cross-platform GUI/XML/SQL C++ framework. It allows the creation of interfaces to be run in Linux.

- Rosette API: It is a binding that allows to interact with the Rosette API server and therefore to take advantage of all the features offered such entity extraction, sentiment analysis, topic extraction and so on.

Furthermore, the folder of the tool, containing the files *tool.py*, *parent.py* and *cascades.py,* must be copied inside the hard disk of the computer.
To launch the ReTREC tool, the following steps must be carried out:

- Open the Terminal (Ctrl+T).

- Stand inside the tool folder: use the command “ls” to visualize documents and folders inside the current folder, “cd” to go the parent folder, and “cd NAME_FOLDER” to go to a folder with a name NAME_FOLDER which is inside the current folder.

- Execute the command “python3 tool.py”.

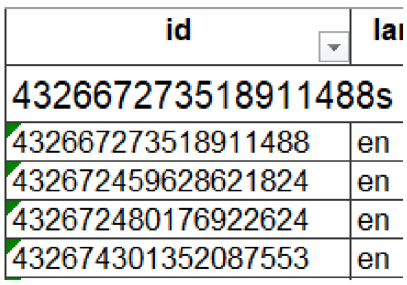
The first phase of the interface (Figure 79) contains a set of instructions to be followed before starting the process. These instructions deal with the manner in which the ids provided as input must be presented, and are:

- The name of the Excel file containing the ids must be ‘a.xls’.

- The name of the sheet within the file containing the ids must be ‘TWEETS’.

- The name of the column in which the ids are registered must be ‘id’.

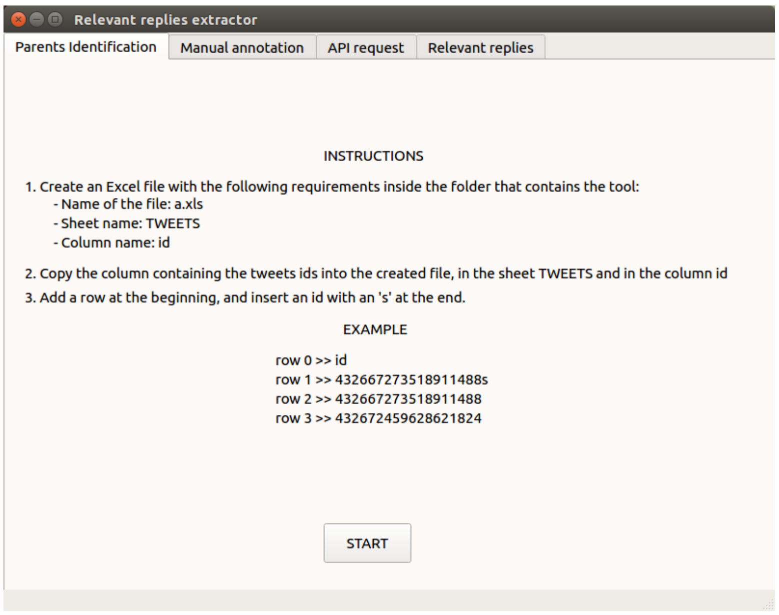
- The first row of the column (without considering the header row) must not contain an id number, but a string.



id	la
432667273518911488s	
432667273518911488	en
432672459628621824	en
432672480176922624	en
432674301352087553	en

Figure 78: Example of the column containing the ids of the tweets that are used as data input in an Excel file.

Once all the instructions are completed, the 'START' button can be clicked, and the process corresponding to the finding of parents' ids will begin.



INSTRUCTIONS

1. Create an Excel file with the following requirements inside the folder that contains the tool:
   - Name of the file: a.xls
   - Sheet name: TWEETS
   - Column name: id

2. Copy the column containing the tweets ids into the created file, in the sheet TWEETS and in the column id

3. Add a row at the beginning, and insert an id with an 's' at the end.

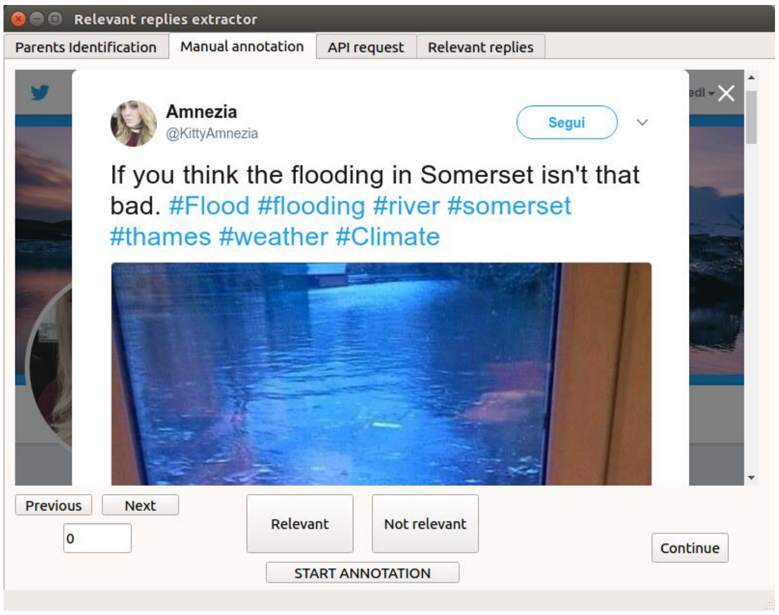
EXAMPLE

```
row 0 >> id
row 1 >> 432667273518911488s
row 2 >> 432667273518911488
row 3 >> 432672459628621824
```

Figure 79: Screenshot of the first phase of the interface (1/4).

The second phase of the interface (Figure 80) deals with the manual annotation of tweets. With the aim of minimising the time required to complete this task, for each tweet in the dataset its corresponding webpage 'http://twitter.com/statuses/id' is generated automatically and directly in a window within the interface. After considering a tweet, the user must consider if either it is relevant or not and make the decision by making use of the buttons "Relevant" and "Irrelevant". Those tweets annotated as irrelevant will be immediately removed from the dataset and not recoverable without relaunching the tool. Besides, the user can move into the dataset by making use of the buttons "Previous" and "Next", without the necessity of annotating tweets. Once the annotation process is finished, the user can click the "Continue" button to pass to the next step.

The third phase (Figure 81) only consist of providing the tool with the Twitter and Rosette API server credentials needed to make calls to the service. Hence, the user must have a Twitter Developer and Rosette accounts to be able to continue with the process. Once the required passwords (Consumer Key, Consumer Secret, Access Token, Access Token Secret in the case of Twitter, and User Key in the case of Rosette) are inserted, the user can click the “Continue button”. It is recommended to let the interface in “Always on Top” option since the next step deals with the construction of the tweets’ cascade and Google Chrome pop-up windows emerge constantly.



If you think the flooding in Somerset isn't that bad. #Flood #flooding #river #somerset #thames #weather #Climate

Figure 80: Screenshot of the second phase of the interface (2/4).

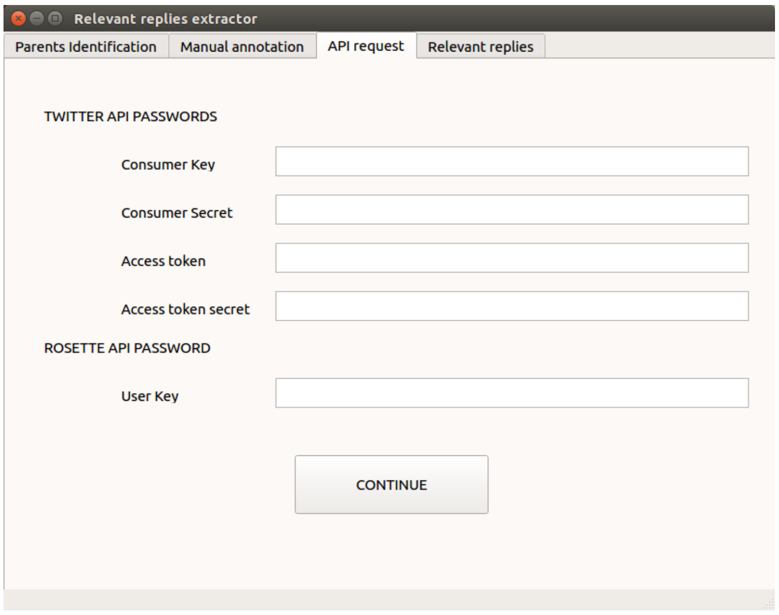
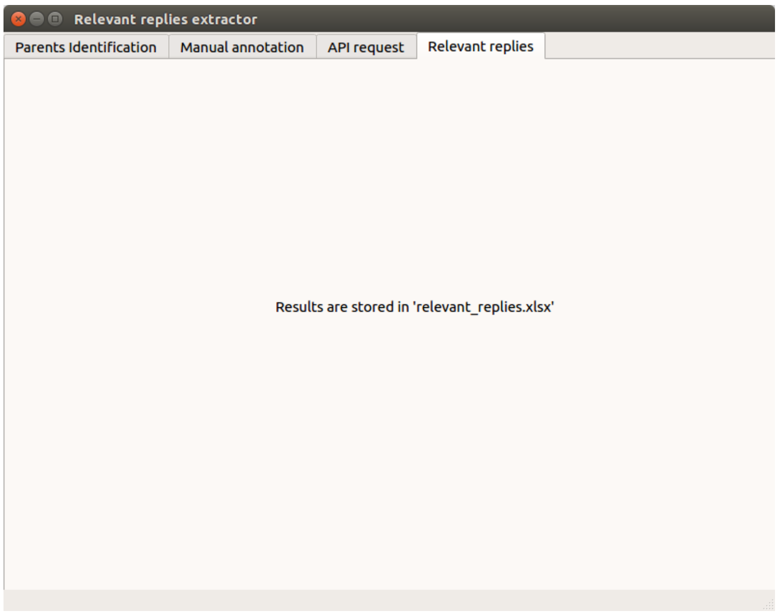


Figure 81: Screenshot of the third phase of the interface (3/4).

Finally, the fourth phase (Figure 82) just mention the name of the Excel file containing the selected set of tweets "Relevant tweets.xlsx", and therefore to be consulted. At this moment the user can quit the interface.



Results are stored in 'relevant_replies.xlsx'

Figure 82: Screenshot of the fourth phase of the interface (4/4).

8.3. Evaluation

The mission of the evaluation taking place in this section is threefold:

- To evaluate the yield of the tool, that is to say, to verify if the performance results theoretically obtained in Section 7.2 are also reached in practice when using the tool, and to analyse the possible causes of mismatches between both. This evaluation is performed in Section 8.3.1.1.

- To evaluate the effect derived from the input dataset in the overall precision achieved. This evaluation is performed in Section 8.3.1.2.1 and 8.3.1.2.2.

- To evaluate the proposed classification method, in other words, to assess if the classification process devised through the Southern England 2014 floods case study is also efficient when applied to datasets originated from other case studies, in particular the Hurricane Harvey 2017 storms in Texas case study. This evaluation is performed in Section 8.3.2.

As just mentioned, two datasets corresponding to two different case studies are employed for the evaluation. The case studies are the Southern England 2014 floods and Hurricane Harvey 2017 storms in Texas.

8.3.1. Southern England 2014 floods dataset

The dataset used in this section for the evaluation is the same that has been exploited throughout the thesis. For this reason, the results that will be obtained during the evaluation will be able to be compared with the theoretical ones previously calculated.

Furthermore, since all the tweets in the dataset have been previously manually annotated, performance corresponding to the discarded set and classification accuracy can be computed too.

Section 8.3.1.1 evaluates the results achieved in the case the first filter of the filtering step is used, while in Section 8.3.1.2 the evaluation is performed in the case the filter is not used.

Moreover, this latter section is divided into two subsections which employ different input datasets: the input dataset used in Section 8.3.1.2.1 only contains L0 tweets related to the case study (relevant and irrelevant), while in Section 8.3.1.2.2 the input dataset also contains L0 tweets concerning other topics, namely out-of-topic.

8.3.1.1. First filter considered

The purpose of this section is to evaluate the performance of the proposed ReTREC tool, comparing the results obtained through it with the theoretical ones presented in Section 7.2. The possible mismatches between both results are identified, and the causes provoking them analysed.

In this section, the first filter of the process, which aims to consider just cascades originated from relevant L0 tweets, has been used to be able to properly compare theoretical with practical results[30], meaning that the precision of the input dataset is of 100% (see discussion of Section 7.2.1). Therefore, the precision obtained in the selected set (output dataset) is the precision of the classification method when using the tool.

In the table below (Table 25) is presented the number of tweets studied and the quantity obtained in each of the sets, selected and discarded, and segmented by their relevance, as done in Table 23. Also, between parenthesis is highlighted the gap with regards to the theoretical results.

	Total tweets	Relevant	Irrelevant
Reply tweets collected (from L1 to L3)	148	19	129
Selected set	27 (+ 4)	12 (- 3)	15 (+ 7)
Discarded set	121 (- 4)	7 (+ 3)	114 (- 7)

Table 25: Number of tweets (total, relevant and irrelevant) in each of the studied sets obtained using the tool. Between parenthesis is highlighted the gap with the theoretical results.

It can be observed that all the obtained gaps are highlighted in red, pointing out its negativity. On the one hand, the selected set have obtained less relevant tweets and more irrelevant employing the tool than theoretically expected. On the other hand, the discarded set encompass more relevant tweets and less irrelevant. Hence, it can be affirmed that in practise performance will be lower than that one theoretically calculated in Section 7.2. Nevertheless, before computing the obtained performance value it is aimed to understand the causes which originate these gaps.

To do so, the 3 relevant tweets that were transferred from the selected to the discarded set and the 7 irrelevant tweets that were transferred from the discarded to the selected set have been firstly identified. Then, the causes of their transfer were studied, presented in the following table (Table 26).

[30] In the theoretical study performed in Section 7.2 the first filter was used.

	Cause	Comments	Number of tweets
RELEVANT TWEETS	No identification of locations	Locations are not provided in capital letters, or are of a relative type	3
IRRELEVANT TWEETS	No identification of copied tweets	The texts don't adopt any typical structure used to identify copies	3
	Incorrect identification of locations	Rosette doesn't distinguish precision of locations	4

Table 26: Causes of the mismatch between the theoretical and practical results.

Causes in the table highlight that the main source of mistakes is derived from the location identification in tweets' texts. In this sense, two different problems coexist: locations provided but no identified, and non-considered locations identified. Locations not identified are mainly consequence of the lack of use of capital letters and the use of relative terms, such as "town centre", to refer to places. Non-considered locations during the annotation process refer to those locations not provided with an acceptable degree of precision, which Rosette is not able to recognize and therefore identifies. The match (accuracy) between the manual annotation of locations and the outcome provided by Rosette considering all the tweets in the dataset (148 tweets) is of an 93,24%, which is considered high but insufficient, as just seen.

The non-identification of copied tweets leads to the incorrect selection of three tweets. This circumstance is consequence of a unique structure adopted in the texts of these tweets, different from the usual forms adopted by texts referred to copies. The match (accuracy) between the manual annotation of copied tweets and the outcome provided by the tool is of an 98,16%, which is considered very high and hardly improvable, since the mismatch refer to unique structures and therefore vary in every case.

Finally, the following table (Table 27) presents the performance of the tool, highlighting the obtained recall, precision and F-value values for each of the sets, and the accuracy obtained considering both. Also, between parenthesis is highlighted the gap with regards to the theoretical results.

	Selected set	Discarded set
Recall	63,15 % (-15,8%)	88,37 % (-5,4%)
Precision	44,44 % (-20,7%)	94,21 % (-2,6%)
F-value	52,16 % (-19,2%)	91,19 % (-4,1%)
Accuracy	85,13 % (-6,76 %)	

Table 27: Recall, precision, and F-value (performance) values of both selected and discarded sets, along with the accuracy value of the tool. Between parenthesis is highlighted the gap with the theoretical results.

The selected set reaches a performance of a 52,16%, which is considered as an acceptable value. Nevertheless, it can be observed that the decrease in accuracy has been much lower, nearly three times less, obtaining an 85,13% which is considered as high.

With the aim of reaching similar performance results to those obtained theoretically, it is evidenced the need of a new algorithm capable to detect all the locations provided in texts and discard those ones provided without a minimum level of precision to be incorporated to the tool, since Rosette has been proved to use a too inaccurate algorithm. Having disposed of an algorithm able to match with the manual annotation at a 100%, performance and accuracy results would have been of 66,66% and 89,86%, respectively.

8.3.1.2. First filter not considered

8.3.1.2.1. Case's study related cascades

This section aims to find out how performance results, in particular precision, differ in the case the first filter of the process is not used (skipped) with the obtained in Section 8.3.1.1, which used the step, in the case that the input dataset contains only L0 tweets related to the case study. Therefore, in this section reply tweets originated from the 310 L0 tweets [31](see Figure 22) related to the case study, without segmenting them according to their relevance, are studied. Results obtained in this section are compared with the ones registered in Table 27 (corresponding to Section 8.3.1.1).

[31] The fact of 310 L0 tweets is equal to say 310 cascades, since from each L0 tweet a cascade is originated.

