



## beAWARE

Enhancing decision support and management services in extreme weather  
climate events

700475

### D4.1

## Social media crawling and monitoring of multiple sensing platforms

<b>Dissemination level:</b>	Public
<b>Contractual date of delivery:</b>	Month 12, 31 December 2017
<b>Actual date of delivery:</b>	Month 12, 22 December 2017
<b>Workpackage:</b>	WP4 Aggregation and semantic integration of emergency information for decision support and early warnings generation
<b>Task:</b>	T4.1 Social media monitoring T4.2 Monitoring machine sourcing information from IoT and M2M platforms
<b>Type:</b>	Report
<b>Approval Status:</b>	Final
<b>Version:</b>	1.0
<b>Number of pages:</b>	48
<b>Filename:</b>	D4.1_beAWARE_social-media-crawling-monitoring-multiple-sensing-platforms_2017-12-19_v1.0



## Abstract

This deliverable reports on social media crawling and multiple sensing platforms. Social media monitoring involves the collection and relevance classification analysis of Twitter content and multiple sensing platforms are associated with the processing of beAWARE's sensor data, along with their storage and metadata indexing. The initial version of the considered modules is presented in this document, in accordance with the pilot use case requirements in terms of data collection, and the position of the modules within the beAWARE overall platform. Deliverable D4.1 sets also the basis for further improvements by presenting the directions towards the advanced version of social media monitoring and of the framework for managing the multiple sensing platforms.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.



Co-funded by the European Union



This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

## History

Version	Date	Reason	Revised by
0.1	18.10.2017	Deliverable structure	Stefanos Vrochidis (CERTH)
0.2	01.11.2017	Deliverable draft	Stelios Andreadis (CERTH)
0.3	01.12.2017	Additional contribution	Hylke van der Schaaf (IOSB), Jürgen Moßgraber (IOSB)
0.4	12.12.2017	Ready for internal review	Stefanos Vrochidis (CERTH), Anastasios Karakostas (CERTH)
0.5	15.12.2017	Internal review	Stamatia Dasiopoulou (UPF)
1.0	22.12.2017	Final version	Anastasios Karakostas (CERTH)

## Author list

Organisation	Name	Contact Information
CERTH	Stelios Andreadis	andreadisst@iti.gr
CERTH	Stefanos Vrochidis	stefanos@iti.gr
CERTH	Anastasios Karakostas	akarakos@iti.gr
IOSB	Hylke van der Schaaf	hylke.vanderschaaf@iosb.fraunhofer.de
IOSB	Jürgen Moßgraber	juergen.mossgraber@iosb.fraunhofer.de
IOSB	Philipp Hertweck	Philipp.hertweck@iosb.fraunhofer.de

## **Executive Summary**

This deliverable reports on the first version of beAWARE's social media crawling and multiple sensing platforms, which collect and process social and sensor data. Sensor data and social data are two different sources of information which are involved in beAWARE project. The analysed content from both sources of information contributes to the delivery of a more comprehensive user experience. Both tools are discussed, in their first version, including directions for further improvements, in the context of beAWARE.

The social media monitoring tool involves the crawling, representation, storage and analysis of Twitter content, aiming to classify tweets as relevant or irrelevant to each use case scenario, for all languages considered, i.e. Greek, Italian, Spanish, and English. Relevance classification analysis aims to deliver to the beAWARE end user information that is potentially useful for decision makers, emergency managers and operators, through effective visualisations. The social media information is then passed to the text analysis module to extract concepts and locations from text, before sent to the Knowledge Base for integration and decision making. The social media monitoring module offers information from citizen observations that serves complementary to the text messages, audio messages, images and videos which are sent by the first responders.

Sensor data are also involved in the decision making process. This report presents a unified way to access not just the data from a multitude of different types of sensors, but also the meta-data about these sensors. A sensor network is crucial for making decisions in any area of operation, so the methods and tools for utilising the information from sensor data are presented in detail in this document.

## Abbreviations and Acronyms

<b>AAWA</b>	Alto Adriatico Water Authority
<b>API</b>	Application Programming Interface
<b>BOW</b>	Bag-of-Words
<b>CBOW</b>	Continuous Bag-of-Words
<b>DCNN</b>	Deep Convolutional Neural Network
<b>DF</b>	Document Frequency
<b>FMI</b>	Finnish Meteorological Institute
<b>HTTP</b>	Hypertext Transfer Protocol
<b>IG</b>	Information Gain
<b>ISO</b>	International Standards Organisation
<b>JSON</b>	JavaScript Object Notation
<b>KB</b>	Knowledge Base
<b>K-NN</b>	k-Nearest Neighbors
<b>LSI</b>	Latent Semantic Indexing
<b>MI</b>	Mutual Information
<b>MPEG</b>	Moving Picture Experts Group
<b>NN</b>	Neural Networks
<b>O&amp;M</b>	Observation & Measurement model
<b>OGC</b>	Open Geospatial Consortium
<b>PSAP</b>	Public Service Answering Point
<b>REST</b>	Representational State Transfer
<b>RF</b>	Random Forests
<b>SBS</b>	Sequential Backward Selection
<b>SFS</b>	Sequential Forward Selection
<b>SIFT</b>	Scale-Invariant Feature Transform
<b>SOS</b>	Sensor Observation Service
<b>SURF</b>	Speeded Up Robust Features
<b>SVM</b>	Support Vector Machine
<b>TF</b>	Term Frequency

<b>TFIDF</b>	Term Frequency Inversed Document Frequency
<b>URL</b>	Uniform Resource Locator
<b>VSM</b>	Vector Space Model
<b>WCS</b>	Web Coverage Service
<b>WFS</b>	Web Feature Service
<b>WMS</b>	Web Map Service

## Table of Contents

<b>1</b>	<b>INTRODUCTION.....</b>	<b>8</b>
<b>2</b>	<b>ARCHITECTURE.....</b>	<b>9</b>
<b>3</b>	<b>SOCIAL MEDIA MONITORING .....</b>	<b>10</b>
3.1	Framework overview.....	10
3.2	Data collection from Twitter .....	12
3.3	Data collection requirements.....	12
3.4	Data representation from Twitter content .....	14
3.5	Data annotation and training set creation .....	17
3.6	Social media multimodal classification .....	20
3.6.1	Social media image classification .....	20
3.6.2	Social media text classification .....	22
3.6.3	Evaluation .....	26
3.7	Integration of the social media module into the beAWARE system .....	35
<b>4</b>	<b>MULTIPLE SENSING PLATFORMS .....</b>	<b>37</b>
4.1	Sensor Types .....	37
4.2	Data Types .....	38
4.2.1	Time series.....	38
4.2.2	Geospatial Coverages .....	38
4.2.3	Images, video and text.....	39
4.3	Pilots .....	39
4.3.1	Flood Pilot.....	39
4.3.2	Fire Pilot.....	40
4.3.3	Heatwave Pilot.....	40
4.4	OGC SensorThings API Data Model .....	40
4.5	Mapping the sensors onto the Ontology .....	43
4.5.1	Time series.....	44
4.5.2	Geospatial Coverages .....	44
4.5.3	Images and Video .....	44
<b>5</b>	<b>CONCLUSIONS.....</b>	<b>45</b>
<b>6</b>	<b>REFERENCES.....</b>	<b>46</b>

## **1 INTRODUCTION**

Disaster monitoring based on sensor data and social media posts has raised a lot of interest in the domain of computer science the last decade, mainly due to the wide area of applications in public safety and security. The abundant nature of these data renders them as one of the most valuable sources to extract and deduct early warnings or identification of an ongoing or eminent disaster (Imran et al., 2015). This deliverable presents the first version of the considered social media monitoring and sensor data collection framework in beAWARE.

As far as social media are concerned, we present the framework developed for monitoring the flow of Twitter posts. The data collection from Twitter is ensured by connecting to the Streaming API of Twitter. Crawled data are stored and indexed, as a pre-processing step before their analysis, which focuses, at this version, on their classification as relevant or not to the considered pilot use cases. In the proposed beAWARE classification approach both visual (if any) and textual modality participate in the relevance classification stage.

In the case of sensor data, data types are firstly identified and discussed, especially the meta-data which are involved in sensor data wrappers. Moreover, the relation of the considered data and sensor types to the beAWARE pilot use case scenarios is reported. Furthermore, the OGC SensorThings API Data Model is presented in the context of beAWARE. Last but not least, sensor data are mapped to the beAWARE ontology, including time series data, geospatial coverages, images and video information.

Finally, we conclude our report by highlighting some remarks and lessons learnt from current experiments, and we further identify future directions for the improvement of the developed tools, in the context of beAWARE project.



## 2 ARCHITECTURE

The two sources of information which are considered in this document are social media and sensor data, under Task 4.1 (Social media monitoring) and Task 4.2 (Monitoring machine sourcing information from IoT and M2M platforms), respectively. The position of the modules concerning social media and sensor data in the beAWARE architecture are illustrated in Figure 1. The modules are part of WP4 in beAWARE, which aggregates emergency information for decision support, aiming to generate early warnings by semantically fusing data from multiple sources, and to assist the Twitter report generation on the PSAP visualisations.

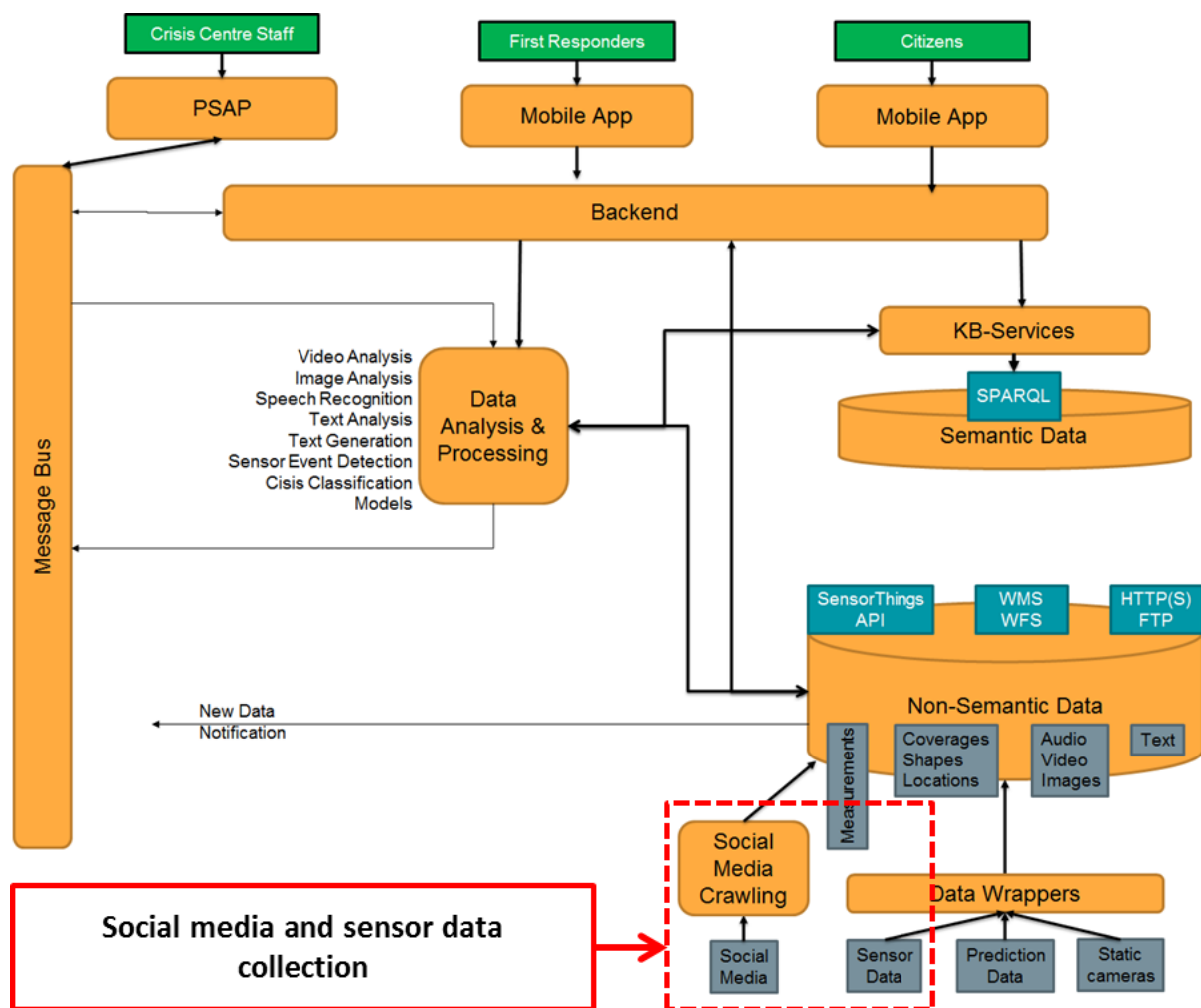


Figure 1: The beAWARE system architecture

The social media monitoring module interacts with the text analysis module, via a message bus. The text from the collected tweets is processed so as to extract high-level concepts and to estimate locations. The output of the text analysis module is then sent to the KB in a JSON format that is parsed for KB semantic integration.

In the following, section 3 describes the methods that deal with social media information and section 4 the framework of beAWARE which are associated with the collection and processing of sensor data.