It must be annotated that during the data collection process executed by the tool (explained in Section 8.1.1) four L0 tweets, and therefore their corresponding reply tweets, and also other reply tweets that were collected four months before in Section 4.2 have not been collected because either they have been eliminated or their corresponding user has changed its account to 'private' during these months. This circumstance entails that the number of tweets finally studied are not equal to those presented in Figure 22.

In the table below (Table 28) is presented the number of tweets studied and the quantity obtained in each of the sets, selected and discarded, and segmented by their relevance.

	Total tweets	Relevant	Irrelevant
Reply tweets collected (from L1 to L3) (first filter not used)	214	19	197
Selected set	41	12	29
Discarded set	173	7	166

Table 28: Number of tweets (total, relevant and irrelevant) in each of the studied sets obtained considering cascades related to the case study.

Finally, the following table (Table 29) presents the performance, highlighting the obtained recall, precision and F-value values for each of the sets, and the accuracy obtained considering both. Also, between parenthesis is highlighted the gap with regards to the results obtained in Table 27.

	Selected set	Discarded set
Recall	63,15%	84,26% (-4,1%)
Precision	29,27% (-15,1%)	95,95% (+1,7%)
F-value	40,00% (-12,1%)	89,72% (-1,4%)
Accuracy	83,17% (-2,0%)	

Table 29: Recall, precision, and F-value (performance) values of both selected and discarded sets, along with the accuracy value considering cascades related to the case study. Between parenthesis is highlighted the gap with the Table 27 results.

As it is observed, the recall in the selected set is maintained because the relevant tweets under study are the same ones, and therefore are classified in the same way.

The precision in the selected set has been reduced because of the introduction of only irrelevant tweets, in other words, because the precision of the input dataset is not 100%. Nonetheless, this circumstance was already expected (see discussion 7.2.1) inasmuch as precision only could have maintained or decreased since the precision of the classification method has not changed.

Then, it is proceeded to calculate the Factor ET, concept introduced in Section 7.2.1, considering the precision of the classification method obtained in Table 27:

$$Overall\,precision = Factor\ ET \cdot Precision\ CM$$

$$Factor\ ET = \frac{Overall\,precision}{Precision\ CM} = \frac{0{,}2927}{0{,}4444} = 0{,}6586$$

As it is seen, the value of the Factor ET is very close to the percentage of relevant L0 tweets in the dataset, which is of 66,13% [32](Figure 55), namely the precision of the input dataset. This circumstance is due to the fact that cascades related to the case study behave in the same way regardless of their relevance, meaning that they own on average a very similar number of tweets in each level as demonstrated in Figure ii (Section ii in the Appendix).

8.3.1.2.2. All cascades

This section aims to find out how performance results, in particular precision, differ with the obtained in Section 8.3.1.1 and 8.3.1.2.1 considering all the tweets provided in the initial dataset[33], including those which deal with different topics unrelated to the case study and were eliminated[34] (52 tweets) after performing the annotation process of Section 4.3. Hence, in this sect ion the first filter of the

[32] 66,13% is the percentage presented in the analyses of Section 6.1 considering 310 L0 tweets. However, as four irrelevant L0 tweets were not found, the percentage changed to 66,99%.

[33] Initial dataset provided refers to the dataset introduced in the beginning of Section 4.2.

[34] L0 tweets, and therefore cascades, annotated as out of topic were eliminated before the start of the analyses, as described in Section 4.3.

process is also not used (skipped). Therefore, in this section reply tweets originated from 358 L0 tweets, without segmenting them according to their relevance, are studied. Results obtained in this section are compared with the ones registered in Table 27 and Table 29 (corresponding to Section 8.3.1.1 and 8.3.1.2.1, respectively).

In the table below (Table 30) is presented the number of tweets studied and the quantity obtained in each of the sets, selected and discarded, and segmented by their relevance.

	Total tweets	Relevant	Irrelevant
Reply tweets collected (from L1 to L3) (first filter not used)	376	19	357
Selected set	76	12	64
Discarded set	300	7	293

Table 30: Number of tweets (total, relevant and irrelevant) in each of the studied sets obtained considering all the cascades.

Finally, the following table (Table 31) presents the performance, highlighting the obtained recall, precision and F-value values for each of the sets, and the accuracy obtained considering both. Also, between parenthesis is highlighted the gap with regards to the results obtained in Table 27 and Table 29.

	Selected set	Discarded set
Recall	63,15%	82,07% (-6,3% / -2,2%)
Precision	15,78% (-28,6% / -13,5%)	97,66% (+3,4% / +1,7%)
F-value	25,25% (-26,9 / -14,7%)	89,19% (-2,0% / -0,5%)
Accuracy	81,11% (-4,0% / -2,0%)	

Table 31: Recall, precision, and F-value (performance) values of both selected and discarded sets, along with the accuracy value considering all the cascades. Between parenthesis is highlighted the gap with the (Table 27 / Table 29) results.

The maintenance and decrease of recall and precision levels, respectively, in the selected set are also consequence of the reasoning exposed in the previous section. Nonetheless, as the number of irrelevant tweets has increased compared to the previous case, the precision value has been further reduced.

Then, it is proceeded to calculate the Factor ET considering the precision of the classification method obtained in Table 27:

$$Overall\,precision = Factor\ ET \cdot Precision\ CM$$

$$Factor\ ET = \frac{Overall\,precision}{Precision\ CM} = \frac{0{,}1578}{0{,}4444} = 0{,}3550$$

In contradistinction to the Factor ET value obtained in the previous section, this one is far to the percentage of relevant L0 tweets in the dataset, namely the precision of the input dataset, which is of 57,26%.[35]. This circumstance is due to the fact that cascades out-of-topic behave differently to those related to the case study. For example, during the annotation process performed in Section 4.3 it was observed that cascades originating from a L0 tweet related to football encompass a very high number of reply tweets, unseen in cascades dealing with the floods.

In conclusion, the poor precision result obtained in this section highlights the need to provide to the classification process a precise input dataset to extract their replies in the case the first filter of the filtering step (manual annotation) is not feasible to be used. This is especially critical when the input dataset includes tweets unrelated to the case study in question since they behave differently, fact that causes worse performance results.

8.3.2. Hurricane Harvey 2017 storms in Texas dataset

The dataset used in this section for the evaluation allow the verification, in terms of precision, of the proposed classification method designed in the previous chapter. This fact is equivalent to say that the identified patterns are indeed common for all the datasets originated from natural disasters, and therefore were not owned from a specific one (2014 Southern England floods).

Unlike the previous dataset, no previous manual annotation of the tweets has been performed. Furthermore, because of the high volume of the dataset, which contains 468 cascades, annotation will be just performed to those tweets included

[35] From the 358 L0 tweets: 205 are relevant (57,26%), 101 irrelevant (28,21%) but related to the case study, and 52 out-of-topic (14,52%).

in the selected set. Hence, as the total number of relevant reply tweets is uncertain, only performance values corresponding to the precision of the selected set can be computed.

Moreover, for the same dataset's volume reason, it is unfeasible to perform the first filter included within the first step of the classification process, which is the manual annotation of L0 tweets (see discussion in Section 7.2.1). Consequently, to get the precision of the classification method, the overall precision obtained in the output dataset (selected set) will be corrected with the Factor ET.

Accordingly, the classification method precision will be calculated as follows:

$$Precision\ CM = \frac{Overall\ precision}{Factor\ ET}$$

As already explained in Section 7.2.1, the Factor ET depends on the precision of the provided dataset. Nevertheless, this value is unknown for this case study. Hence, the hypothesis stating that the precision of this dataset is equal[36] to the one provided in Section 8.3.1.2.2 is used. Therefore, it is employed the Factor ET value obtained from that section to calculate the classification method precision:

$$Precision\ CM = \frac{Overall\ precision}{0{,}3550}$$

During the reply's tweets collection step executed by the tool, it was noticed that there were L0 tweets presenting an extremely high number of replies (in some cases more than 6000 replies). Since manual annotation of the selected set of reply tweets must be subsequently performed, it was decided to keep in the dataset only L0 tweets originating no more than 30 replies for the study. Hence, the number of tweets contained in the initial dataset went from being 468 to 314.

These 314 L0 tweets leaded to obtain a total of 1585 reply tweets, of which 243 were included in the selected set and 1342 were discarded.

In Table 32 is presented the number of tweets obtained in the selected set, segmented by their relevance.

[36] This hypothesis is based on the fact that the dataset was provided by the same source. Knowing that the source used the same algorithm to extract the tweets, a very similar precision reliably was obtained.

	Total tweets	Relevant	Irrelevant
Selected set	243	22	221

Table 32: Number of tweets (total, relevant and irrelevant) included in the selected set.

In front of this larger number of irrelevant reply tweets selected, resulting in a very low precision of a 9,05%, the causes originating it are to be identified and analysed.

The first of the two causes responsible for the selection of so many irrelevant messages is the massive reference to non-specific locations in tweet texts, and therefore not valuable, such as Texas and America. Next table (Table 33) highlights this fact, presenting the number of tweets providing non-specific locations, segmenting by their relevance. The non-specific locations considered in this study are: Texas (and Tx), Houston, America, U.S., and U.S.A. It must be annotated that tweets providing any of these non-specific locations along with other locations are not included in the "Non-specific location provided" set.

	Total tweets	Location provided	Non-specific location provided
Relevant	22	13	1
Irrelevant	221	140	89

Table 33: Total, providing a location and providing a non-specific location number of tweets, segmenting by their relevance.

From the table can be observed that in the case that tweets providing any of these five non-specific locations were not considered as location providers, and therefore not selected in that corresponding step, precision level would largely increase.

The second of the two causes is the insertion of memes and GIFs, totally irrelevant, as media in tweets, which are finally selected during the image selection step of the classification process. This circumstance arisen in this case study was not occurred in the previous one, mainly because memes were not used in 2014 and Twitter didn't allow to enclose GIFs as media in tweets. Next table (Table 34) highlights this fact, presenting the number of tweets including irrelevant memes and GIFs.

	Total tweets	Selected during image selection step (thus including media)	Including memes or GIFs
Relevant	22	2	0
Irrelevant	221	62	46

Table 34: Total, selected in the image selection step, and including memes or GIFs number of tweets, segmenting by their relevance.

From the table can be observed that the insertion of memes and GIFs as media very negatively affects the precision in the set, since it represents the 20,8% of irrelevant tweets selected. Nevertheless, this cause cannot be currently addressed due to the lack of techniques to separate images according to their content. Moreover, the image selection step introduced in the classification method is not convenient to be removed because of the importance [37]of images in rapid mapping purposes, fact the makes prioritise recall rather than precision levels.

In order to improve the performance of the classification method in this particular case study, the first of the causes is addressed. Therefore, the classification of reply tweets is executed again ignoring the identification as locations of those non-specific locations previously mentioned. Hence, the table below (Table 35) presents the number of tweets obtained in the selected set in this second simulation, segmented by their relevance.

	Total tweets	Relevant	Irrelevant
Selected set	153	21	132

Table 35: Number of tweets (total, relevant and irrelevant) included in the selected set (second simulation).

Finally, the following table (Table 36) presents the precision obtained of the selected set.

	Selected set
Precision obtained (overall)	13,72%

Table 36: Precision value (overall) of the selected set.

[37] The reasons of this importance were presented in Section 6.1, 6.2.2.1.2 and 7.1.

Then, the Factor ET value is applied to calculate the classification method precision:

$$Precision\,CM = \frac{Overall\,precision}{0{,}3550} = \frac{0{,}1372}{0{,}3550} = 0{,}3865 = 38{,}65\%$$

The precision of the classification obtained is lower (5,79%) compared to the one achieved in the previous case study of 44,44%, presented in Table 27. Nonetheless, this negative difference is considered to be direct consequence of the insertion of memes and GIFs, fact not occurred in the first case study.

Hence, the classification method proposed in Chapter 7 has demonstrated its effectiveness, and therefore its usefulness, to be employed in other natural disasters situations for the extraction of reply tweets relevant to enhance situational awareness. Nevertheless, the main finding resulting from this case study is that some steps are convenient to be slightly adapted according to the content of tweets originated in each situation to improve the performance of the process.

Chapter 9

Conclusion and future work

In this thesis an automatic extraction method of relevant reply messages originated during emergencies in the social media of Twitter was presented, to be subsequently used by humanitarian organizations in order to enhance situational awareness with the aim of improving the decision-making of operations.

Firstly, a descriptive analysis of cascades was conducted to gain insights on their behaviour. Findings in that section, such as most of the tweets were posted in lower levels, were used later during the design process of the classification method to improve its efficiency.

Then, an analysis regarding the relevance of cascades and tweets was performed with the goal of finding patterns that could allow the effective extraction of those tweets considered relevant in the classification method.

The proposed classification process has been theoretically evaluated through the set of tweets corresponding to the case study of the Southern England 2014 floods, and has demonstrate very high performance and accuracy values, accounting more than 70% and 90% respectively. Hence, the different filters and steps considered, resulting of the previous analyses, have been proved to be efficient and effective. However, the resilience of the method in terms of tweets' language is not guaranteed, since in a step two unigrams employing English language are used and therefore the performance could be negatively affected when applied to a set of tweets in different language.

Finally, a tool (ReTREC) incorporating this classification method has been created to automatically extract relevant tweets' replies. The tool stands out for its usage simplicity and for the comfortable design of its corresponding interface. Furthermore, it allows the extraction of tweets in real-time, and its generated database enable queries in the case the user was just interested in tweets containing certain features such as media types.

The practical evaluation performed through the tool by means of the Southern England 2014 floods and Hurricane Harvey 2017 storms in Texas case studies confirmed the effectiveness of the classification method. Nonetheless, performance results obtained using the tool were not as high as the ones computed theoretically. This circumstance was due to the limitations of the algorithm used to identify locations in texts and the complexity to automatically detect copies between tweets. Notwithstanding, they account for a 52% and 85% in terms of performance and accuracy, respectively, which are still considered high values.

The process corresponding to the extraction and classification of reply tweets is only the second stage of an overall process whose first phase is the extraction of tweets through queries into the API Twitter server. Then, being not an independent stage, output results obtained directly depends on the performance of the first phase, particularly on its precision. Accordingly, it has been demonstrated throughout the evaluation of the classification method that in order to reach the above-commented performance levels, it is fundamental to firstly provide a precise dataset as input.

This thesis represents the first attempt ever done on considering reply messages as a source to extract additional information from Twitter. Consequently, the work presented can be employed as a basis to frame the problem for subsequent optimizations regarding the effectiveness of the classification process, since it is considered that there is still scope for improvement.

Future improvements of the classification process could be focused on the searching of more patterns considering the text in tweets. Consequently, a possible approach could be the creation of more text types and subtypes, in order to subsequently find, either manually or employing machine learning techniques, more precise n-grams according to each type and therefore increase the overall performance. Furthermore, the substitution of the image selection step with an image analysis step analogous to the one performed for text, meaning that images would be analysed according to their relevance, definitely could enhance the precision of the classification method.

Future work on the tool would consist of the introduction of a more accurate method to detect copies in tweets, and of a new algorithm able to detect locations provided with a precision higher than an established threshold and resilient to ambiguous texts. Besides, the implementation of a technique in those algorithms able to detect those relative location references (i.e. “town centre”) regardless of the language employed would help to improve recall values. Finally, the possibility of combining the programmed functions already running in the tool with existing functions encompassed in other related work must be evaluated since it could help increasing the performance of the method.

Bibliography

[1] Alan M. MacEachren, Anthony C. Robinson, Anuj Jaiswal, Scott Pezanowski, Alexander Savelyev, Justine Blanford, and Prasenjit Mitra. *Geo-Twitter Analytics: Applications in Crisis Management*. Proceedings 25th International Carthographic Conference, 2011.

[2] Alex Bruns, and Jean Burgess. Crisis Communication in Natural Disasters: The Queensland Floods and Christchurch Earthquakes. *Twitter and Society*, 373 – 384, 2014.

[3] Amartya Hatua, Trung T. Nguyen, and Andrew H. Sung. An Approach for Pattern Recognition and Prediction of Information Diffusion Model on Twitter. *World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering,* Vol:12, No:3, 2018.

[4] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A Review of Machine Learning Algorithms for Text-Documents Classification. *Journal of advances in information technology*, Vol:1, No:1, 2010.

[5] Benjamin Herfort, Joao Porto de Albuquerque, Svend-Jonas Schelhorn, and Alexander Zipf. Does the spatiotemporal distribution of tweets match the spatiotemporal distribution of flood phenomena? A study about the River Elbe Flood in June 2013. *International Conference on Information Systems for Crisis Response and Management*, 11, 2014.

[6] Bernd Resch, Florian Usländer, and Clemens Havas. Combining machine-learning topic models and spatiotemporal analysis of social media for disaster footprint and damage assessment. *Cartography and Geographic Information Science*, 2017.

[7] Carlos Castillo. Big Crisis Data: Social Media in Disasters and Time-Critical Situations. Cambridge University Press, 2016.

[8] Chiara Francalanci, Barbara Pernici, and Gabriele Scalia. Exploratory spatio-temporal queries in evolving information. *IEEE RCIS Conference Brighton 2017*, 2017.