### 3 SOCIAL MEDIA MONITORING

In this Section, we present the social media monitoring module and explicate how the classification of tweets as relevant or not is accomplished. Firstly, the data collection process is presented, based on the data requirements that have been identified by beAWARE user partners. Secondly, the data representation that is followed in beAWARE is discussed, based on the standard JSON format given by Twitter. Thirdly, the analysis on each tweet is done in two levels; textual and visual modalities are involved in the machine learning process. The module delivers only the classified-as-relevant tweets, as they are obtained by a multimodal classification service, being part of the social media monitoring module.

#### 3.1 Framework overview

The aim of the social media monitoring module is to collect posts from Twitter that appear to be relevant to the three main pilots, i.e. floods, fire, and heatwave, in their respective geographical locations. The crawling process needs to be real-time and effective, able to handle large streams of data, especially when keywords such as “fire” have multiple meanings and needs disambiguation. The module collects tweets in English, Greek, Italian and Spanish, which are published by citizens, civil protection organizations, online news websites or any other account, aiming to provide relevant information about crisis events.

The complete flow of beAWARE’s first version of the social media monitoring tool is demonstrated in Figure 2.

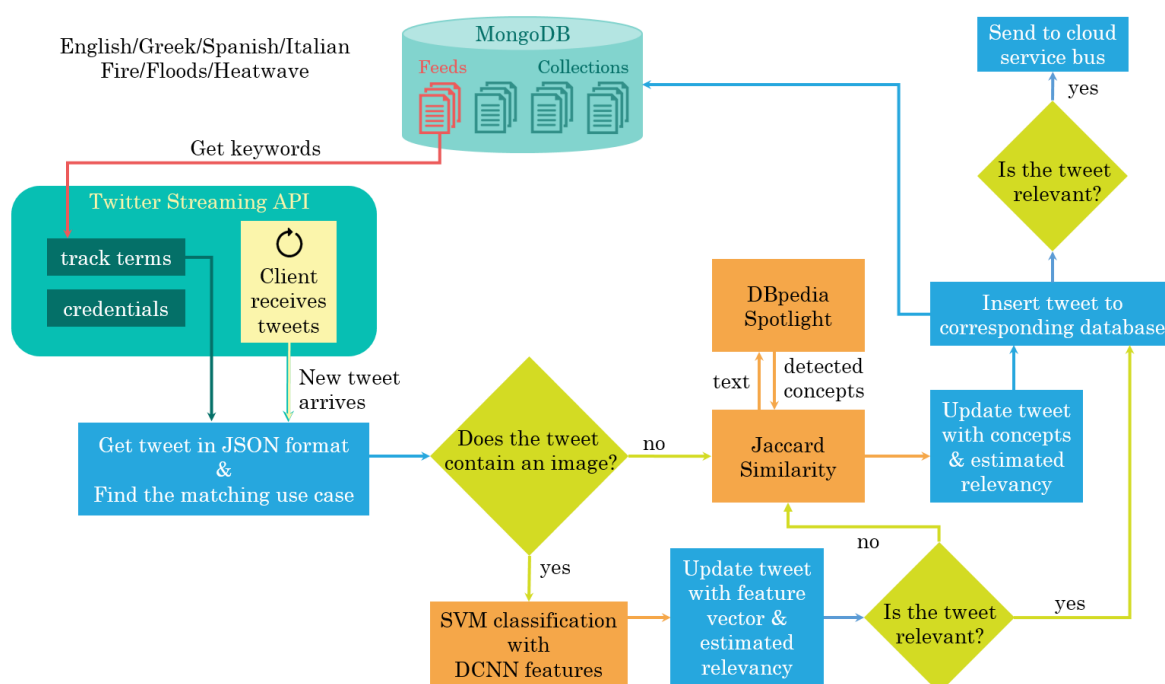


Figure 2: Complete flow of the beAWARE framework

Before the insertion to the database, we determine whether the new tweet is relevant or irrelevant to the pertinent use case (floods, fire or heatwave), so that only the relevant posts are forwarded to the respective analysis components, where the extraction of pertinent information (events, location, etc.) takes place. If an image was uploaded along with the tweet, we use the URL of the media to extract visual features based and then feed them to a pre-trained SVM classifier which returns a binary score, i.e. 1 for relevant and 0 for irrelevant, as described in Section 3.6.1 . Please note that this classification is language-independent, since only visual characteristics are taken into account. Then, the JSON object containing all the information of the tweet is updated to include the visual features and the estimated relevancy.

If the received tweet does not include an image or the SVM classifier returns that the attached image is irrelevant, we use the actual text of the tweet in order to estimate the relevancy, by comparing it with Twitter posts that were manually annotated. As described in the following, a dedicated graphical interface has been developed for browsing the compiled social media collections and annotating tweets as relevant/irrelevant. To enhance text comparison, we utilize the DBpedia Spotlight<sup>1</sup> that identifies and links (i.e. disambiguates) natural language mentions to respective DBpedia resources, e.g. given a tweet that mentions “Heavy rain and flood threat in northern Italy”, the DBpedia concepts “Rain”, “Flood”, and “Italy” will be extracted. Relevancy is estimated by performing Jaccard Similarity<sup>2</sup> between extracted concepts of the received tweet and the pre-detected concepts of every tweet of the targeted collection that was annotated as relevant. If there is an adequate number of annotated tweets, namely more than 20, and the maximum calculated similarity is larger than a constant “epsilon” (currently set at 0.3), the new tweet is considered as relevant; otherwise as irrelevant. Afterwards, the JSON object is again updated to include the extracted concepts and the estimated relevancy.

Finally, the updated JSON is inserted to the corresponding collection. In case that it has been estimated as relevant, either from the SVM classifier or the Jaccard Similarity method, it is pushed to the cloud service bus as a message that consists of the tweet’s unique identification, the matching use case, and a timestamp. All subscribers can access the marked Twitter post directly from the MongoDB database using the id provided in the message.

---

<sup>1</sup> <https://github.com/dbpedia-spotlight/dbpedia-spotlight>

<sup>2</sup> [https://en.wikipedia.org/wiki/Jaccard\\_index](https://en.wikipedia.org/wiki/Jaccard_index)

---

### 3.2 Data collection from Twitter

In order to gain access to Twitter’s global stream of data, we have exploited the Streaming APIs<sup>3</sup>, a streaming client that receives tweets the moment that they are published. Compared to Twitter’s REST APIs<sup>4</sup>, this option offers a real-time stream of tweets instead of constantly making requests and thus overriding any rate limiting, i.e. maximum number of requests. The only limitation when using the Streaming APIs is that each account is allowed to create only one standing connection.

There are various streaming endpoints that can be divided into the following categories: Public streams, User streams, and Site streams. In our case, the “POST statuses/filter” endpoint of public streams is the most suitable, since it focuses on public data flowing through Twitter that matches one or more filter predicates. Specifically, the “track” field can be used to define up to 400 search keywords, combined with an OR operator, so that the API will return tweets matching any of these keywords.

To consume Twitter’s Streaming API, we chose to adopt the Hosebird Client (hbc)<sup>5</sup>, an open-source and easy-to-use Java HTTP client. A required parameter is the user account’s credentials, while an optional parameter is the track keywords. For each combination of pilot and language we sustain a separate collection in a MongoDB database to store the crawled tweets. In addition, there is a “Feeds” collection where we define a set of relevant keywords to serve as track terms during the crawling procedure.

After connection with the Streaming API is established, the client constantly receives newly created tweets in JSON format. We choose to maintain the structure provided by the API, since the JSON format fits well with a MongoDB database. Every time a new tweet is retrieved, we examine which one of the track terms exists in the tweet’s text. In this way we can match the tweet to the corresponding language and pilot (e.g. “inundación” matches to Spanish and floods), in order to insert it to the respective collection.

### 3.3 Data collection requirements

The main target of the social media monitoring framework is to collect in a real-time manner any Twitter post that could be possibly reporting a crisis event. The crawling process is

---

<sup>3</sup> <https://dev.twitter.com/streaming/overview>

<sup>4</sup> <https://dev.twitter.com/rest/public>

<sup>5</sup> <https://github.com/twitter/hbc>

---

achieved with Twitter’s Streaming API, which offers three alternative ways to filter what data should be consumed:

1. Get statuses that contain any keyword of a predefined list
2. Get statuses from particular user accounts.
3. Get statuses that were posted in certain locations, specified by bounding boxes.

The first option fits our goal the most, since a set of keywords is a usual practice to detect data that refer to specific use cases, and thus it was adopted in this framework. On the other hand, the third option could serve in the future as an extension.

Language	Use case scenario		
	Floods	Fires	Heatwave
English	flooding	forest_fires	heatwave
Greek	πλημμύρες πλημμύρες GSCP_GR	πυρκαγιά πυρκαγια πυρκαγιες πυρκαγιές pyrosvestiki GSCP_GR	καύσωνας Κελσίου θερμοκρασία_ρεκόρ GSCP_GR
Italian	alluvione alluvionevicenza allagamento bacchiglione fiumepiena allertameteo sottopassoallagato alluvione2017 allertameteovicenza esondazione livellofiume	fiamme vigilidelfuoco piromane pompieri	ondatedicalore allertacaldo emergenzacaldo altetemperature troppocaldo
Spanish	fuertesprecipitaciones gotafría inundación inundaciones desbordamiento riada riadas lluviastorrenciales tormentas caudal-desbordado	incendio llamasdefuego bomberos focodeincendio	oladecolor altastemperaturas nochetropical golpedecolor sequía

Table 1: Search keywords per language and pilot

The initial set of keywords was composed of words suggested by the beAWARE user group members, for the languages covered in beAWARE’s use cases, namely Greek, Italian, and Spanish, and for English, as an additional control and demonstration language. However,

after using the proposed keywords in practice, it was noticed that some of them were bringing a large number of irrelevant tweets, so they were ignored in the current implementation that also involves the creation of a training set with balanced ratio between relevant and irrelevant Twitter posts. The set of tested and ignored keywords are: allerta, pioggia, maltempo, sottacqua, soccorso, Vicenza, protezionecivile, regioneveneto, piena, veneto, and corte de carretera in the case of floods; incendio, forestali, fumo, allertaincendio, fuego, humo, quema/quemas, conato, quemada/quemado, extinción, inhalación de humo, intoxicación de humo, evacuación/confinamiento in the case of fires; protezionecivile and hipertermia in the case of heatwave scenarios.

The complete list of keywords is shown in Table 1, separated by language and use case scenario. It can be noticed that some of the listed keywords are aggregated words (e.g. allertameteo), since (i) it is a common format in Twitter due to hashtags or the character limit, and (ii) these words separately are expected to return more irrelevant than relevant tweets.

After Twitter's response, we store the collected data so as to be able to proceed with further analysis on the collected content. The representation of the gathered tweets is presented in the following section.

### **3.4 Data representation from Twitter content**

After a connection is opened between our framework and the Twitter Streaming API, new results are sent through this connection whenever a matched post is published. The received tweets are encoded in JSON format, which is based on key-value pairs, with named attributes and associated values. We prefer to keep this provided structure while storing the tweets in our database, because JSON is indicated for MongoDB installations.

Figure 3 displays a screenshot of a tweet that is stored in Mongo, in a graphic representation of the JSON format, and all fields can be seen together with their content and type.






































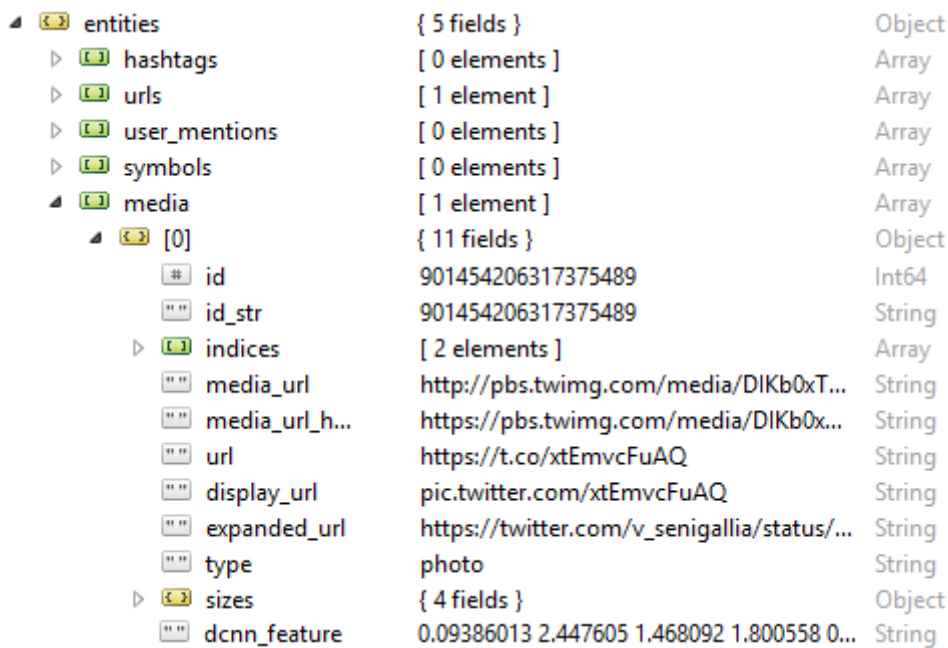
	_id	ObjectId("59a1884acac8ed4f63a47fa5")	ObjectId
	created_at	Sat Aug 26 14:40:06 +0000 2017	String
	id	901454209962332160	Int64
	id_str	901454209962332160	String
	text	Vivi Senigallia sulle indagini per l'alluvi...	String
	display_text_range	[ 2 elements ]	Array
	source	<a href="https://dlvrit.com/" rel="nof...	String
	truncated	true	Boolean
	in_reply_to_status_id	null	Null
	in_reply_to_status_id_str	null	Null
	in_reply_to_user_id	null	Null
	in_reply_to_user_id_str	null	Null
	in_reply_to_screen_name	null	Null
	user	{ 37 fields }	Object
	geo	{ 2 fields }	Object
	type	Point	String
	coordinates	[ 2 elements ]	Array
	[0]	43.7197926	Double
	[1]	13.2152224	Double
	coordinates	{ 2 fields }	Object
	place	{ 9 fields }	Object
	contributors	null	Null
	is_quote_status	false	Boolean
	extended_tweet	{ 4 fields }	Object
	retweet_count	0	Int32
	favorite_count	0	Int32
	entities	{ 4 fields }	Object
	favorited	false	Boolean
	retweeted	false	Boolean
	possibly_sensitive	false	Boolean
	filter_level	low	String
	lang	it	String
	timestamp_ms	1503758406521	String
	concepts	Senigallia Alluvione	String
	estimated_relevancy	true	Boolean
	relevant	false	Boolean
	is_retweeted_status	false	Boolean