[9] Chiara Francalanci, Paolo Guglielmino, Matteo Montalcini, Gabriele Scalia and Barbara Pernici. IMEXT: A method and system to extract geolocated images from Tweets – Analysis of a case study. *ULDB MATES workshop Springer*, 2017.

[10] Demetris Antoniades, and Constatine Dovrolis. Co-evolutionary dynamics in social networks: a case study on Twitter. *Computational Social Networks, 2:14*, 2015.

[11] Dong Li, Yongchao Zhang, Zhiming Xu, Dianhui Chu, and Sheng Li. Exploiting Information Diffusion Feature for Link Prediction in Sina Weibo. *Scientific Reports, 6*, 2016.

[12] Eunae Yoo, William Rand, Mahyar Eftekhar, and Elliot Rabinovich. Evaluating information diffusion speed and its determinants in social media networks during humanitarian crises. *Journal of Operations Management*, 45: 123 – 133, 2016.

[13] Farhad Laylavi, Abbas Rajabifard, and Mohsen Kalantari. Event relatedness assessment of Twitter messages for emergency response. *Information Processing and Management*, 53: 266 – 280, 2017.

[14] Farida Vis, Simon Faulkner, Katy Parry, Yana Manykhina, and Lisa Evans. Twitpic-ing the Riots. Analysing Images Shared on Twitter during the 2011 U.K. Riots. *Twitter and Society*, 385 – 398, 2013.

[15] Freddy Chong Tat Chua, and Sitaram Asur. Automatic Summarization of Events from Social Media. *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*, 81 – 90, 2013.

[16] Haji Mohammad Saleem, Yishi Xu, and Derek Ruths. Novel Situational Information in Mass Emergencies: What does Twitter provide? *Procedia Engineering*, 78: 155 – 164, 2014.

[17] Ido Guy, Victor Makarenkov, Niva Hazon, Lior Rokach, and Bracha Shapira. Identifying Informational vs. Conversational Questions on Community Question Answering Archives. *WSDM'18*, 2018.

[18] J.Fohringer, D.Dransch, H.Kreibich, and K.Schröter. Social media as an information source for rapid flood inundation mapping. *Natural Hazards Earth System Sciences*, 15: 2725 – 2738, 2015.

[19] James H. Fowler, and Nicholas A. Christakis. Cooperative behaviour cascades in human social networks. *PNAS*, Vol: 107, No: 12, 5334 – 5338, 2010.

[20] Ji Ao, Peng Zhang, and Yanan Cao. Estimating the Locations of Emergency Events from Twitter Streams. *Procedia Computer Science 31*, 731 – 739, 2014.

[21] Jiang Yang, and Scott Counts. Predicting the Speed, Scale and Range of Information Diffusion in Twitter. *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[22] Ling Feng, Yanqing Hu, Baowen Li, H.Eugene Stanley, Shlomo Havlin, and Lidia A. Braunstein. Competing for Attention in Social Media under Information Overload Conditions. *PLoS ONE 10(7),* 2015.

[23] Luke Sloan, and Jeffrey Morgan. Who Tweets with Their Location? Understanding the Relationship between Demographic Characteristics and the Use of Geoservices and Geotagging on Twitter. *PLoS ONE 10(11),* 2015.

[24] Mohammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier. Extracting Information Nuggets from Disaster-Related

Messages in Social Media. *Proceedings of the 10th International ISCRAM Conference,* 2013.

[25] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. Twitter as a Lifeline: Human-annotated Twitter Corpora for NLP of Crisis-related Messages. 2016.

[26] Patric Meier. Digital Humanitarians: How Big Data Is Changing the Face of Humanitarian Response. *Bioethical Inquiry*, 14: 567 – 569, 2017.

[27] Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Polyglot - NER: Massive Multilingual Named Entity Recognition. *Proceedings of the 2015 SIAM International Conference on Data Mining*, 2015.

[28] Rowan Wilken. Twitter and Geographical Location. *Twitter and Society*, 155 – 167, 2014.

[29] Starr Roxanne Hiltz, Jane Kushma, and Linda Plotnick. Use of Social Media by U.S. Public Sector Emergency Managers: Barriers and Wish Lists. *Proceedings of the 11th International ISCRAM Conference*, 2014.

[30] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Tweet Analysis for Real-Time Event Detection and Earthquake Reporting System Development. *IEEE Transactions on Knowledge and Data Engineering*, Vol:25, No:4, 2013.

[31] Tatsuro Kawamoto. A stochastic model of tweet diffusion on the Twitter network. *Physica A*, 392: 3470 – 3475, 2013.

[32] Thomas Risse, Wim Peters, Pierre Senellart, and Diana Maynard. Documenting Contemporary Society by Preserving Relevant Information from Twitter. *Twitter and Society*, 207 – 219, 2014.

[33] Tomer Simon, Avishay Goldberg, and Bruria Adini. Socializing in emergencies – A review of the use of social media in emergency situations. *International Journal of Information Management*, 35: 609 – 619, 2015.

[34] Víctor Fresno, Arkaitz Zubiaga, Heng Ji, and Raquel Martínez. Exploiting Geolocation, User and Temporal Information for Monitoring Natural Hazards on Twitter. *Procesamiento del Lenguaje Natural*, 54: 85 – 92, 2015.

[35] Xiaohua Liu, Yitong Li, Haocheng Wu, Ming Zhou, Furu Wei, and Yi Lu. Entity Linking for Tweets. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1304 – 1311, 2013.

[36] Yadong Zhou, Beibei Zhang, Xiaoxiao Sun, Qinghua Zheng, and Ting Liu. Analyzing and modelling dynamics of information diffusion in microblogging social network. *Journal of Network and Computer Applications*, 86: 92 – 102, 2017.

[37] Ying Hu, Rachel Jeungeun Song, and Min Chen. Modeling for Information Diffusion in Online Social Networks via Hydrodynamics. *Special section on socially enabled networking and computing, IEEE*, 128 – 135, 2017.

[38] Zheng Xu, Hui Zhang, Vijayan Sugumaran, Kim-Kwang Raymond Choo, Lin Mei, and Yiwei Zhu. *EURASIP Journal on Wireless Communications and Networking,* 44, 2016.

Appendix

i. Annotation criterion

COPY/RETWEET

YES if:

- Tweet (text + media (in case)) is very similar (added text not contain new relevant information) or exact as the level 0 tweet in the same cascade (if new text or new media doesn't apply) (if it is in another cascade doesn't apply).

NO if:

- When not YES.

LOCATION

(Corpus = Text without considering the location in it)

YES if (necessary condition: location has to be provided <u>and</u> his_precision higher than 1, otherwise is NO) (here we not consider location as part of text = corpus):

- Corpus is 1 <u>and</u> corpus related to location (media doesn't care).
- If Corpus=0, but media is related to location <u>and</u> is 1.

NO if:

- Location Precision lower than 1.
- When not YES:
  - Corpus=0 <u>and</u> [media is not related to location <u>or</u> is 0 <u>or</u> is None].
  - Corpus=1 <u>and</u> [text not related to location].

LOCATION PRECISION

- 4 if it is detailed (street numbers and/or specific name of buildings or similar).
- 3 if it provides street's names (or similar).
- 2 if it provides city's zone names (or similar).
- 1 if it provides city's names (or similar).

TEXT

1 if :

- Location is YES (here location is part of text = No corpus).
- Any information that allows to assess flood conditions.
- Any information that allows to assess outdoor damages caused by the flood.

OT if:

- Not related with England floods of 2014.

0 if:

- When not 1 or OT.
- If it is a reply of a tweet of 0 and 1, even if it is considered OT.
- If it is a text used in an upper level, but it contains a different media item (must use media by obligation).

ORIGINAL USER

YES if:

- The tweet is issued by the same user of the tweet in level 0 in the same cascade.

NO if:

- When not YES.

URL

1 if:

- Text inside the webpage meets the '1' established conditions of the tweet text.
- Media inside the webpage meets the '1' established conditions of the tweet media.

OT if:

- Not related with England floods of 2014.

NA if:

- No URL provided / Cannot access to the webpage.

0 if:

- When not NA, 1 or OT.

MEDIA

1 if (in the case of photos, if at least 1 image is OK):

- Referred to outdoor captures (necessary condition) (no maps/screenshots/etc).
- Any clear enough media (in terms of quality) that allows to assess flood conditions.
- Any clear enough media (in terms of quality) that allows to assess outdoor damages caused by the flood.

OT if:

- Not related with England floods of 2014.

0 if:

- When not 1 or OT.
- If same media used of an upper level, but the text differs of that same tweet.

ii. Figures related to cascade structure characteristics considering their relevance

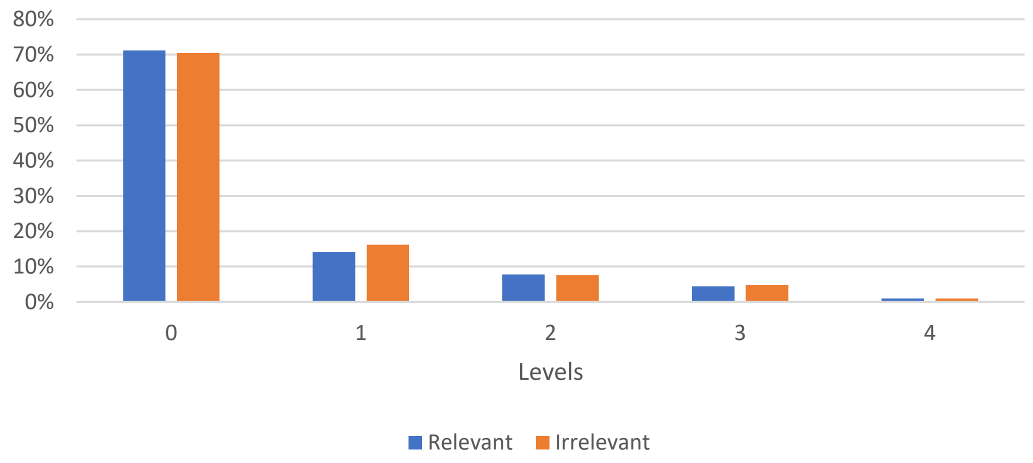


Figure i: Percentage of cascades that share the maximum level reached, for levels 1-4, and segmented by their relevance.

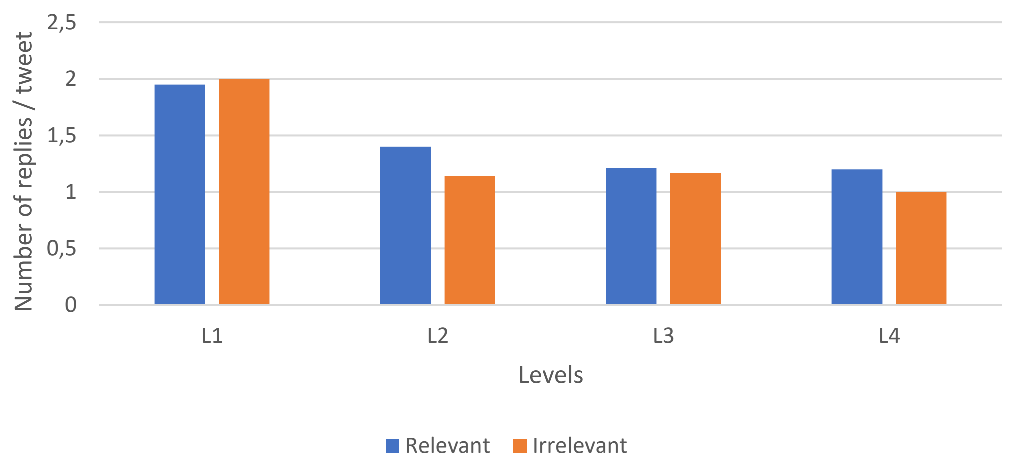


Figure ii: Average number of tweets per cascade, for levels 1-4, and segmented by the relevance of their corresponding cascades.

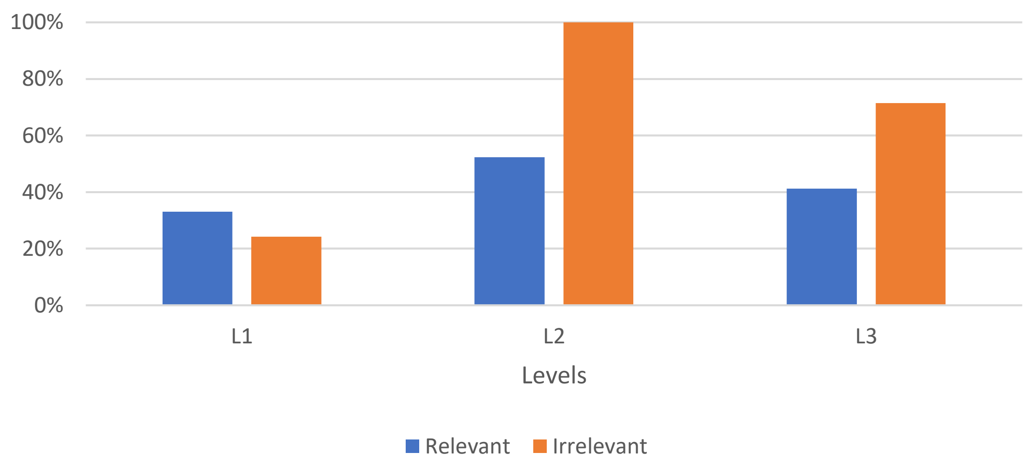


Figure iii: Percentage of replied tweets per cascade, for levels 1-3, in the case the cascade has reached the corresponding levels, and segmented by the relevance of these cascades.

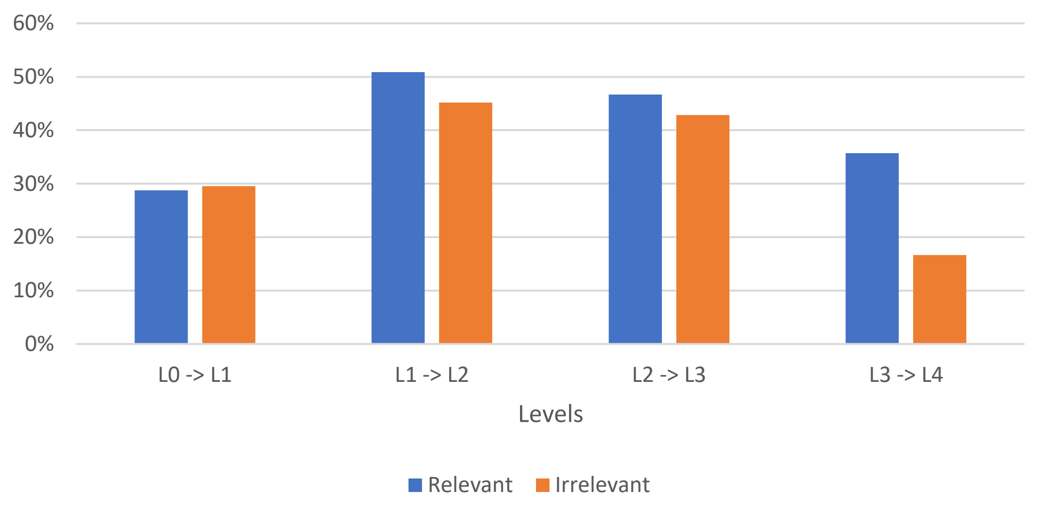


Figure iv: Percentage of cascades that once reached a level, reach the following one, for levels 1-3, and segmented by their relevance.