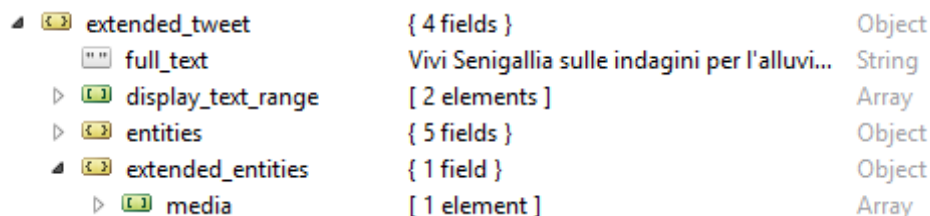
Figure 3: Original and additional fields of a tweet in a tree view

Figures Figure 4 and Figure 5 display an expanded view of two important fields, i.e. *media* and *extended\_tweet*, which are described in the next paragraph.



entities	{ 5 fields }	Object
hashtags	[ 0 elements ]	Array
urls	[ 1 element ]	Array
user_mentions	[ 0 elements ]	Array
symbols	[ 0 elements ]	Array
media	[ 1 element ]	Array
[0]	{ 11 fields }	Object
id	901454206317375489	Int64
id_str	901454206317375489	String
indices	[ 2 elements ]	Array
media_url	http://pbs.twimg.com/media/DIKb0xT...	String
media_url_h...	https://pbs.twimg.com/media/DIKb0x...	String
url	https://t.co/xtEmvcFuAQ	String
display_url	pic.twitter.com/xtEmvcFuAQ	String
expanded_url	https://twitter.com/v_senigallia/status/...	String
type	photo	String
sizes	{ 4 fields }	Object
dcnn_feature	0.09386013 2.447605 1.468092 1.800558 0...	String

Figure 4: Expanded view of field “entities”



extended_tweet	{ 4 fields }	Object
full_text	Vivi Senigallia sulle indagini per l'alluvi...	String
display_text_range	[ 2 elements ]	Array
entities	{ 5 fields }	Object
extended_entities	{ 1 field }	Object
media	[ 1 element ]	Array

Figure 5: Expanded view of field “extended\_tweet”

As it can be seen in the above figures, plenty of information is offered for a single post, but there are some fields that play a significant role in the social media monitoring procedure. To begin with, *id\_str* serves as a string identifier that can be used in different stages of the beAWARE framework in order to refer to a certain tweet, e.g. when communicating with the Text Analysis module. *text* is the main content of the Twitter post and it is used in text classification (Section 3.6.2) to estimate the tweet’s relevancy, while *entities.media.media\_url* (periods should be interpreted as “has subfield”) refers to the link of an attached image and it is used in image classification (Section 3.6.1), again for relevancy estimation. *created\_at* is the date when the post was published and *geo.coordinates* (if available) indicate the location where the tweet originated. Sometimes a tweet might exceed the maximum character limit and then a field named *extended\_tweet* is included. In that case, subfields *extended\_tweet.full\_text* and *extended\_tweet.entities.media.media\_url* are used in text and image classification respectively.



Apart from the aforementioned important fields, there are also some fields that are not originally included in the JSON format of a consumed tweet, but are added later by our framework, in the context of beAWARE project. In particular, the Boolean field *estimated\_relevancy* refers to the result of the multimodal classification, while the Boolean field *relevant* to the human annotation (Section 3.5 ). In order to decrease the response time of common queries to the database, *is\_retweeted\_status* defines whether the Object field *retweeted\_status* exists or not. Finally, *concepts* are the extracted concepts during text classification and *entities.media.dcnf\_feature* is the image feature vector during image classification. These are the fields that have been used so far to enrich the JSON structure of a Twitter post, in order to assist the analysis by making the information flow more efficient.

Before we proceed with the analysis of the Twitter content and its classification as relevant or not, we create an annotated set of tweets, having binary classification values, so as to incorporate and avail of user feedback in the classification stage. The annotated set of tweets is used as a training set, to train the models so as to be able to classify the incoming streams of Twitter content. For the purposes of beAWARE we created an annotation tool, as described below.

### 3.5 Data annotation and training set creation

The classification procedure, which is described in the following section, needs training data, i.e. a set of labeled examples. So, it is necessary to have a large number of tweets that are characterized as relevant/irrelevant. This is a manual task and thus requires human effort, such as end users that will serve as annotators. To facilitate this effort, an online application has been implemented, aiming to present the collected tweets in a straightforward manner and to provide an easy way to annotate. In this section we present a detailed description of the online annotation tool of beAWARE, which has been used to initialize the training set for each language and pilot use case scenario considered. Our approach allows for regular updates of the training set for further improvements of the developed classification models.

The homepage<sup>6</sup> of the web tool (Figure 6) offers the end users the ability to select the type of Twitter posts that they would like to be displayed, based on two criteria: language, i.e. English, Greek, Italian, and Spanish, and pilot case, i.e. fire, flood, and heatwave. Each combination of language and pilot case defines a different collection of crawled tweets. In a minimal, but efficient way the users are able to set their preferences; first, by a drop-down list on the upper left corner of the website that provides all the available languages and then

---

<sup>6</sup> [http://mklab-services.iti.gr/beAWARE\\_tweets](http://mklab-services.iti.gr/beAWARE_tweets)

---

by clicking one of the three pilot buttons in the middle of the page. This will navigate to the presentation of the respective tweets.