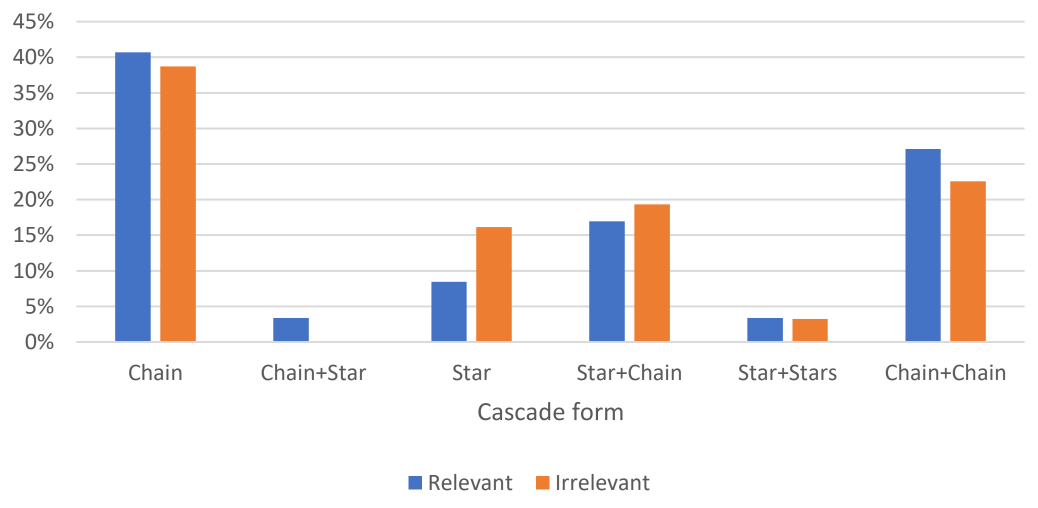


Figure v: Percentage of cascades that share the same form, for each form, and segmented by their relevance.

iii. Programming code (in Python language)

a. Files and functions used in the Reply's tweets network chapter (Chapter 4)

File: p.py / Function: ex_parents

```python
from bs4 import BeautifulSoup
import requests
from selenium import webdriver
from selenium.common.exceptions import StaleElementReferenceException, Timeout
     Exception
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd
import pandas
from openpyxl import load_workbook

def ex_parents():
   excel_file='a.xls'
   sheet='TWEETS'
   ex=pandas.read_excel(excel_file, sheet)
   column='id'
   values=ex[column].values
   tweet_list=[]
   df=pd.DataFrame(columns=['0_par_tweet','1_n_tweets'])
   for tweet in values:
      str_tweet=[str(tweet)]
      tweet_list.append(str_tweet)
   del(tweet_list[0]) #anar en compte
   for tweet in tweet_list:
      par_tweet=prova_par(tweet[0])
      row=pd.DataFrame({'0_par_tweet':[[par_tweet[0]]],'1_n_tweets':[par_tweet[
       1]]})
      df=df.append(row,ignore_index=True)
   writer=pd.ExcelWriter('parents.xls',engine='xlsxwriter')
   df.to_excel(writer, sheet_name='TWEETS')
   workbook=writer.book
   workbook.add_format({'bold':True,'valign':'top','fg_color':'#D7E4BC'})
   worksheet=writer.sheets['TWEETS']
   worksheet.set_column('B:B',60,None)
   writer.save()
```

File: p.py / Function: prova_par

```
def prova_par(tweet_id):
  page=requests.get('http://twitter.com/statuses/'+tweet_id)
  soup=BeautifulSoup(page.content, 'html.parser')
  parent=soup.find_all('div',attrs={'data-replied-tweet-id':True})
  direct_par=True
  partial_id=tweet_id
  quote_tweet=False
  while len(parent)>0:
    direct_par=False
    page=requests.get('http://twitter.com/statuses/'+(str(parent[0]['data-
     replied-tweet-id'])))
    partial_id=str(parent[0]['data-replied-tweet-id'])
    ask_quote=soup.find_all('a',attrs={'class':'QuoteTweet-link js-nav'})
    ask_n_tweets=soup.find_all('div',attrs={'data-tweet-id':True})
    ask_n_tweets=len(ask_n_tweets)
    if len(ask_quote)>0:
      quote_tweet=str(ask_quote[0]['data-conversation-id'])
    soup=BeautifulSoup(page.content, 'html.parser')
    parent=soup.find_all('div',attrs={'data-replied-tweet-id':True})
  if direct_par==True:
    ask_quote=soup.find_all('a',attrs={'class':'QuoteTweet-link js-nav'})
    ask_n_tweets=soup.find_all('div',attrs={'data-tweet-id':True})
    ask_n_tweets=len(ask_n_tweets)
    if len(ask_quote)>0:
      quote_tweet=str(ask_quote[0]['data-conversation-id'])
  level0_par=partial_id
  return level0_par,ask_n_tweets
```

File: p.py / Function: ex_cascades

```
def ex_cascades():
  excel_file='parents.xls'
  sheet='TWEETS'
  ex=pandas.read_excel(excel_file, sheet)
  column0='0_par_tweet'
  column1='1_n_tweets'
  values0=ex[column0].values
  values1=ex[column1].values
  tweet_list=[]
```

```
    for tweet in values0:
        str_tweet=tweet[2:][:-2]
        tweet_list.append(str_tweet)
    df=pd.DataFrame(columns=['0_cascade','1_level','2_id_tweet','3_id_parent','4_id_replies'])
    cas=1
    n=0
    for tweet in tweet_list:
        if values1[n]==1:
            cas_df=pd.DataFrame(columns=['0_cascade','1_level','2_id_tweet','3_id_parent','4_id_replies'])
            replies=['-']
            row=pd.DataFrame({'0_cascade':[cas],'1_level':[0],'2_id_tweet':[[tweet]],'3_id_parent':[['-']],'4_id_replies':[replies]})
            cas_df=df.append(row,ignore_index=True)
        else:
            cas_df=tree(tweet,cas)
        cas=cas+1
        n=n+1
        book=load_workbook('tree_pestanyes.xlsx')
        writer=pd.ExcelWriter('tree_pestanyes.xlsx',engine='openpyxl')
        writer.book=book
        cas_df.to_excel(writer, sheet_name='TWEETS', startrow=1)
        workbook=writer.book
        worksheet=writer.sheets['TWEETS']
        writer.save()
```

File: p.py / Function: tree

```
def tree(tweet_par_id,cascade):
    df=pd.DataFrame(columns=['0_cascade','1_level','2_id_tweet','3_id_parent','4_id_replies'])
    act_level=0
    replies=prova_nova(tweet_par_id)
    if replies==None:
        replies=['-']
        row=pd.DataFrame({'0_cascade':[cascade],'1_level':[act_level],'2_id_tweet':[[tweet_par_id]],'3_id_parent':[['-']],'4_id_replies':[replies]})
        df=df.append(row,ignore_index=True)
        return df
    row=pd.DataFrame({'0_cascade':[cascade],'1_level':[act_level],'2_id_tweet':[[tweet_par_id]],'3_id_parent':[['-']],'4_id_replies':[replies]})
    df=df.append(row,ignore_index=True)
```

```
  while (df['1_level']==act_level).sum()>0:
    for line in df.values:
      if (line[1]==act_level and line[4]!=['-']):
        for tweet in line[4]:
          replies=prova_nova(tweet)
          if replies!=None:
            row=pd.DataFrame({'0_cascade':[cascade],'1_level':[act_level+1],
     '2_id_tweet':[[tweet]],'3_id_parent':[line[2]],'4_id_replies':[replies]})

            df=df.append(row,ignore_index=True)
          elif replies==None:
            row=pd.DataFrame({'0_cascade':[cascade],'1_level':[act_level+1],
     '2_id_tweet':[[tweet]],'3_id_parent':[line[2]],'4_id_replies':[['-']]})
            df=df.append(row,ignore_index=True)
    act_level=act_level+1
  return df
```

File: p.py / Function: prova_nova

```
class wait_for_more_than_n_elements_to_be_present(object):
  def __init__(self, locator, count):
    self.locator = locator
    self.count = count

  def __call__(self, driver):
    try:
      elements = EC._find_elements(driver, self.locator)
      return len(elements) > self.count
    except StaleElementReferenceException:
      return False


def prova_nova(tweet_id):
  url = 'http://twitter.com/statuses/'+tweet_id
  driver = webdriver.Chrome()
  driver.maximize_window()
  driver.get(url)

  # initial wait for the tweets to load
  wait = WebDriverWait(driver, 10)
  wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, "li[data-item-
     id]")))
```

```python

# scroll down to the last tweet until there is no more tweets loaded
while True:
    tweets = driver.find_elements_by_css_selector("li[data-item-id]")
    number_of_tweets = len(tweets)

    driver.execute_script("arguments[0].scrollIntoView();", tweets[-1])

    try:
        wait.until(wait_for_more_than_n_elements_to_be_present((By.CSS_SELECTOR
      , "li[data-item-id]"), number_of_tweets))

    except TimeoutException:
        break

page_source = driver.page_source
driver.close()
soup = BeautifulSoup(page_source,'lxml')
reply_list_1=soup.find_all('div',attrs={'data-tweet-id':True})
if len(reply_list_1)==0:
    return None
start=False
while start==False:
    if len(reply_list_1)==0:
        return None
    elif str(reply_list_1[0]['data-tweet-id'])==tweet_id:
        name_original_user=str(reply_list_1[0]['data-screen-name'])
        del(reply_list_1[0])
        start=True
    else:
        del(reply_list_1[0])
if len(reply_list_1)==0:
    return None
reply_list_2=[]
for l in reply_list_1:
    if ('data-has-parent-tweet="true"' and 'data-mentions') in str(l):
        data_mentions_str=str(l['data-mentions'])
        data_mentions_list=data_mentions_str.split(' ')
        if name_original_user==data_mentions_list[0]:
            partial_id=str(l['data-tweet-id'])
            reply_list_2.append(partial_id)
if reply_list_2==[]:
    return None
else:
    return reply_list_2
```

File: p.py / Function: concatenar_ex

```
def concatenar_ex():
    excel_file='tree_pestanyes.xlsx'
    n=1
    df=pd.DataFrame(columns=['0_cascade','1_level','2_id_tweet','3_id_parent','4
        _id_replies'])
    while n<371: #canviar quan més pestanyes
        sheet='TWEETS'+str(n)
        ex=pandas.read_excel(excel_file, sheet)
        for line in ex.values:
            row=pd.DataFrame({'0_cascade':[line[0]],'1_level':[line[1]],'2_id_tweet
        ':[line[2]],'3_id_parent':[line[3]],'4_id_replies':[line[4]]})
            df=df.append(row,ignore_index=True)
        n=n+1
    writer=pd.ExcelWriter('tree_final.xls',engine='xlsxwriter')
    df.to_excel(writer, sheet_name='TWEETS')
    workbook=writer.book
    workbook.add_format({'bold':True,'valign':'top','fg_color':'#D7E4BC'})
    worksheet=writer.sheets['TWEETS']
    worksheet.set_column('D:D',30,None)
    worksheet.set_column('E:E',30,None)
    worksheet.set_column('F:F',100,None)
    writer.save()
```

File: status.py

```
import tweepy
import json
import pandas as pd
import pandas

def get_tw(num):

    consumer_key= ''
    consumer_secret=''

    access_token=''
    access_token_secret=''
```

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)


info=api.get_status(num)
j=info._json
dictlist=[]
for key,value in j.items():
   linia=[key,value]
   dictlist.append(linia)

obt_l_t=['id_str','text','truncated','retweet_count','favorite_count','creat
    ed_at','geo','place','coordinates','lang','source','user']
obt_l_u=['id_str','screen_name','name','verified','geo_enabled','location','
    description','followers_count','friends_count','statuses_count']
f_list=[]
userlist=[]

df=pd.DataFrame(columns=['00_t_id','01_t_url','02_t_text','03_t_truncated','
    04_t_retweets','05_t_likes','06_t_date','07_t_geo','08_t_place','09_t_coo
    r','10_t_lang','11_t_source','12_u_id','13_u_tname','14_u_name','15_u_ver
    ified','16_u_geo','17_u_place','18_u_description','19_u_followers','20_u_
    following','21_u_posts'])

for line in dictlist:
   if line[0] in obt_l_t:
      n=obt_l_t.index(line[0])

      if n==0:
         output='['+line[1]+']'
      elif (n==2 or n==3 or n==4 or n==6 or n==7 or n==8):
         output=str(line[1])

      elif n==11:
         output=[]
         for key,value in line[1].items():
            linia=[key,value]
            userlist.append(linia)
         for x in userlist:
```

```
        if x[0] in obt_l_u:
          w=obt_l_u.index(x[0])
          user_output=str(x[1])
          output.append([w,x[0],user_output])
      w=0
      output_2=[]
      while w<len(output):
        for linia in output:
          if w==linia[0]:
            output_2.append(linia)
        w=w+1
      output=output_2

    else:
      output=line[1]

    f_list.append([n,line[0],output])

n=0
f2_list=[]
while n<len(f_list)+1:
  for line in f_list:
    if n==line[0]:
      f2_list.append(line)
  n=n+1

return f2_list

url_tweet='https://www.twitter.com/statuses/'+num

row=pd.DataFrame({'00_t_id':[f2_list[0][2]],'01_t_url':[url_tweet],'02_t_tex
   t':[f2_list[1][2]],'03_t_truncated':[f2_list[2][2]],'04_t_retweets':[f2_l
   ist[3][2]],'05_t_likes':[f2_list[4][2]],'06_t_date':[f2_list[5][2]],'07_t
   _geo':[f2_list[6][2]],'08_t_place':[f2_list[7][2]],'09_t_coor':[f2_list[8
   ][2]],'10_t_lang':[f2_list[9][2]],'11_t_source':[f2_list[10][2]],'12_u_id
   ':[f2_list[11][2][0][2]],'13_u_tname':[f2_list[11][2][1][2]],'14_u_name':
   [f2_list[11][2][2][2]],'15_u_verified':[f2_list[11][2][3][2]],'16_u_geo':
   [f2_list[11][2][4][2]],'17_u_place':[f2_list[11][2][5][2]],'18_u_descript
   ion':[f2_list[11][2][6][2]],'19_u_followers':[f2_list[11][2][7][2]],'20_u
   _following':[f2_list[11][2][8][2]],'21_u_posts':[f2_list[11][2][9][2]]})

df=df.append(row,ignore_index=True)
return df
```

File: graph.py

```
import pandas as pd
import pandas
import networkx as nx
import matplotlib.pyplot as plt

def create(cascada):
   excel='Total_Dataset_stats.xlsx'
   sheet='TWEETS'
   ex=pandas.read_excel(excel, sheet)
   column0='2_id_tweet'
   column1='3_id_parent'
   column2='0_cascade'
   column3='1_level'
   values_id=ex[column0].values
   values_par=ex[column1].values
   values_cas=ex[column2].values
   values_lev=ex[column3].values
   first_list=[]
   n=0
   for line in values_cas:
      if line==cascada:
         first_list.append([values_id[n],values_par[n],values_lev[n]])
      n=n+1

   if len(first_list)==1:
      return None

   G=nx.DiGraph()
   for line in first_list:
      G.add_node(line[0])
   for line in first_list:
      if line[1]!=str(['-']):
         G.add_edge(line[1],line[0])

   n=0
   color_list=[]
   list_nodes=[]

   for node in G.nodes():
      list_nodes.append(node)
```

```
    while n<len(G.nodes()):
      for linia in first_list:
        if linia[0]==list_nodes[n]:
          if (linia[2]==0 or linia[2]==4 or linia[2]==8):
            color_list.append('r')
          elif (linia[2]==1 or linia[2]==5):
            color_list.append('b')
          elif (linia[2]==2 or linia[2]==6):
            color_list.append('g')
          elif (linia[2]==3 or linia[2]==7):
            color_list.append('y')
      n=n+1

    pos=nx.spring_layout(G)
    nx.draw_networkx_nodes(G, pos, node_color=color_list, cmap=plt.get_cmap('jet
       '), node_size = 100)
    nx.draw_networkx_edges(G, pos)
    plt.axis('off')
    plt.savefig('G_'+str(cascada)+'.png',bbox_inches='tight')

def create_predictive(cascada):
    excel='Total_Dataset_stats.xlsx'
    sheet='TWEETS'
    ex=pandas.read_excel(excel, sheet)
    column0='2_id_tweet'
    column1='3_id_parent'
    column2='0_cascade'
    column3='1_level'
    column4='RELEVANCE'
    values_id=ex[column0].values
    values_par=ex[column1].values
    values_cas=ex[column2].values
    values_lev=ex[column3].values
    values_rel=ex[column4].values
    first_list=[]
    n=0
    for line in values_cas:
      if line==cascada:
        first_list.append([values_id[n],values_par[n],values_lev[n],values_rel[
       n]])
      n=n+1

    if len(first_list)==1:
        return None

    G=nx.DiGraph()
    for line in first_list:
        G.add_node(line[0])
    for line in first_list:
        if line[1]!=str(['-']):
            G.add_edge(line[1],line[0])

    n=0
    color_list=[]
    list_nodes=[]

    for node in G.nodes():
        list_nodes.append(node)

    while n<len(G.nodes()):
        for linia in first_list:
            if linia[0]==list_nodes[n]:
                if linia[3]==1:
                    color_list.append('g')
                elif (linia[3]==0 or linia[3]=='-'):
                    color_list.append('r')
        n=n+1

    pos=nx.spring_layout(G)
    nx.draw_networkx_nodes(G, pos, node_color=color_list, cmap=plt.get_cmap('jet
        '), node_size = 100)
    nx.draw_networkx_edges(G, pos)
    plt.axis('off')
    plt.savefig('G_'+str(cascada)+'.png',bbox_inches='tight')
```

File: qua_ana.py

```
import pandas as pd
import pandas

def qua_ana():
    excel='Qualitative analysis TO PYTHON.xlsx'
    sheet1='TWEETS'
    sheet2='AUX'
    ex1=pandas.read_excel(excel,sheet1)
```