English ▼

Please select a **pilot**



Figure 6: Data annotation tool – homepage

Figure 7 depicts a set of tweets that are relative to floods and are written in Italian. As it can be easily seen, the page consists of two main components: a pagination header that also includes other useful utilities, and a panel of textboxes in vertical direction where each box displays a Twitter post.

In order to avoid presenting thousands of tweets in a single page, the application gives a pagination option that divides the posts into pages of fifty. Using the left and right arrows inside the header, end users are able to navigate back and forth through pages, while the page numbering and the total number of tweets are always shown. In addition, there are two filtering options: posts can be filtered either by their date of creation, or by the existence of certain keywords inside their text. For the former, a date slider serves to define the time period during which posts were created and then pressing the reload button, which lies next to the slider, is necessary to return the new results. For the latter, users can type an unlimited number of words into a text input field on the upper right side of the header and by clicking on the magnifier icon or pressing the Enter key, results will be updated with tweets that contain at least one of the given words in their text. In order to switch use cases, the “Change pilot” button navigates back to the homepage, while switching languages is possible through the drop-down list that was previously described and is always visible in the annotation tool. In addition to this header, a more compact version of it, which includes only the pagination functionality, was added under the panel of tweets, according to feedback from real end users.

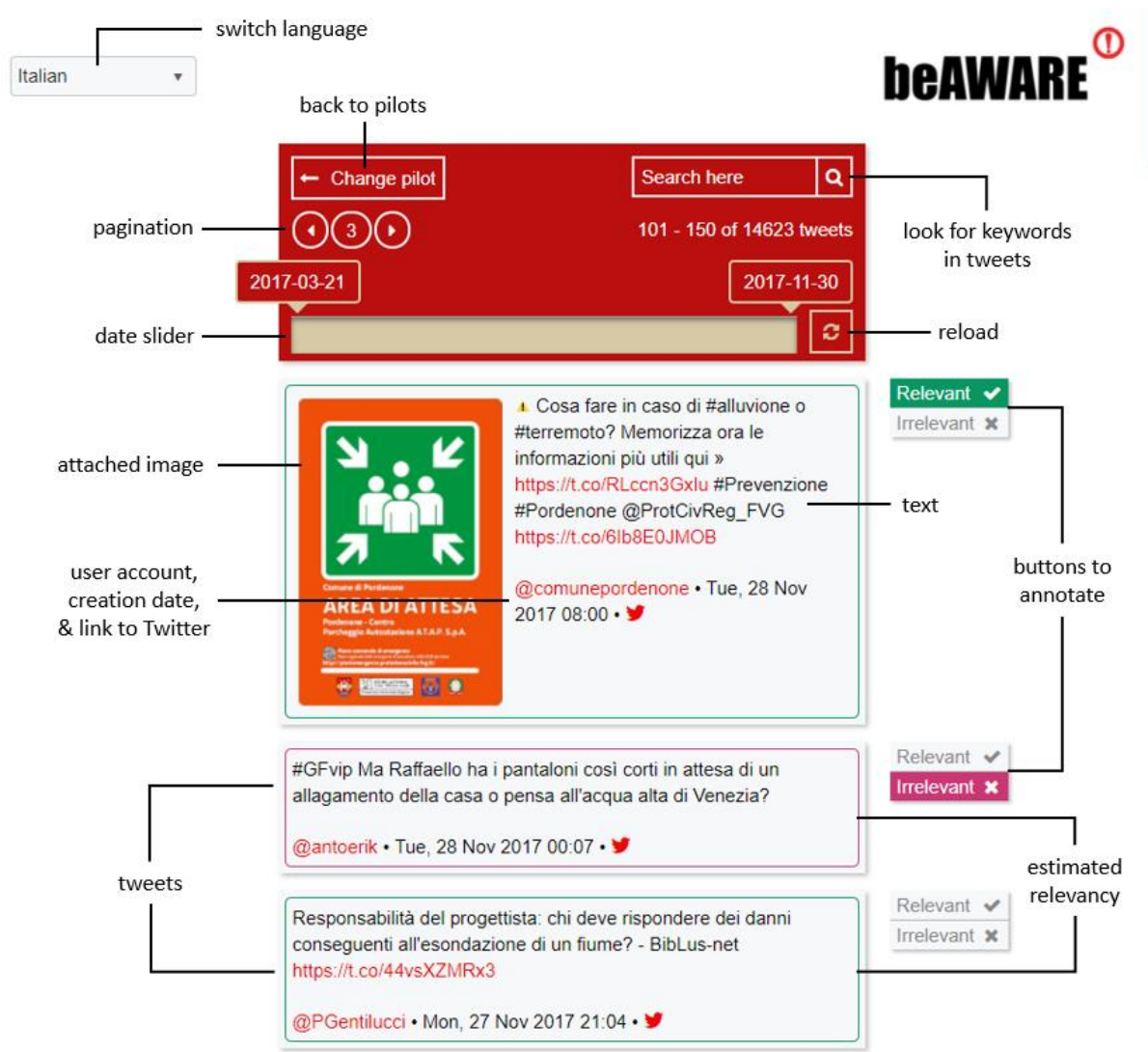


Figure 7: Data annotation tool – tweets presentation and buttons to annotate

Regarding the crawled Twitter posts, they are represented by a list of boxes and they are sorted from most recent to oldest. For each tweet, a variety of information is available. In detail, the main text of the post, along with images or active links if existing, the username of the author, which links to the user account's Twitter page, the time and date when the tweet was published, and a link to the original post in Twitter. Moreover, the colored border around each tweet indicates its relevancy to the pilot case as estimated by the multimodal classification framework (see Section 3.6 ) during the crawling phase. Green border refers to relevant texts and red violet border to irrelevant.

Next to every tweet box there are two buttons that offer the core functionality of the described application. By clicking the "Relevant" or the "Irrelevant" button, users are able to annotate all posts and the annotation is instantly saved in the database where the crawled tweets are stored, as an extra field (see Section 3.4 ). When a tweet is already annotated,

then the corresponding button is colored, but it can always be changed. However, the relevancy of a post can only have one value, regardless of the annotator, so the latest selection overwrites a previous choice.

In summary, the online annotation tool provides two basic functionalities; (i) a presentation of all crawled Twitter posts per language and pilot, together with filter options (e.g. getting tweets that were published in a specific date or that contain certain words), and (ii) a simple way to select if a tweet is relevant or not, thus creating the training set. The tool has already been used by Greek and Italian users, and since March 2017 more than 11,000 and 16,000 tweets respectively have been successfully annotated.