```
ex2=pandas.read_excel(excel,sheet2)

column_text='02_t_text'

column_pre_loc='Prepositions of loc'
column_pre_time='Prepositions of time'
column_adv_time='Adverbs of time and synonims'
column_abs_time='Time absolute'
column_rel_time='Time relative'
column_day_exp='Time of the day expressions'
column_intro='Introducing yourself'

list_text=ex1[column_text].values

list_pre_loc=ex2[column_pre_loc].values
list_pre_time=ex2[column_pre_time].values
list_adv_time=ex2[column_adv_time].values
list_abs_time=ex2[column_abs_time].values
list_rel_time=ex2[column_rel_time].values
list_day_exp=ex2[column_day_exp].values
list_intro=ex2[column_intro].values

res_pre_loc=[] #1
res_cont_verb=[] #2
res_pre_time=[] #3
res_adv_time=[] #4
res_abs_time=[] #5
res_rel_time=[] #6
res_day_exp=[] #7
res_intro=[] #8


for tweet in list_text:

  n=0 #1
  for x in list_pre_loc:
    if (type(x)==str and x in tweet):
      n=n+1
  if n==0:
    res_pre_loc.append(0)
  else:
    res_pre_loc.append(1)
    n=0 #2
    if ('ing ' or 'ING ') in tweet:
            n=1
    if n==0:
        res_cont_verb.append(0)
    else:
        res_cont_verb.append(1)

    n=0 #3
    for x in list_pre_time:
        if (type(x)==str and x in tweet):
            n=n+1
    if n==0:
        res_pre_time.append(0)
    else:
        res_pre_time.append(1)

    n=0 #4
    for x in list_adv_time:
        if (type(x)==str and x in tweet):
            n=n+1
    if n==0:
        res_adv_time.append(0)
    else:
        res_adv_time.append(1)

    n=0 #5
    for x in list_abs_time:
        if (type(x)==str and x in tweet):
            n=n+1
    if n==0:
        res_abs_time.append(0)
    else:
        res_abs_time.append(1)

    n=0 #6
    for x in list_rel_time:
        if (type(x)==str and x in tweet):
            n=n+1
    if n==0:
        res_rel_time.append(0)
    else:
      res_rel_time.append(1)

    n=0 #7
    for x in list_day_exp:
      if (type(x)==str and x in tweet):
        n=n+1
    if n==0:
      res_day_exp.append(0)
    else:
      res_day_exp.append(1)

    n=0 #8
    for x in list_intro:
      if (type(x)==str and x in tweet):
        n=n+1
    if n==0:
      res_intro.append(0)
    else:
      res_intro.append(1)


  df=pd.DataFrame({'1_pre_loc':res_pre_loc,'2_cont_verb':res_cont_verb,'3_pre_
     time':res_pre_time,'4_adv_time':res_adv_time,'5_abs_time':res_abs_time,'6
     _rel_time':res_rel_time,'7_day_exp':res_day_exp,'8_intro':res_intro})

  writer=pd.ExcelWriter('Qualitative analysis FROM PYTHON.xlsx', engine='xlsxw
     riter')
  df.to_excel(writer, sheet_name='TWEETS')
  workbook=writer.book
  workbook.add_format({'bold':True,'valign':'top','fg_color':'#D7E4BC'})
  writer.save()
```

File: inter.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'inter.ui'
#
# Created by: PyQt5 UI code generator 5.9
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets
```

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1114, 733)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.cas_label = QtWidgets.QLabel(self.centralwidget)
        self.cas_label.setGeometry(QtCore.QRect(150, 0, 71, 31))
        self.cas_label.setObjectName("cas_label")
        self.level_label = QtWidgets.QLabel(self.centralwidget)
        self.level_label.setGeometry(QtCore.QRect(290, 0, 51, 31))
        self.level_label.setObjectName("level_label")
        self.max_level_label = QtWidgets.QLabel(self.centralwidget)
        self.max_level_label.setGeometry(QtCore.QRect(380, 0, 161, 31))
        self.max_level_label.setObjectName("max_level_label")
        self.replies_label = QtWidgets.QLabel(self.centralwidget)
        self.replies_label.setGeometry(QtCore.QRect(570, 10, 121, 16))
        self.replies_label.setObjectName("replies_label")
        self.tweet_label = QtWidgets.QLabel(self.centralwidget)
        self.tweet_label.setGeometry(QtCore.QRect(750, 10, 121, 16))
        self.tweet_label.setObjectName("tweet_label")
        self.start_button = QtWidgets.QPushButton(self.centralwidget)
        self.start_button.setGeometry(QtCore.QRect(20, 0, 80, 31))
        self.start_button.setObjectName("start_button")
        self.previous_button = QtWidgets.QPushButton(self.centralwidget)
        self.previous_button.setGeometry(QtCore.QRect(10, 40, 80, 22))
        self.previous_button.setObjectName("previous_button")
        self.next_button = QtWidgets.QPushButton(self.centralwidget)
        self.next_button.setGeometry(QtCore.QRect(150, 40, 80, 22))
        self.next_button.setObjectName("next_button")
        self.tweet_text = QtWidgets.QPlainTextEdit(self.centralwidget)
        self.tweet_text.setGeometry(QtCore.QRect(10, 70, 221, 51))
        font = QtGui.QFont()
        font.setPointSize(6)
        self.tweet_text.setFont(font)
        self.tweet_text.setReadOnly(True)
        self.tweet_text.setObjectName("tweet_text")
        self.photos_label = QtWidgets.QLabel(self.centralwidget)
        self.photos_label.setGeometry(QtCore.QRect(10, 490, 59, 21))
        self.photos_label.setObjectName("photos_label")
        self.web = QtWebEngineWidgets.QWebEngineView(self.centralwidget)
        self.web.setGeometry(QtCore.QRect(250, 30, 861, 661))
        self.web.setUrl(QtCore.QUrl("about:blank"))
        self.web.setObjectName("web")
        self.add_button = QtWidgets.QPushButton(self.centralwidget)
        self.add_button.setGeometry(QtCore.QRect(80, 680, 80, 31))
        self.add_button.setObjectName("add_button")
        self.loc_exp_label = QtWidgets.QLabel(self.centralwidget)
        self.loc_exp_label.setGeometry(QtCore.QRect(50, 260, 241, 20))
        font = QtGui.QFont()
        font.setPointSize(6)
        self.loc_exp_label.setFont(font)
        self.loc_exp_label.setObjectName("loc_exp_label")
        self.cas_text = QtWidgets.QLineEdit(self.centralwidget)
        self.cas_text.setGeometry(QtCore.QRect(220, 0, 61, 31))
        self.cas_text.setReadOnly(True)
        self.cas_text.setObjectName("cas_text")
        self.level_text = QtWidgets.QLineEdit(self.centralwidget)
        self.level_text.setGeometry(QtCore.QRect(340, 0, 31, 31))
        self.level_text.setReadOnly(True)
        self.level_text.setObjectName("level_text")
        self.max_level_text = QtWidgets.QLineEdit(self.centralwidget)
        self.max_level_text.setGeometry(QtCore.QRect(540, 0, 31, 31))
        self.max_level_text.setReadOnly(True)
        self.max_level_text.setObjectName("max_level_text")
        self.replies_text = QtWidgets.QLineEdit(self.centralwidget)
        self.replies_text.setGeometry(QtCore.QRect(700, 0, 41, 31))
        self.replies_text.setReadOnly(True)
        self.replies_text.setObjectName("replies_text")
        self.tweet_number = QtWidgets.QLineEdit(self.centralwidget)
        self.tweet_number.setGeometry(QtCore.QRect(800, 0, 301, 31))
        self.tweet_number.setReadOnly(True)
        self.tweet_number.setObjectName("tweet_number")
        self.loc_box = QtWidgets.QGroupBox(self.centralwidget)
        self.loc_box.setGeometry(QtCore.QRect(10, 180, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 127))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 127))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.loc_box.setPalette(palette)
        self.loc_box.setObjectName("loc_box")
        self.loc_yes_button = QtWidgets.QRadioButton(self.loc_box)
        self.loc_yes_button.setGeometry(QtCore.QRect(10, 20, 61, 20))
        self.loc_yes_button.setObjectName("loc_yes_button")
        self.loc_no_button = QtWidgets.QRadioButton(self.loc_box)
        self.loc_no_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.loc_no_button.setObjectName("loc_no_button")
        self.loc_pre_box = QtWidgets.QGroupBox(self.centralwidget)
        self.loc_pre_box.setGeometry(QtCore.QRect(10, 220, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 127))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 127))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.loc_pre_box.setPalette(palette)
        self.loc_pre_box.setObjectName("loc_pre_box")
        self.loc_pre_1_button = QtWidgets.QRadioButton(self.loc_pre_box)
        self.loc_pre_1_button.setGeometry(QtCore.QRect(10, 20, 41, 20))
        self.loc_pre_1_button.setCheckable(True)
        self.loc_pre_1_button.setChecked(False)
        self.loc_pre_1_button.setObjectName("loc_pre_1_button")
        self.loc_pre_2_button = QtWidgets.QRadioButton(self.loc_pre_box)
        self.loc_pre_2_button.setGeometry(QtCore.QRect(60, 20, 41, 20))
        self.loc_pre_2_button.setObjectName("loc_pre_2_button")
        self.loc_pre_3_button = QtWidgets.QRadioButton(self.loc_pre_box)
        self.loc_pre_3_button.setGeometry(QtCore.QRect(110, 20, 41, 20))
        self.loc_pre_3_button.setObjectName("loc_pre_3_button")
        self.loc_pre_4_button = QtWidgets.QRadioButton(self.loc_pre_box)
        self.loc_pre_4_button.setGeometry(QtCore.QRect(160, 20, 31, 20))
        self.loc_pre_4_button.setObjectName("loc_pre_4_button")
        self.text_box = QtWidgets.QGroupBox(self.centralwidget)
        self.text_box.setGeometry(QtCore.QRect(10, 280, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(170, 170, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(170, 170, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.text_box.setPalette(palette)
        self.text_box.setObjectName("text_box")
        self.text_ot_button = QtWidgets.QRadioButton(self.text_box)
        self.text_ot_button.setGeometry(QtCore.QRect(0, 20, 41, 20))
        self.text_ot_button.setObjectName("text_ot_button")
        self.text_0_button = QtWidgets.QRadioButton(self.text_box)
        self.text_0_button.setGeometry(QtCore.QRect(130, 20, 41, 20))
        self.text_0_button.setObjectName("text_0_button")
        self.text_1_button = QtWidgets.QRadioButton(self.text_box)
        self.text_1_button.setGeometry(QtCore.QRect(190, 20, 41, 20))
        self.text_1_button.setObjectName("text_1_button")
        self.url_box = QtWidgets.QGroupBox(self.centralwidget)
        self.url_box.setGeometry(QtCore.QRect(10, 390, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(170, 255, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(170, 255, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.url_box.setPalette(palette)
        self.url_box.setObjectName("url_box")
        self.url_ot_button = QtWidgets.QRadioButton(self.url_box)
        self.url_ot_button.setGeometry(QtCore.QRect(0, 20, 100, 20))
        self.url_ot_button.setObjectName("url_ot_button")
        self.url_na_button = QtWidgets.QRadioButton(self.url_box)
        self.url_na_button.setGeometry(QtCore.QRect(50, 20, 51, 20))
        self.url_na_button.setObjectName("url_na_button")
        self.url_0_button = QtWidgets.QRadioButton(self.url_box)
        self.url_0_button.setGeometry(QtCore.QRect(130, 20, 51, 20))
        self.url_0_button.setObjectName("url_0_button")
        self.url_1_button = QtWidgets.QRadioButton(self.url_box)
        self.url_1_button.setGeometry(QtCore.QRect(190, 20, 51, 20))
        self.url_1_button.setObjectName("url_1_button")
        self.media_box = QtWidgets.QGroupBox(self.centralwidget)
        self.media_box.setGeometry(QtCore.QRect(10, 450, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.media_box.setPalette(palette)
        self.media_box.setObjectName("media_box")
        self.media_none_button = QtWidgets.QRadioButton(self.media_box)
        self.media_none_button.setGeometry(QtCore.QRect(0, 20, 100, 20))
        self.media_none_button.setObjectName("media_none_button")
        self.media_photo_button = QtWidgets.QRadioButton(self.media_box)
        self.media_photo_button.setGeometry(QtCore.QRect(70, 20, 100, 20))
        self.media_photo_button.setObjectName("media_photo_button")
        self.media_video_button = QtWidgets.QRadioButton(self.media_box)
        self.media_video_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.media_video_button.setObjectName("media_video_button")
        self.media_gif_button = QtWidgets.QRadioButton(self.media_box)
        self.media_gif_button.setGeometry(QtCore.QRect(190, 20, 100, 20))
        self.media_gif_button.setObjectName("media_gif_button")
        self.photos_number_box = QtWidgets.QGroupBox(self.centralwidget)
        self.photos_number_box.setGeometry(QtCore.QRect(10, 510, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.photos_number_box.setPalette(palette)
        self.photos_number_box.setObjectName("photos_number_box")
        self.photos_number_1_button = QtWidgets.QRadioButton(self.photos_number_b
ox)
        self.photos_number_1_button.setGeometry(QtCore.QRect(0, 20, 100, 20))
        self.photos_number_1_button.setObjectName("photos_number_1_button")
        self.photos_number_2_button = QtWidgets.QRadioButton(self.photos_number_b
ox)
        self.photos_number_2_button.setGeometry(QtCore.QRect(60, 20, 100, 20))
        self.photos_number_2_button.setObjectName("photos_number_2_button")
        self.photos_number_3_button = QtWidgets.QRadioButton(self.photos_number_b
ox)
        self.photos_number_3_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.photos_number_3_button.setObjectName("photos_number_3_button")
        self.photos_number_4_button = QtWidgets.QRadioButton(self.photos_number_b
ox)
        self.photos_number_4_button.setGeometry(QtCore.QRect(190, 20, 100, 20))
        self.photos_number_4_button.setObjectName("photos_number_4_button")
        self.photos_box = QtWidgets.QGroupBox(self.centralwidget)
        self.photos_box.setGeometry(QtCore.QRect(10, 550, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.photos_box.setPalette(palette)
        self.photos_box.setObjectName("photos_box")
        self.photos_ot_button = QtWidgets.QRadioButton(self.photos_box)
        self.photos_ot_button.setGeometry(QtCore.QRect(0, 20, 100, 20))
        self.photos_ot_button.setObjectName("photos_ot_button")
        self.photos_0_button = QtWidgets.QRadioButton(self.photos_box)
        self.photos_0_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.photos_0_button.setObjectName("photos_0_button")
        self.photos_1_button = QtWidgets.QRadioButton(self.photos_box)
        self.photos_1_button.setGeometry(QtCore.QRect(190, 20, 100, 20))
        self.photos_1_button.setObjectName("photos_1_button")
        self.video_box = QtWidgets.QGroupBox(self.centralwidget)
        self.video_box.setGeometry(QtCore.QRect(10, 600, 231, 41))
        self.video_box.setObjectName("video_box")
        self.video_ot_button = QtWidgets.QRadioButton(self.video_box)
        self.video_ot_button.setGeometry(QtCore.QRect(0, 20, 100, 20))
        self.video_ot_button.setObjectName("video_ot_button")
        self.video_0_button = QtWidgets.QRadioButton(self.video_box)
        self.video_0_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.video_0_button.setObjectName("video_0_button")
        self.video_1_button = QtWidgets.QRadioButton(self.video_box)
        self.video_1_button.setGeometry(QtCore.QRect(190, 20, 100, 20))
        self.video_1_button.setObjectName("video_1_button")
        self.gif_box = QtWidgets.QGroupBox(self.centralwidget)
        self.gif_box.setGeometry(QtCore.QRect(10, 640, 231, 41))
        self.gif_box.setObjectName("gif_box")
        self.gif_ot_button = QtWidgets.QRadioButton(self.gif_box)
        self.gif_ot_button.setGeometry(QtCore.QRect(0, 20, 100, 20))
        self.gif_ot_button.setObjectName("gif_ot_button")
        self.gif_0_button = QtWidgets.QRadioButton(self.gif_box)
        self.gif_0_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.gif_0_button.setObjectName("gif_0_button")
        self.gif_1_button = QtWidgets.QRadioButton(self.gif_box)
        self.gif_1_button.setGeometry(QtCore.QRect(190, 20, 100, 20))
        self.gif_1_button.setObjectName("gif_1_button")
        self.copy_box = QtWidgets.QGroupBox(self.centralwidget)
        self.copy_box.setGeometry(QtCore.QRect(10, 130, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 170, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 170, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.copy_box.setPalette(palette)
        self.copy_box.setObjectName("copy_box")
        self.copy_no_button = QtWidgets.QRadioButton(self.copy_box)
        self.copy_no_button.setGeometry(QtCore.QRect(10, 20, 100, 20))
        self.copy_no_button.setObjectName("copy_no_button")
        self.copy_yes_button = QtWidgets.QRadioButton(self.copy_box)
        self.copy_yes_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.copy_yes_button.setObjectName("copy_yes_button")
        self.oruser_box = QtWidgets.QGroupBox(self.centralwidget)
        self.oruser_box.setGeometry(QtCore.QRect(10, 330, 231, 41))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(170, 170, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(170, 170, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(239, 235, 231))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        self.oruser_box.setPalette(palette)
        self.oruser_box.setObjectName("oruser_box")
        self.oruser_no_button = QtWidgets.QRadioButton(self.oruser_box)
        self.oruser_no_button.setGeometry(QtCore.QRect(10, 20, 100, 20))
        self.oruser_no_button.setObjectName("oruser_no_button")
        self.oruser_yes_button = QtWidgets.QRadioButton(self.oruser_box)
        self.oruser_yes_button.setGeometry(QtCore.QRect(130, 20, 100, 20))
        self.oruser_yes_button.setObjectName("oruser_yes_button")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Assessment of tweets"
         ))
        self.cas_label.setText(_translate("MainWindow", "CASCADE:"))
        self.level_label.setText(_translate("MainWindow", "LEVEL:"))
        self.max_level_label.setText(_translate("MainWindow", "MAX LEVEL OF CASCA
         DE:"))
        self.replies_label.setText(_translate("MainWindow", "REPLIES TO TWEET:"))

        self.tweet_label.setText(_translate("MainWindow", "TWEET:"))
        self.start_button.setText(_translate("MainWindow", "Start"))
        self.previous_button.setText(_translate("MainWindow", "Previous"))
        self.next_button.setText(_translate("MainWindow", "Next"))
        self.photos_label.setText(_translate("MainWindow", "PHOTOS"))
        self.add_button.setText(_translate("MainWindow", "Add row"))
        self.loc_exp_label.setText(_translate("MainWindow", "1:city 2:zone city 3
         :street 4:detailed"))
        self.loc_box.setTitle(_translate("MainWindow", "Location in text?"))
        self.loc_yes_button.setText(_translate("MainWindow", "YES"))
        self.loc_no_button.setText(_translate("MainWindow", "NO"))
        self.loc_pre_box.setTitle(_translate("MainWindow", "Precision"))
        self.loc_pre_1_button.setText(_translate("MainWindow", "1"))
        self.loc_pre_2_button.setText(_translate("MainWindow", "2"))
        self.loc_pre_3_button.setText(_translate("MainWindow", "3"))
        self.loc_pre_4_button.setText(_translate("MainWindow", "4"))
        self.text_box.setTitle(_translate("MainWindow", "Text"))
        self.text_ot_button.setText(_translate("MainWindow", "OT"))
        self.text_0_button.setText(_translate("MainWindow", "0"))
        self.text_1_button.setText(_translate("MainWindow", "1"))
        self.url_box.setTitle(_translate("MainWindow", "Url"))
        self.url_ot_button.setText(_translate("MainWindow", "OT"))
        self.url_na_button.setText(_translate("MainWindow", "NA"))
        self.url_0_button.setText(_translate("MainWindow", "0"))
        self.url_1_button.setText(_translate("MainWindow", "1"))
        self.media_box.setTitle(_translate("MainWindow", "Media"))
        self.media_none_button.setText(_translate("MainWindow", "None"))
        self.media_photo_button.setText(_translate("MainWindow", "Pho"))
        self.media_video_button.setText(_translate("MainWindow", "Vid"))
        self.media_gif_button.setText(_translate("MainWindow", "GIF"))
        self.photos_number_box.setTitle(_translate("MainWindow", "Number"))
        self.photos_number_1_button.setText(_translate("MainWindow", "1"))
        self.photos_number_2_button.setText(_translate("MainWindow", "2"))
        self.photos_number_3_button.setText(_translate("MainWindow", "3"))
        self.photos_number_4_button.setText(_translate("MainWindow", "4"))
        self.photos_box.setTitle(_translate("MainWindow", "Assessment"))
        self.photos_ot_button.setText(_translate("MainWindow", "OT"))
        self.photos_0_button.setText(_translate("MainWindow", "0"))
        self.photos_1_button.setText(_translate("MainWindow", "1"))
        self.video_box.setTitle(_translate("MainWindow", "VIdeo"))
        self.video_ot_button.setText(_translate("MainWindow", "OT"))
        self.video_0_button.setText(_translate("MainWindow", "0"))
        self.video_1_button.setText(_translate("MainWindow", "1"))
        self.gif_box.setTitle(_translate("MainWindow", "GIF"))
        self.gif_ot_button.setText(_translate("MainWindow", "OT"))
        self.gif_0_button.setText(_translate("MainWindow", "0"))
        self.gif_1_button.setText(_translate("MainWindow", "1"))
        self.copy_box.setTitle(_translate("MainWindow", "Tweet is a copy / retwee
         t"))
        self.copy_no_button.setText(_translate("MainWindow", "NO"))
        self.copy_yes_button.setText(_translate("MainWindow", "YES"))
        self.oruser_box.setTitle(_translate("MainWindow", "Original user?"))
        self.oruser_no_button.setText(_translate("MainWindow", "NO"))
        self.oruser_yes_button.setText(_translate("MainWindow", "YES"))


        #LINK BUTTONS - FUNCTIONS
        self.media_none_button.clicked.connect(lambda:act().enable_panels())
        self.media_photo_button.clicked.connect(lambda:act().enable_panels())
        self.media_video_button.clicked.connect(lambda:act().enable_panels())
        self.media_gif_button.clicked.connect(lambda:act().enable_panels())
        self.text_ot_button.clicked.connect(lambda:act().enable_panels())
        self.text_1_button.clicked.connect(lambda:act().enable_panels())
        self.text_0_button.clicked.connect(lambda:act().enable_panels())
        self.copy_no_button.clicked.connect(lambda:act().enable_panels())
        self.copy_yes_button.clicked.connect(lambda:act().enable_panels())
        self.loc_yes_button.clicked.connect(lambda:act().enable_panels())
        self.loc_no_button.clicked.connect(lambda:act().enable_panels())

        self.add_button.clicked.connect(lambda:act().add())
        self.start_button.clicked.connect(lambda:act().start())
        self.next_button.clicked.connect(lambda:act().next())
        self.previous_button.clicked.connect(lambda:act().previous())
        #self.write_button.clicked.connect(lambda:act().write())


from PyQt5 import QtWebEngineWidgets

import pandas
import pandas as pd
from pandas import ExcelWriter


class act(object):
    def __init__(self):
        self.df1=pandas.read_excel('tree.xls','TWEETS')
        self.df2=pandas.read_excel('DataTweets.xls','DATA')

    def start(self):

        ui.cas_text.setText(str(self.df1.values[0][0]))
        ui.level_text.setText(str(self.df1.values[0][1]))
        self.num_replies=str(self.df1.values[0][4]).split(',')
        if len(self.num_replies)==1:
            if '-' in self.num_replies:
                self.num_replies='0'
            else:
                self.num_replies='1'
        else:
            self.num_replies=str(len(self.num_replies))
        ui.replies_text.setText(self.num_replies)
        ui.tweet_number.setText(self.df1.values[0][2])
        ui.tweet_text.setPlainText(self.df2.values[0][2])
        ui.web.setUrl(QtCore.QUrl(self.df2.values[0][1]))

    def farray(self):
        now=str(ui.tweet_number.text())
        i=0
        for x in self.df1.values:  #do this with index of df1
            if now==str(x[2]):
                array=i
            i=i+1
        return array

    def next(self):
        array=self.farray()
        finish=len(self.df1)-1
        if array==finish:
            self.start()
        else:
            ui.cas_text.setText(str(self.df1.values[array+1][0]))
            ui.level_text.setText(str(self.df1.values[array+1][1]))
            self.num_replies=str(self.df1.values[array+1][4]).split(',')
            if len(self.num_replies)==1:
                if '-' in self.num_replies:
                    self.num_replies='0'
                else:
                    self.num_replies='1'
            else:
                self.num_replies=str(len(self.num_replies))
            ui.replies_text.setText(self.num_replies)
            ui.tweet_number.setText(self.df1.values[array+1][2])
        ui.tweet_text.setPlainText(self.df2.values[array+1][2])
        ui.web.setUrl(QtCore.QUrl(self.df2.values[array+1][1]))

  def previous(self):
     array=self.farray()
     ui.cas_text.setText(str(self.df1.values[array-1][0]))
     ui.level_text.setText(str(self.df1.values[array-1][1]))
     self.num_replies=str(self.df1.values[array-1][4]).split(',')
     if len(self.num_replies)==1:
        if '-' in self.num_replies:
           self.num_replies='0'
        else:
           self.num_replies='1'
     else:
        self.num_replies=str(len(self.num_replies))
     ui.replies_text.setText(self.num_replies)
     ui.tweet_number.setText(self.df1.values[array-1][2])
     ui.tweet_text.setPlainText(self.df2.values[array-1][2])
     ui.web.setUrl(QtCore.QUrl(self.df2.values[array-1][1]))


  def add(self):
     if ui.text_ot_button.isChecked():
        self.add_ex(0)

     elif ui.media_none_button.isChecked():
        self.add_ex(1)

     elif ui.media_photo_button.isChecked():
        self.add_ex(2)

     elif ui.media_video_button.isChecked():
        self.add_ex(3)

     elif ui.media_gif_button.isChecked():
        self.add_ex(4)

     self.next()

  def add_ex(self,number):
     df3=pd.read_excel('Assessment.xls','TWEETS')
     num_tweet=str(ui.tweet_number.text())
     if number==0:

        iscopy='-'
        isloc='-'
        locpre='-'
        text='OT'
        oruser='-'
        url='-'
        media='-'
        num_photos='-'
        ass_photos='-'
        video='-'
        gif='-'

        row=pd.DataFrame({'00_Tweet':[num_tweet],'01_IsCopy':[iscopy],'02_IsLoc
      ':[isloc],'03_LocPre':[locpre],'04_Text':[text],'05_OrUser':[oruser],'06_
      Url':[url],'07_MediaType':[media],'08_PhotosNumber':[num_photos],'09_Phot
      osAssess':[ass_photos],'10_Video':[video],'11_GIF':[gif]})
        df3=df3.append(row,ignore_index=True)

     elif number==1:
        if ui.copy_yes_button.isChecked()==True:
           iscopy='True'
        elif ui.copy_no_button.isChecked()==True:
           iscopy='False'

        if ui.oruser_no_button.isChecked()==True:
           oruser='False'
        elif ui.oruser_yes_button.isChecked()==True:
           oruser='True'

        if ui.loc_yes_button.isChecked()==True:
           isloc='True'
        elif ui.loc_no_button.isChecked()==True:
           isloc='False'
        else:
           isloc='Not done'

        if ui.loc_pre_1_button.isChecked()==True:
           locpre='1'
        elif ui.loc_pre_2_button.isChecked()==True:
           locpre='2'
        elif ui.loc_pre_3_button.isChecked()==True:
            locpre='3'
        elif ui.loc_pre_4_button.isChecked()==True:
            locpre='4'

        if ui.text_0_button.isChecked()==True:
            text='0'
        elif ui.text_1_button.isChecked()==True:
            text='1'

        if ui.url_ot_button.isChecked()==True:
            url='OT'
        elif ui.url_na_button.isChecked()==True:
            url='NA'
        elif ui.url_1_button.isChecked()==True:
            url='1'
        elif ui.url_0_button.isChecked()==True:
            url='0'

        media='None'
        num_photos='-'
        ass_photos='-'
        video='-'
        gif='-'

        row=pd.DataFrame({'00_Tweet':[num_tweet],'01_IsCopy':[iscopy],'02_IsLoc
      ':[isloc],'03_LocPre':[locpre],'04_Text':[text],'05_OrUser':[oruser],'06_
      Url':[url],'07_MediaType':[media],'08_PhotosNumber':[num_photos],'09_Phot
      osAssess':[ass_photos],'10_Video':[video],'11_GIF':[gif]})
        df3=df3.append(row,ignore_index=True)

    elif number==2:
        if ui.copy_yes_button.isChecked()==True:
            iscopy='True'
        elif ui.copy_no_button.isChecked()==True:
            iscopy='False'

        if ui.oruser_no_button.isChecked()==True:
            oruser='False'
        elif ui.oruser_yes_button.isChecked()==True:
            oruser='True'

        if ui.loc_yes_button.isChecked()==True:
            isloc='True'
        elif ui.loc_no_button.isChecked()==True:
            isloc='False'
        else:
            isloc='Not done'

        if ui.loc_pre_1_button.isChecked()==True:
            locpre='1'
        elif ui.loc_pre_2_button.isChecked()==True:
            locpre='2'
        elif ui.loc_pre_3_button.isChecked()==True:
            locpre='3'
        elif ui.loc_pre_4_button.isChecked()==True:
            locpre='4'

        if ui.text_0_button.isChecked()==True:
            text='0'
        elif ui.text_1_button.isChecked()==True:
            text='1'

        if ui.url_ot_button.isChecked()==True:
            url='OT'
        elif ui.url_na_button.isChecked()==True:
            url='NA'
        elif ui.url_1_button.isChecked()==True:
            url='1'
        elif ui.url_0_button.isChecked()==True:
            url='0'

        media='photos'

        if ui.photos_number_1_button.isChecked()==True:
            num_photos='1'
        elif ui.photos_number_2_button.isChecked()==True:
            num_photos='2'
        elif ui.photos_number_3_button.isChecked()==True:
            num_photos='3'
        elif ui.photos_number_4_button.isChecked()==True:
            num_photos='4'

        if ui.photos_ot_button.isChecked()==True:
            ass_photos='OT'
        elif ui.photos_1_button.isChecked()==True:
            ass_photos='1'
        elif ui.photos_0_button.isChecked()==True:
            ass_photos='0'

        video='-'
        gif='-'

        row=pd.DataFrame({'00_Tweet':[num_tweet],'01_IsCopy':[iscopy],'02_IsLoc
      ':[isloc],'03_LocPre':[locpre],'04_Text':[text],'05_OrUser':[oruser],'06_
      Url':[url],'07_MediaType':[media],'08_PhotosNumber':[num_photos],'09_Phot
      osAssess':[ass_photos],'10_Video':[video],'11_GIF':[gif]})
        df3=df3.append(row,ignore_index=True)

    elif number==3:
        if ui.copy_yes_button.isChecked()==True:
            iscopy='True'
        elif ui.copy_no_button.isChecked()==True:
            iscopy='False'

        if ui.oruser_no_button.isChecked()==True:
            oruser='False'
        elif ui.oruser_yes_button.isChecked()==True:
            oruser='True'

        if ui.loc_yes_button.isChecked()==True:
            isloc='True'
        elif ui.loc_no_button.isChecked()==True:
            isloc='False'
        else:
            isloc='Not done'

        if ui.loc_pre_1_button.isChecked()==True:
            locpre='1'
        elif ui.loc_pre_2_button.isChecked()==True:
            locpre='2'
        elif ui.loc_pre_3_button.isChecked()==True:
            locpre='3'
        elif ui.loc_pre_4_button.isChecked()==True:
            locpre='4'

        if ui.text_0_button.isChecked()==True:
            text='0'
        elif ui.text_1_button.isChecked()==True:
            text='1'

        if ui.url_ot_button.isChecked()==True:
            url='OT'
        elif ui.url_na_button.isChecked()==True:
            url='NA'
        elif ui.url_1_button.isChecked()==True:
            url='1'
        elif ui.url_0_button.isChecked()==True:
            url='0'

        media='video'
        num_photos='-'
        ass_photos='-'

        if ui.video_ot_button.isChecked()==True:
            video='OT'
        elif ui.video_1_button.isChecked()==True:
            video='1'
        elif ui.video_0_button.isChecked()==True:
            video='0'

        gif='-'

        row=pd.DataFrame({'00_Tweet':[num_tweet],'01_IsCopy':[iscopy],'02_IsLoc':[isloc],'03_LocPre':[locpre],'04_Text':[text],'05_OrUser':[oruser],'06_Url':[url],'07_MediaType':[media],'08_PhotosNumber':[num_photos],'09_PhotosAssess':[ass_photos],'10_Video':[video],'11_GIF':[gif]})
        df3=df3.append(row,ignore_index=True)

    elif number==4:
        if ui.copy_yes_button.isChecked()==True:
            iscopy='True'
        elif ui.copy_no_button.isChecked()==True:
            iscopy='False'

        if ui.oruser_no_button.isChecked()==True:
            oruser='False'
        elif ui.oruser_yes_button.isChecked()==True:
            oruser='True'

        if ui.loc_yes_button.isChecked()==True:
```

```
          isloc='True'
        elif ui.loc_no_button.isChecked()==True:
          isloc='False'
        else:
          isloc='Not done'

        if ui.loc_pre_1_button.isChecked()==True:
          locpre='1'
        elif ui.loc_pre_2_button.isChecked()==True:
          locpre='2'
        elif ui.loc_pre_3_button.isChecked()==True:
          locpre='3'
        elif ui.loc_pre_4_button.isChecked()==True:
          locpre='4'

        if ui.text_0_button.isChecked()==True:
          text='0'
        elif ui.text_1_button.isChecked()==True:
          text='1'

        if ui.url_ot_button.isChecked()==True:
          url='OT'
        elif ui.url_na_button.isChecked()==True:
          url='NA'
        elif ui.url_1_button.isChecked()==True:
          url='1'
        elif ui.url_0_button.isChecked()==True:
          url='0'

        media='gif'
        num_photos='-'
        ass_photos='-'
        video='-'

        if ui.gif_ot_button.isChecked()==True:
          gif='OT'
        elif ui.gif_1_button.isChecked()==True:
          gif='1'
        elif ui.gif_0_button.isChecked()==True:
          gif='0'
```

```
            row=pd.DataFrame({'00_Tweet':[num_tweet],'01_IsCopy':[iscopy],'02_IsLoc':[isloc],'03_LocPre':[locpre],'04_Text':[text],'05_OrUser':[oruser],'06_Url':[url],'07_MediaType':[media],'08_PhotosNumber':[num_photos],'09_PhotosAssess':[ass_photos],'10_Video':[video],'11_GIF':[gif]})
            df3=df3.append(row,ignore_index=True)

        writer=pd.ExcelWriter('Assessment.xls',engine='xlsxwriter')
        df3.to_excel(writer, sheet_name='TWEETS')
        writer.save()



    def write(self):
        writer=pd.ExcelWriter('Assessment.xls',engine='xlsxwriter')
        self.df3.to_excel(writer, sheet_name='TWEETS')
        writer.save()

    def enable_panels(self):
        if ui.media_none_button.isChecked():
            ui.photos_number_box.setEnabled(False)
            ui.photos_box.setEnabled(False)
            ui.video_box.setEnabled(False)
            ui.gif_box.setEnabled(False)

        elif ui.media_photo_button.isChecked():
            ui.photos_number_box.setEnabled(True)
            ui.photos_box.setEnabled(True)
            ui.video_box.setEnabled(False)
            ui.gif_box.setEnabled(False)

        elif ui.media_video_button.isChecked():
            ui.photos_number_box.setEnabled(False)
            ui.photos_box.setEnabled(False)
            ui.video_box.setEnabled(True)
            ui.gif_box.setEnabled(False)

        elif ui.media_gif_button.isChecked():
            ui.photos_number_box.setEnabled(False)
            ui.photos_box.setEnabled(False)
            ui.video_box.setEnabled(False)
            ui.gif_box.setEnabled(True)
```

```
     elif ui.text_ot_button.isChecked():
        ui.media_box.setEnabled(False)
        ui.copy_box.setEnabled(False)
        ui.oruser_box.setEnabled(False)
        ui.loc_box.setEnabled(False)
        ui.loc_pre_box.setEnabled(False)
        ui.url_box.setEnabled(False)

     elif ui.text_0_button.isChecked():
        ui.media_box.setEnabled(True)
        ui.copy_box.setEnabled(True)
        ui.oruser_box.setEnabled(True)
        ui.loc_box.setEnabled(True)
        ui.loc_pre_box.setEnabled(True)
        ui.url_box.setEnabled(True)

     elif ui.text_0_button.isChecked():
        ui.media_box.setEnabled(True)
        ui.copy_box.setEnabled(True)
        ui.oruser_box.setEnabled(True)
        ui.loc_box.setEnabled(False)
        ui.loc_pre_box.setEnabled(False)
        ui.url_box.setEnabled(True)


from PyQt5 import QtWebEngineWidgets
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QMainWindow

if __name__ == "__main__":
   import sys
   app = QtWidgets.QApplication(sys.argv)
   MainWindow = QtWidgets.QMainWindow()
   ui = Ui_MainWindow()
   ui.setupUi(MainWindow)
   MainWindow.show()
   sys.exit(app.exec_())
```

b. Files and functions to run the tool

File: parent.py / Function: ex_parents

```
from bs4 import BeautifulSoup
import requests
from selenium import webdriver
from selenium.common.exceptions import StaleElementReferenceException, Timeout
     Exception
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd
import pandas
from openpyxl import load_workbook
import requests
import urllib.request

def ex_parents():
   excel_file='a.xls'
   sheet='TWEETS'
   ex=pandas.read_excel(excel_file, sheet)
   column='id'
   values=ex[column].values
   tweet_list=[]
   df=pd.DataFrame(columns=['0_num','1_par_tweet','2_url_tweet','3_n_tweets'])

   for tweet in values:
      str_tweet=[str(tweet)]
      tweet_list.append(str_tweet)
   del(tweet_list[0]) #anar en compte
   i=0
   parent_list=[]
   for tweet in tweet_list:
      par_tweet=prova_par(tweet[0])
      url='https://twitter.com/statuses/'+par_tweet[0]
      try:
         req=urllib.request.urlopen(url)
         available=req.geturl()
      except:
         available='Page not exist'
      if par_tweet[0] in available:
         if par_tweet[0] not in parent_list:
            row=pd.DataFrame({'0_num':[i],'1_par_tweet':[[par_tweet[0]]],'2_url_t
     weet':[url],'3_n_tweets':[par_tweet[1]]})
            df=df.append(row,ignore_index=True)
            i=i+1
            parent_list.append(par_tweet[0])
      else:
         pass

   writer=pd.ExcelWriter('parents_tool.xls',engine='xlsxwriter')
   df.to_excel(writer, sheet_name='TWEETS')
   workbook=writer.book
   workbook.add_format({'bold':True,'valign':'top','fg_color':'#D7E4BC'})
   worksheet=writer.sheets['TWEETS']
   worksheet.set_column('B:B',60,None)
   writer.save()
```

File: parent.py / Function: prova_par

```
def prova_par(tweet_id):
   page=requests.get('http://twitter.com/statuses/'+tweet_id)
    soup=BeautifulSoup(page.content, 'html.parser')
    parent=soup.find_all('div',attrs={'data-replied-tweet-id':True})
    direct_par=True
    partial_id=tweet_id
    quote_tweet=False
    while len(parent)>0:
        direct_par=False
        page=requests.get('http://twitter.com/statuses/'+(str(parent[0]['data-
replied-tweet-id'])))
        partial_id=str(parent[0]['data-replied-tweet-id'])
        ask_n_tweets=soup.find_all('div',attrs={'data-tweet-id':True})
        ask_n_tweets=len(ask_n_tweets)
        soup=BeautifulSoup(page.content, 'html.parser')
        parent=soup.find_all('div',attrs={'data-replied-tweet-id':True})
    if direct_par==True:
        ask_n_tweets=soup.find_all('div',attrs={'data-tweet-id':True})
        ask_n_tweets=len(ask_n_tweets)
    level0_par=partial_id
    return level0_par,ask_n_tweets
```

File: cascades.py / Function: ex_cascades

```
from bs4 import BeautifulSoup
import requests
from selenium import webdriver
from selenium.common.exceptions import StaleElementReferenceException, Timeout
     Exception
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd
import pandas
from openpyxl import load_workbook
import requests
import urllib.request


class wait_for_more_than_n_elements_to_be_present(object):
   def __init__(self, locator, count):
      self.locator = locator
      self.count = count

   def __call__(self, driver):
      try:
         elements = EC._find_elements(driver, self.locator)
         return len(elements) > self.count
      except StaleElementReferenceException:
         return False

def ex_cascades(ck,cs,at,ats,uk):
   excel_file='parents_tool_2.xlsx'
   sheet='TWEETS'
   ex=pandas.read_excel(excel_file, sheet)
   column0='1_par_tweet'
   column1='2_url_tweet'
   column2='3_n_tweets'
   values0=ex[column0].values
   values1=ex[column1].values
   values2=ex[column2].values
   list_parents=[]
   final_list=[]
   for tweet in values0:
      str_tweet=tweet[2:][:-2]
      list_parents.append(str_tweet)

   i=0
   while i<len(list_parents):
      tw_cas=ordenar(list_parents[i],values2[i],ck,cs,at,ats)
      for tw in tw_cas:
         final_list.append(tw)
```

```
        i=i+1

    df=pd.DataFrame(columns=['0_level','1_id','2_text','3_mediatype','4_oruser',
       '5_copy','6_chain'])
    for tw in final_list:
        row=pd.DataFrame({'0_level':[tw[0]],'1_id':[tw[1]],'2_text':[tw[2]],'3_me
        diatype':[tw[3]],'4_oruser':[tw[4]],'5_copy':[tw[5]],'6_chain':[tw[6]]})

        df=df.append(row,ignore_index=True)

    writer=pd.ExcelWriter('auxiliar.xlsx',engine='openpyxl')
    df.to_excel(writer, sheet_name='TWEETS', startrow=0)
    writer.save()

    c=classifier(uk)
    return None
```

File: cascades.py / Function: ordenar

```
def ordenar(parent_tweet,num,ck,cs,at,ats):
  tweet_casc_api=[]

    if num==1:
        list_ids=[parent_tweet]
    else:
        list_ids=id_parsing(parent_tweet)

    for tweet in list_ids:
        api_info=get_tw(tweet,ck,cs,at,ats)
        if api_info==None:
            pass
        else:
            tweet_casc_api.append(api_info)

    L0_tweets=[]
    L1_tweets=[]
    L2_tweets=[]
    L3_tweets=[]

    L0_chain=[]
    L1_chain=[]
    L2_chain=[]
    L3_chain=[]
```

```
for tweet in tweet_casc_api:
   if tweet[0]==parent_tweet:
      tweet[1]='None'
      L0_chain.append([tweet[0],'-','-','-'])
      L0_tweets.append(tweet[0])

for tweet in tweet_casc_api:
   if tweet[1] in L0_tweets:
      L1_tweets.append(tweet[0])

      for chain in L0_chain:
         if tweet[1] in chain[0]:
            L1_chain.append([chain[0],tweet[0],'-','-'])

for tweet in tweet_casc_api:
   if tweet[1] in L1_tweets:
      L2_tweets.append(tweet[0])

      for chain in L1_chain:
         if tweet[1] in chain[1]:
            L2_chain.append([chain[0],chain[1],tweet[0],'-'])

for tweet in tweet_casc_api:
   if tweet[1] in L2_tweets:
      L3_tweets.append(tweet[0])

      for chain in L2_chain:
         if tweet[1] in chain[2]:
            L3_chain.append([chain[0],chain[1],chain[2],tweet[0]])

for tweet in tweet_casc_api:
   if tweet[0] in L0_tweets:
      original_text=tweet[2]
      original_user_id=tweet[4]

copied=[]
or_user=[]
for tweet in tweet_casc_api:
   if tweet[0] in L0_tweets:
      copied.append(0)
      or_user.append(0)
```

```
  else:
    #if tweet[2] in original_text:
    #    copied.append(1)
    #else:
    #    copied.append(0)
    if ('“@' in tweet[2]) or ('RT' in tweet[2]) or (' via ' in tweet[2]) or (tweet[2] in original_text):
      copied.append(1)
    else:
      copied.append(0)

    if tweet[4]==original_user_id:
      or_user.append(1)
    else:
      or_user.append(0)


i=0
final_list=[]
while i<len(tweet_casc_api):
  if tweet_casc_api[i][0] in L0_tweets:
    for chain in L0_chain:
      if tweet_casc_api[i][0]==chain[0]:
        seq=chain
    final_list.append([0,tweet_casc_api[i][0],tweet_casc_api[i][2],tweet_casc_api[i][3],or_user[i],copied[i],seq])

  elif tweet_casc_api[i][0] in L1_tweets:
    for chain in L1_chain:
      if tweet_casc_api[i][0]==chain[1]:
        seq=chain
    final_list.append([1,tweet_casc_api[i][0],tweet_casc_api[i][2],tweet_casc_api[i][3],or_user[i],copied[i],seq])

  elif tweet_casc_api[i][0] in L2_tweets:
    for chain in L2_chain:
      if tweet_casc_api[i][0]==chain[2]:
        seq=chain
    final_list.append([2,tweet_casc_api[i][0],tweet_casc_api[i][2],tweet_casc_api[i][3],or_user[i],copied[i],seq])

  elif tweet_casc_api[i][0] in L3_tweets:
    for chain in L3_chain:
            if tweet_casc_api[i][0]==chain[3]:
               seq=chain
         final_list.append([3,tweet_casc_api[i][0],tweet_casc_api[i][2],tweet_ca
   sc_api[i][3],or_user[i],copied[i],seq])

      i=i+1

   return final_list
```

File: cascades.py / Function: id_parsing

```
def id_parsing(L0_tweet):
  l_collected=[]
   l_already_parsed=[]
   l_to_be_parsed=[]
   l_collected.append(L0_tweet)
   l_to_be_parsed.append(L0_tweet)
   ids_obtained=prova_nova(L0_tweet)
   l_already_parsed.append(L0_tweet)
   l_to_be_parsed=[x for x in l_to_be_parsed if x!=L0_tweet]

   for id in ids_obtained:
      for id in ids_obtained:
         if id in l_collected:
            pass
         else:
            l_collected.append(str(id))

         if id in l_already_parsed:
            pass
         else:
            l_to_be_parsed.append(str(id))
            l_to_be_parsed=list(set(l_to_be_parsed))

   while len(l_to_be_parsed)>0:
      for id1 in l_to_be_parsed:
         ids_obtained=prova_nova(id1)
         l_already_parsed.append(id1)
         l_to_be_parsed=[x for x in l_to_be_parsed if x!=id1]

         for id2 in ids_obtained:
            if id2 in l_collected:
                pass
            else:
                l_collected.append(str(id2))

                if id2 in l_already_parsed:
                    pass
                else:
                    l_to_be_parsed.append(str(id2))
                    l_to_be_parsed=list(set(l_to_be_parsed))


    return l_collected
```

File: cascades.py / Function: prova_nova

```
def prova_nova(tweet_id):
     url = 'http://twitter.com/statuses/'+tweet_id
    try:
        req=urllib.request.urlopen(url)
        available=req.geturl()
    except:
        available='Page not exist'

    if tweet_id not in available:
        l=[]
        return l

    driver = webdriver.Chrome()
    try:
        driver.maximize_window()
    except:
        l=[]
        return l
    try:
        driver.get(url)
    except:
        l=[]
        return l

    # initial wait for the tweets to load
    try:
        wait = WebDriverWait(driver, 10)
```

```
      wait.until(EC.visibility_of_element_located((By.CSS_SELECTOR, "li[data-
       item-id]")))
    except:
      l=[]
      return l

    # scroll down to the last tweet until there is no more tweets loaded
    while True:
      tweets = driver.find_elements_by_css_selector("li[data-item-id]")
      number_of_tweets = len(tweets)

      driver.execute_script("arguments[0].scrollIntoView();", tweets[-1])

      try:
        wait.until(wait_for_more_than_n_elements_to_be_present((By.CSS_SELECTOR
       , "li[data-item-id]"), number_of_tweets))

      except TimeoutException:
        break

    page_source = driver.page_source
    driver.close()
    soup = BeautifulSoup(page_source,'lxml')
    reply_list_1=soup.find_all('div',attrs={'data-tweet-id':True,'data-has-
       parent-tweet':True})
    ids_list=[]
    for tweet in reply_list_1:
      id=str(tweet['data-tweet-id'])
      ids_list.append(id)

    return ids_list
```

File: cascades.py / Function: get_tw

```
import tweepy
import json
import pandas as pd
import pandas

def get_tw(num,ck,cs,at,ats):

    auth = tweepy.OAuthHandler(ck, cs)
    auth.set_access_token(at, ats)
```

```

api = tweepy.API(auth)


try:
  info=api.get_status(num)
except:
  return None
j=info._json
dictlist=[]
for key,value in j.items():
  linia=[key,value]
  dictlist.append(linia)

obt_l_t=['text','in_reply_to_status_id_str','entities','user']
obt_l_u=['id_str']
obt_l_e1=['media']
obt_l_e2=['type']
f_list=[]
userlist=[]
medialist1=[]
medialist2=[]
media='None'

#df=pd.DataFrame(columns=['00_t_id','01_t_url','02_t_text','03_t_truncated',
   '04_t_retweets','05_t_likes','06_t_date','07_t_geo','08_t_place','09_t_co
   or','10_t_lang','11_t_source','12_u_id','13_u_tname','14_u_name','15_u_ve
   rified','16_u_geo','17_u_place','18_u_description','19_u_followers','20_u
   _following','21_u_posts'])

for line in dictlist:
  if line[0] in obt_l_t:
    n=obt_l_t.index(line[0])

    if n==0:
      text=str(line[1])
    elif n==1:
      id_parent=str(line[1])

    elif n==2:
      for key,value in line[1].items():
        linia=[key,value]
        medialist1.append(linia)
```

|

```
        for x in medialist1:
          if x[0] in obt_l_e1:
            for key,value in x[1][0].items():
              linia=[key,value]
              medialist2.append(linia)
            for z in medialist2:
              if z[0] in obt_l_e2:
                media=str(z[1])

      elif n==3:
        for key,value in line[1].items():
          linia=[key,value]
          userlist.append(linia)
        for x in userlist:
          if x[0] in obt_l_u:
            id_user=str(x[1])

  l=[num,id_parent,text,media,id_user]
  return l
```

File: cascades.py / Function: classifier

```
def classifier(uk):
   excel_file='auxiliar.xlsx'
  sheet='TWEETS'
  ex=pandas.read_excel(excel_file, sheet)
  column0='0_level'
  column1='1_id'
  column2='2_text'
  column3='3_mediatype'
  column4='4_oruser'
  column5='5_copy'
  column6='6_chain'
  values0=ex[column0].values
  values1=ex[column1].values
  values2=ex[column2].values
  values3=ex[column3].values
  values4=ex[column4].values
  values5=ex[column5].values
  values6=ex[column6].values

  list=[]
```

```
discarded_list=[]
i=0
while i<len(values0):
   rel_text=False
   if (int(values0[i])!=0) and (int(values5[i])==0):
      qm=question_mark(values2[i])
      location=get_loc(values2[i],uk)
      ou=int(values4[i])
      uni=unigrams(values2[i])
      media=values3[i]
      chain=values6[i]
      chain=chain.replace('[','')
      chain=chain.replace(']','')
      chain=chain.split(',')

      if qm==0:
         if location==True:
            if media=='None':
               rel_text=True
               list.append([chain,'text'])
            else:
               rel_text=True
               list.append([chain,'text and '+media])
         else:
            if ou==1:
               if uni==0:
                  rel_text=True
                  if media=='None':
                     list.append([chain,'text'])
                  else:
                     list.append([chain,'text and '+media])
               else:
                  discarded_list.append(values1[i])
            else:
               discarded_list.append(values1[i])
      else:
         discarded_list.append(values1[i])

      if (rel_text==False) and (media!='None'):
         list.append([chain,media])
   i=i+1
```

```

    df=pd.DataFrame(columns=['0_L0','1_L1','2_L2','3_L3','4_TYPE'])
    for tweet in list:
        row=pd.DataFrame({'0_L0':[tweet[0][0]],'1_L1':[tweet[0][1]],'2_L2':[tweet
         [0][2]],'3_L3':[tweet[0][3]],'4_TYPE':[tweet[1]]})
        df=df.append(row,ignore_index=True)

    writer=pd.ExcelWriter('Relevant tweets.xlsx',engine='openpyxl')
    df.to_excel(writer, sheet_name='TWEETS', startrow=0)
    writer.save()

    df2=pd.DataFrame(columns=['0_id'])
    for tweet in discarded_list:
        row2=pd.DataFrame({'0_id':[[str(tweet)]]})
        df2=df2.append(row2,ignore_index=True)

    writer_2=pd.ExcelWriter('Discarded tweets.xlsx',engine='openpyxl')
    df2.to_excel(writer_2, sheet_name='TWEETS', startrow=0)
    writer_2.save()

def question_mark(tweet):
    if '?' in tweet:
        return 1
    else:
        return 0

def unigrams(tweet):
    set_unigrams=[' your ','Your ',' YOUR ',' thank','Thank','THANK','(',')','!'
        ]
    for uni in set_unigrams:
        if uni in tweet:
            return 1
    return 0

import json
from rosette.api import API, DocumentParameters, RosetteException

def get_loc(text,uk):

    api=API(user_key=uk)
    params=DocumentParameters()
    params["content"]=text
```

```
  params["genre"]="social-media"
  try:
    info=api.entities(params)
  except RosetteException as exception:
    return False

  dictlist=[]
  for key,value in info.items():
    linia=[key,value]
    dictlist.append(linia)

  string=str(dictlist)
  location=False
  if "'type': 'LOCATION'" in string:
    location=True
  return location
```

File: tool.py

```
# -*- coding: utf-8 -*-
# Form implementation generated from reading ui file 'tool.ui'
#
# Created by: PyQt5 UI code generator 5.9
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
  def setupUi(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(800, 600)
    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")
    self.Widget = QtWidgets.QTabWidget(self.centralwidget)
    self.Widget.setGeometry(QtCore.QRect(0, 0, 801, 591))
    self.Widget.setObjectName("Widget")
    self.Parents_wid = QtWidgets.QWidget()
    self.Parents_wid.setObjectName("Parents_wid")
    self.Explanation_label = QtWidgets.QLabel(self.Parents_wid)
    self.Explanation_label.setGeometry(QtCore.QRect(20, 230, 831, 31))
    self.Explanation_label.setObjectName("Explanation_label")
    self.Start_parents_button = QtWidgets.QPushButton(self.Parents_wid)
    self.Start_parents_button.setGeometry(QtCore.QRect(330, 480, 91, 41))
```

```
        self.Start_parents_button.setObjectName("Start_parents_button")
        self.label = QtWidgets.QLabel(self.Parents_wid)
        self.label.setGeometry(QtCore.QRect(330, 90, 241, 16))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.Parents_wid)
        self.label_2.setGeometry(QtCore.QRect(20, 120, 761, 20))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.Parents_wid)
        self.label_3.setGeometry(QtCore.QRect(50, 140, 581, 16))
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.Parents_wid)
        self.label_4.setGeometry(QtCore.QRect(50, 160, 261, 16))
        self.label_4.setObjectName("label_4")
        self.label_5 = QtWidgets.QLabel(self.Parents_wid)
        self.label_5.setGeometry(QtCore.QRect(50, 180, 301, 16))
        self.label_5.setObjectName("label_5")
        self.label_7 = QtWidgets.QLabel(self.Parents_wid)
        self.label_7.setGeometry(QtCore.QRect(20, 210, 761, 20))
        self.label_7.setObjectName("label_7")
        self.label_6 = QtWidgets.QLabel(self.Parents_wid)
        self.label_6.setGeometry(QtCore.QRect(350, 270, 151, 16))
        self.label_6.setObjectName("label_6")
        self.label_8 = QtWidgets.QLabel(self.Parents_wid)
        self.label_8.setGeometry(QtCore.QRect(280, 300, 241, 20))
        self.label_8.setObjectName("label_8")
        self.label_9 = QtWidgets.QLabel(self.Parents_wid)
        self.label_9.setGeometry(QtCore.QRect(280, 320, 301, 20))
        self.label_9.setObjectName("label_9")
        self.label_10 = QtWidgets.QLabel(self.Parents_wid)
        self.label_10.setGeometry(QtCore.QRect(280, 340, 351, 20))
        self.label_10.setObjectName("label_10")
        self.label_11 = QtWidgets.QLabel(self.Parents_wid)
        self.label_11.setGeometry(QtCore.QRect(280, 360, 371, 20))
        self.label_11.setObjectName("label_11")
        self.Widget.addTab(self.Parents_wid, "")
        self.Manual_wid = QtWidgets.QWidget()
        self.Manual_wid.setEnabled(False)
        self.Manual_wid.setObjectName("Manual_wid")
        self.Web_tweet = QtWebEngineWidgets.QWebEngineView(self.Manual_wid)
        self.Web_tweet.setGeometry(QtCore.QRect(10, 10, 771, 431))
        self.Web_tweet.setUrl(QtCore.QUrl("about:blank"))
        self.Web_tweet.setObjectName("Web_tweet")
        self.Prevous_button = QtWidgets.QPushButton(self.Manual_wid)
        self.Prevous_button.setGeometry(QtCore.QRect(10, 450, 80, 22))
        self.Prevous_button.setObjectName("Prevous_button")
        self.Next_button = QtWidgets.QPushButton(self.Manual_wid)
        self.Next_button.setGeometry(QtCore.QRect(100, 450, 80, 22))
        self.Next_button.setObjectName("Next_button")
        self.Num_tweet = QtWidgets.QLineEdit(self.Manual_wid)
        self.Num_tweet.setGeometry(QtCore.QRect(60, 480, 71, 31))
        self.Num_tweet.setReadOnly(True)
        self.Num_tweet.setObjectName("Num_tweet")
        self.Relevant_button = QtWidgets.QPushButton(self.Manual_wid)
        self.Relevant_button.setGeometry(QtCore.QRect(250, 450, 111, 61))
        self.Relevant_button.setObjectName("Relevant_button")
        self.Irrelevant_button = QtWidgets.QPushButton(self.Manual_wid)
        self.Irrelevant_button.setGeometry(QtCore.QRect(380, 450, 111, 61))
        self.Irrelevant_button.setObjectName("Irrelevant_button")
        self.Continue_button = QtWidgets.QPushButton(self.Manual_wid)
        self.Continue_button.setGeometry(QtCore.QRect(670, 490, 80, 31))
        self.Continue_button.setObjectName("Continue_button")
        self.Start_button = QtWidgets.QPushButton(self.Manual_wid)
        self.Start_button.setEnabled(False)
        self.Start_button.setGeometry(QtCore.QRect(270, 520, 201, 22))
        self.Start_button.setObjectName("Start_button")
        self.Widget.addTab(self.Manual_wid, "")
        self.API_wid = QtWidgets.QWidget()
        self.API_wid.setEnabled(False)
        self.API_wid.setObjectName("API_wid")
        self.ck_text = QtWidgets.QTextEdit(self.API_wid)
        self.ck_text.setGeometry(QtCore.QRect(280, 90, 491, 31))
        self.ck_text.setObjectName("ck_text")
        self.cs_text = QtWidgets.QTextEdit(self.API_wid)
        self.cs_text.setGeometry(QtCore.QRect(280, 140, 491, 31))
        self.cs_text.setObjectName("cs_text")
        self.at_text = QtWidgets.QTextEdit(self.API_wid)
        self.at_text.setGeometry(QtCore.QRect(280, 190, 491, 31))
        self.at_text.setObjectName("at_text")
        self.ats_text = QtWidgets.QTextEdit(self.API_wid)
        self.ats_text.setGeometry(QtCore.QRect(280, 240, 491, 31))
        self.ats_text.setObjectName("ats_text")
        self.ck_label = QtWidgets.QLabel(self.API_wid)
        self.ck_label.setGeometry(QtCore.QRect(120, 100, 101, 16))
        self.ck_label.setObjectName("ck_label")
        self.cs_label = QtWidgets.QLabel(self.API_wid)
        self.cs_label.setGeometry(QtCore.QRect(120, 150, 131, 16))
        self.cs_label.setObjectName("cs_label")
        self.at_label = QtWidgets.QLabel(self.API_wid)
        self.at_label.setGeometry(QtCore.QRect(120, 200, 141, 16))
        self.at_label.setObjectName("at_label")
        self.ats_label = QtWidgets.QLabel(self.API_wid)
        self.ats_label.setGeometry(QtCore.QRect(120, 250, 161, 16))
        self.ats_label.setObjectName("ats_label")
        self.Continue_button_2 = QtWidgets.QPushButton(self.API_wid)
        self.Continue_button_2.setGeometry(QtCore.QRect(300, 410, 201, 61))
        self.Continue_button_2.setObjectName("Continue_button_2")
        self.label_12 = QtWidgets.QLabel(self.API_wid)
        self.label_12.setGeometry(QtCore.QRect(40, 50, 211, 16))
        self.label_12.setObjectName("label_12")
        self.label_13 = QtWidgets.QLabel(self.API_wid)
        self.label_13.setGeometry(QtCore.QRect(40, 290, 211, 16))
        self.label_13.setObjectName("label_13")
        self.ats_text_2 = QtWidgets.QTextEdit(self.API_wid)
        self.ats_text_2.setGeometry(QtCore.QRect(280, 330, 491, 31))
        self.ats_text_2.setObjectName("ats_text_2")
        self.ats_label_2 = QtWidgets.QLabel(self.API_wid)
        self.ats_label_2.setGeometry(QtCore.QRect(120, 340, 161, 16))
        self.ats_label_2.setObjectName("ats_label_2")
        self.Widget.addTab(self.API_wid, "")
        self.Relevant_wid = QtWidgets.QWidget()
        self.Relevant_wid.setEnabled(False)
        self.Relevant_wid.setObjectName("Relevant_wid")
        self.final_label = QtWidgets.QLabel(self.Relevant_wid)
        self.final_label.setGeometry(QtCore.QRect(280, 220, 321, 71))
        self.final_label.setObjectName("final_label")
        self.Widget.addTab(self.Relevant_wid, "")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.Widget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Relevant replies extr
         actor"))
        self.Explanation_label.setText(_translate("MainWindow", "3. Add a row at
         the beginning, and insert an id with an \'s\' at the end."))
        self.Start_parents_button.setText(_translate("MainWindow", "START"))
        self.label.setText(_translate("MainWindow", "INSTRUCTIONS"))
        self.label_2.setText(_translate("MainWindow", "1. Create an Excel file wi
         th the following requirements inside the folder that contains the tool:")
         )
        self.label_3.setText(_translate("MainWindow", "- Name of the file: a.xls"
         ))
        self.label_4.setText(_translate("MainWindow", "- Sheet name: TWEETS"))
        self.label_5.setText(_translate("MainWindow", "- Column name: id"))
        self.label_7.setText(_translate("MainWindow", "2. Copy the column contain
         ing the tweets ids into the created file, in the sheet TWEETS and in the
         column id"))
        self.label_6.setText(_translate("MainWindow", "EXAMPLE"))
        self.label_8.setText(_translate("MainWindow", "row 0 >> id"))
        self.label_9.setText(_translate("MainWindow", "row 1 >> 43266727351891148
         8s"))
        self.label_10.setText(_translate("MainWindow", "row 2 >> 4326672735189114
         88"))
        self.label_11.setText(_translate("MainWindow", "row 3 >> 4326724596286218
         24"))
        self.Widget.setTabText(self.Widget.indexOf(self.Parents_wid), _translate(
         "MainWindow", "Parents Identification"))
        self.Prevous_button.setText(_translate("MainWindow", "Previous"))
        self.Next_button.setText(_translate("MainWindow", "Next"))
        self.Relevant_button.setText(_translate("MainWindow", "Relevant"))
        self.Irrelevant_button.setText(_translate("MainWindow", "Not relevant"))

        self.Continue_button.setText(_translate("MainWindow", "Continue"))
        self.Start_button.setText(_translate("MainWindow", "START ANNOTATION"))
        self.Widget.setTabText(self.Widget.indexOf(self.Manual_wid), _translate("
         MainWindow", "Manual annotation"))
        self.ck_label.setText(_translate("MainWindow", "Consumer Key"))
        self.cs_label.setText(_translate("MainWindow", "Consumer Secret"))
        self.at_label.setText(_translate("MainWindow", "Access token"))
        self.ats_label.setText(_translate("MainWindow", "Access token secret"))
        self.Continue_button_2.setText(_translate("MainWindow", "CONTINUE"))
        self.label_12.setText(_translate("MainWindow", "TWITTER API PASSWORDS"))

        self.label_13.setText(_translate("MainWindow", "ROSETTE API PASSWORD"))
        self.ats_label_2.setText(_translate("MainWindow", "User Key"))
        self.Widget.setTabText(self.Widget.indexOf(self.API_wid), _translate("Mai
nWindow", "API request"))
        self.final_label.setText(_translate("MainWindow", "Results are stored in
\'relevant_replies.xlsx\'"))
        self.Widget.setTabText(self.Widget.indexOf(self.Relevant_wid), _translate
("MainWindow", "Relevant replies"))


        self.Start_button.clicked.connect(lambda:act().start())
        self.Prevous_button.clicked.connect(lambda:act().previous())
        self.Next_button.clicked.connect(lambda:act().next())
        self.Start_parents_button.clicked.connect(lambda:parents_explore())
        self.Relevant_button.clicked.connect(lambda:act().next())
        self.Irrelevant_button.clicked.connect(lambda:act().delete())
        self.Continue_button.clicked.connect(lambda:act().write())
        self.Continue_button_2.clicked.connect(lambda:replies_explore())

from PyQt5 import QtWebEngineWidgets

import parent
import cascades
import pandas
import pandas as pd
from pandas import ExcelWriter

def parents_explore():
    #name_file=ui.Name_file_text.text()
    #name_column=ui.Name_column_text.text()
    parent.ex_parents()#Name_file_text,Name_column_text
    ui.Manual_wid.setEnabled(True)
    ui.Parents_wid.setEnabled(False)
    ui.Start_button.setEnabled(True)
    ui.Widget.setCurrentIndex(1)

def replies_explore():
    ck=ui.ck_text.toPlainText()
    cs=ui.cs_text.toPlainText()
    at=ui.at_text.toPlainText()
    ats=ui.ats_text.toPlainText()
    uk=ui.ats_text_2.toPlainText()
    cascades.ex_cascades(ck,cs,at,ats,uk)#ck_text,cs_text,at_text,ats_text
    ui.API_wid.setEnabled(False)
    ui.Relevant_wid.setEnabled(True)
    ui.Widget.setCurrentIndex(3)
```

```
class act(object):
   def __init__(self):
      self.sheet='TWEETS'
      self.df=pandas.read_excel('parents_tool.xls',self.sheet)

   def start(self):
      ui.Num_tweet.setText(str(self.df.values[0][0]))
      ui.Web_tweet.setUrl(QtCore.QUrl(self.df.values[0][2]))

   def farray(self):
      now=ui.Num_tweet.text()
      i=0
      for x in self.df.values:
         if now==str(x[0]):
            array=i
         i=i+1
      return array

   def write(self):
      writer=ExcelWriter('parents_tool_2.xlsx',engine='xlsxwriter',options={'st
       rings_to_urls': False})
      self.df.to_excel(writer, 'TWEETS')
      writer.save()
      ui.API_wid.setEnabled(True)
      ui.Manual_wid.setEnabled(False)
      ui.Widget.setCurrentIndex(2)

   def next(self):
      array=self.farray()
      finish=len(self.df)-1
      if array==finish:
         self.start()
      else:
         ui.Num_tweet.setText(str(self.df.values[array+1][0]))
         ui.Web_tweet.setUrl(QtCore.QUrl(self.df.values[array+1][2]))

   def previous(self):
      array=self.farray()
      ui.Num_tweet.setText(str(self.df.values[array-1][0]))
      ui.Web_tweet.setUrl(QtCore.QUrl(self.df.values[array-1][2]))

   def delete(self):
        array=self.farray()
        self.next()
        self.df=self.df.drop(self.df.index[[array]])
        self.write_partial()

    def write_partial(self):
        writer=ExcelWriter('parents_tool.xls',engine='xlsxwriter',options={'strin
         gs_to_urls': False})
        self.df.to_excel(writer, 'TWEETS')
        writer.save()

from PyQt5 import QtWebEngineWidgets
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QMainWindow

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```