### **3.6 Social media multimodal classification**

Classification is the problem of identifying to which of a set of categories (i.e. classes) a new observation belongs to, on the basis of a training set of data containing observations whose class membership is known. It is a two-step process that involves the construction of a model by using a training set of the target category and then the application of the model for classifying previously unseen data. The algorithm that implements classification is known as a classifier, i.e. one function that maps one observation to a pre-defined class. Most algorithms describe an individual instance whose category is to be predicted using a feature vector that is comprised of measurable properties (i.e. features) of the instance. Features may be binary; categorical; integer-valued; or real-valued. In case the instance is an image, the feature values might correspond to the pixels of an image; if the instance is a piece of text, the feature values might be occurrence frequencies of different words.

In beAWARE project, the first version of the social media monitoring module considers both visual and textual information, aiming to filter-out irrelevant social media posts. The relevance is estimated using both textual and visual (if available) information.

In the sequel, we present an overview of the relevant work on visual and textual classification as well the framework applied for both cases, then an evaluation section follows which includes a short description of the datasets used, the experiments realized, the results produced and finally the conclusions drawn.

#### **3.6.1 Social media image classification**

Image classification involves the use of visual concept detection algorithms based on low-level features and classifiers for deciding whether an image shows evidence of flood, fire or heatwave. In this section, we present an overview of state of the art methods for concept detection in images, and then we present the framework proposed within beAWARE.

Concept detection in images aims at annotating them with one or more semantic concepts (e.g. hand, sky) that are chosen from a pre-defined concept pool. Concept detection systems involve the extraction of visual features, the training of classifiers for each concept using a ground-truth annotated training set, and eventually, the application of the trained classifiers to unlabeled images, that return a set of confidence scores for the appearance of the different concepts in the shot. Thus, the first step is feature extraction and the second is the building of the classification model.

Regarding feature extraction, it refers to the methods that aim at the description of the visual content of images. Visual descriptors can be divided in two main groups: hand-crafted and DCNN-based descriptors. Hand-crafted features can be further divided into global and local descriptors. Global descriptors capture global characteristics of the image and some indicative examples of global descriptors are the MPEG-7 descriptors, and the Grid Color Moments. Instead, local descriptors represent local salient points or regions and the most widely used are the SIFT descriptor (Lowe, 2004), and the SURF descriptor (Bay et al., 2008) and their variations. Usually, in the case of local descriptors a clustering algorithm is applied after the feature extraction in order to form a vocabulary of “visual words” that leads eventually to a global descriptor. The most known approaches for visual word assignment are the “bag-of-words” (BoW) representation (Qiu, 2002), the Fisher vector (Perronnin et al., 2010) and the VLAD (Jegou et al., 2010). As far as the DCNN-based features are concerned, they are the most recent trend in feature extraction and image representation and they seem to outperform the hand-crafted features in most applications. They learn features directly from the raw image pixels using Deep Convolutional Neural Networks (DCNNs), which consist of many layers of feature extractors and can be used both as standalone classifiers, i.e., unlabeled images are passed through a pre-trained DCNN that performs the final class label prediction directly, or as generators of image features, i.e., the output of a hidden layer of the pre-trained DCNN is used as a global image representation (Simonyan, 2014; Markatopoulou, 2015). The latter type of features is referred to as DCNN-based and they are usually preferred due to their high performance both in terms of time and accuracy. Several DCNN software libraries are available, e.g., Caffe (Jia, 2014), MatConvNet (Vedaldi, 2015), and different DCNN architectures have been proposed, e.g., CaffeNet (Krizhevsky, 2012), GoogLeNet (Szegedy, 2015).

Classification step is the second step of the multimedia concept detection process, and it involves the construction of models by using the low-level visual features, and then the application of these models for image labelling. Common classifiers that are used for learning the associations between the image representations and concept labels are the Support Vector Machines (SVM) and Logistic Regression (Markatopoulou, 2015). SVMs are trained separately for each concept, on ground-truth annotated corpora, and when a new

unlabeled video shot arrives, the trained concept detectors will return confidence scores that show the belief of each detector that the corresponding concept appears in the shot.

#### **beAWARE framework**

In the employed framework, we trained a 22-layer GoogLeNet network (Szegedy, 2015) on 5055 ImageNet concepts (Pittaras, 2017), which are a subset of the 12,988 ImageNet concepts. The subset of the 5055 concepts was produced by considering the tree structure of the ImageNet and the following assumptions: a) concepts that were very similar were merged, for example all different dog breeds (e.g. Shih-Tzu, Pekinese, Maltese dog) were removed and only the concept dog was kept. The same philosophy was followed for other animals and plants as well, b) concepts that correspond to scientific terms were removed, for example biological terms such as eukaryote, prokaryote, sporozoite etc., and c) concepts with very few number of positive images were removed. Then, this network was applied on the TRECVID SIN 2013 development dataset and we used as a feature (i.e., a global image representation) the output of the last pooling layer with dimension 1024. In the sequel, we used the annotated dataset for training and validating an SVM classifier per concept (i.e. flood, heatwave, fire). It should be noted that the SVM classifiers were tuned by setting different  $t$  and  $g$  values in order to achieve maximum performance.  $t$  parameter in SVM classifier defines the kernel type, while  $g$  stands for the gamma in the kernel function. It should be noted that apart from the DCNN-based features, several other features were evaluated as well, including acc, gabor.

The beAWARE image classification module is evaluated in two data collections, which involve visual and textual information, in Section 3.6.3 below, after the discussion on the state of the art in text classification in Section 3.6.2 .

#### **3.6.2 Social media text classification**

Social media text classification involves the use of text classifiers that consider textual features for deciding whether an image show evidence of flood, fire or heatwave. In this section, we present an overview of state of the art methods for text classification, then we present the framework proposed as well as some directions for the following version of the beAWARE social media text classification module.

Text classification is the assignment of natural language texts into one or more categories/classes drawn from a predefined set according to their content. Text classification involves the following series of steps:

1. Document collection, which involves the collection of data stored in several formats such as doc, html.

2. Preprocessing, which converts the original text data in a data-mining-ready structure, where the most significant text-features that serve to differentiate between text-categories are identified. Commonly, the steps involved are tokenization, where each document is partitioned into a list of tokens, stop word removal, that involves of removal of frequently occurring words (e.g. and, the), and word stemming, which reduces words to their root form.
3. Text representation (Yan, 2009), which models documents and transforms them into numeric vectors. The most commonly used text representation model is the Vector Space Model (VSM) where documents are represented by vectors of words. One of the commonly used VSM is the Bag of Words model (BOW) which uses all words appeared in the given document set  $D$  as the index of the document vectors. Different term weighting schemas were proposed under the BOW model that gives different text representation results. The simplest case of BOW is the Boolean model, where binary vectors represent documents. Extensions of the Boolean model is the Term Frequency model (TF) that uses the frequency of the terms, and the Term Frequency Inversed Document Frequency (TFIDF) model, which uses real values that capture the term distribution among documents to weight terms in each document vector. However, both TF and the TFIDF model have certain limitations such as the fact that they cannot capture polysemy and synonymity as well as the semantics of the documents. Later, more advanced text representation strategies have been proposed including the N-gram statistical language models that were proposed to capture the term correlation within document. However, the exponentially increasing data dimension with the increase of  $N$  limits the application of N-gram models. The Latent Semantic Indexing (LSI) was proposed to reduce the polysemy and synonym problems. One of the latest approaches that seems to outperform the other methods in many cases, is word2vec. Specifically, Mikolov et al. (2013) proposed novel architectures and models for producing word embeddings (i.e. representation of words from a given vocabulary as vectors in a low-dimensional space), based on deep neural networks (NN), namely the Continuous Bag-of-Words (CBOW) and the Skip-gram models, which are also referred as word2vec. CBOW and Skip-gram models are trained first on a large corpus, taking into consideration the neighbouring words in a sentence. The context size one can take into consideration is specified by a parameter called window size. In the CBOW architecture, the NN model tries to predict a word given the context of this word, whereas in the Skip-gram architecture, the exactly opposite function is executed, that is, given a word the NN model tries to predict the context of a word. Regarding the quality of these vectors, it is proved that these methods can capture very efficiently the semantics of the words.
4. Feature selection methods (Aggarwal, 2012; Chandrashekar, 2014), that are used for reducing the dimensionality of the dataset by removing features that are considered



irrelevant for the classification. The aim of these methods is to select a subset of variables from the input which can efficiently describe the input data while reducing effects from noise or irrelevant variables and still provide good prediction results. Feature selection techniques can be classified into two basic categories: filtering techniques and wrapper techniques. Filter methods act as preprocessing to rank the features wherein the highly ranked features are selected and applied to a predictor. In wrapper methods the feature selection criterion is the performance of the predictor i.e. the predictor is wrapped on a search algorithm which will find a subset which gives the highest predictor performance. In general, wrapper methods have low complexity, whereas wrapper methods have higher time complexity and accuracy than filter methods. Some filtering methods are the Document Frequency (DF), Information Gain (IG), and Mutual Information (MI). Some wrapper methods are Sequential Forward Selection (SFS), Sequential Backward Selection (SBS) and Neural Networks.

5. Classification Algorithms, which are used to model classes and label text. There are several methods used to classify text such as Support Vector Machine, Naive Bayes Classifier and Decision Trees.

The aforementioned text classification methods are applied to documents of normal length. However, unlike normal documents, short texts that are available in many application areas, such as Instant Messages, online Chat Logs, Bulletin Board System Titles, and Twitter are usually noisier, less topic-focused, and (way more) shorter, that is, they consist of from a dozen words to a few sentences, and finally they contain many non-standard terms. Because of the short length, they do not provide enough word co-occurrence or shared context for a good similarity measure (Song, 2014). Therefore, traditional machine learning methods, such as SVM, Bayes and K-NN, which rely on the word frequency, tend not to perform as good. Thus, new classifying methods on short text started to appear, such as semantic analysis, semi-supervised short text classification, ensemble models for short text, and real-time classification in order to deal with the problem of short text classification. Popular methodologies (Song, 2014) used for short text classification include short text classification using semantic analysis, semi-supervised short text classification, ensemble short text classification, and real-time classification. Due to the extensive use and increase of popularity of Twitter, a number of methods have been proposed that focus on tweet classification (Selvaperumal, 2014). Some ideas that were proposed for tweet classification are the following: the use of emoticons, the use of a network algorithm that classifies tweets based on finding the similar trend topics, the application of data compression, the use of tweet features like URL's, the retweeted tweets and the influential users tweet, the use of Wikipedia and wordnet to cluster short texts and others methods.



Apart from the (short) text classification approaches discussed above, it is possible to conclude whether a document belongs to a specific class by calculating its similarity with the instances (e.g. tweets) that belong to the class. There are a number of string similarity measures that estimate the similarity between two sequences of strings. The most popular term-based distance measures are the following: Block distance which is known as Manhattan distance, the cosine similarity, the Dice's coefficient, the Euclidean distance, the Jaccard Similarity, the Overlap coefficient and the Matching coefficient (Vijaymeena, 2016). The maximum of the similarity or minimum distance calculated between the query document and the set of documents belonging to the class of interest is compared to a threshold value that is defined empirically in order to decide whether the query document belongs or not to the specific class.

#### **beAWARE framework**

In the employed framework, we evaluated several methods belonging to the traditional text classification, as well as the Jaccard similarity method. Thus, for the traditional text classification we approach each of the aforementioned steps as follows:

1. We collect short text messages from Twitter, as already described in Section 3.2.
2. We pre-process the collected text in the following ways:
  - a) We apply DBpedia Spotlight in order to automatically annotate it with respective DBpedia resources (Daiber, 2013). It should be noted that DBpedia resources have underlying semantics, however currently they are treated as plain words. It is possible that we consider this information in next version of the module.
  - b) We remove punctuation and all non-characters, as well as stop words from the collected text and finally we do word stemming.
3. As far as text representation is concerned, we tested Term Frequency (TF), TFIDF and word2vec. Various experiments were realized for different feature length and n-gram values (i.e. n-gram = 1 or 2) for the first two representation methods, and different corpus and vector dimensions for the third method.
4. We do not apply feature selection in the current version of the text classification module.
5. We serve each text feature vector as input to a classifier (i.e. SVM, Naïve Bayes or Random Forests) which is tuned in order to achieve maximum performance. The textual feature vector is constructed using either DBpedia concepts or raw text, so as to examine which is the most suitable representation in the context of beAWARE.

Regarding the Jaccard similarity approach, we follow the same collection (step 1) and preprocessing (step 2) steps as the ones realized in the traditional text classification method. However, in the sequel we compute for each text representation (i.e. DBpedia concepts, text with stop words removed, and text with stop words removed and with word stemming) the Jaccard similarity coefficient between the new text description and each positively annotated text description, using the mathematical formula  $J(W_q, W_{t_n}) = \frac{|W_q \cap W_{t_n}|}{|W_q \cup W_{t_n}|}$ , where  $W_q$  stands for the set of terms of the new text description, and  $W_{t_n}$  for the set of terms of the  $n$  text description of the positively annotated dataset tests. Then the maximum value of the Jaccard similarity coefficients was compared to a threshold defined empirically in order to determine whether the new text description will be considered as positive or not. Specifically, if the similarity was greater than the selected threshold the new instance was considered as positive.

Finally, we should note that although the text used in beAWARE is retrieved from Twitter, in the current, baseline version of the text classification, we used traditional methods followed for normal length documents. However, it is expected that in the next versions, methods that consider the particular characteristics of tweets will be evaluated.

### 3.6.3 Evaluation

In this section we evaluate the frameworks proposed in two different datasets. Thus, we initially present the datasets and then present the experimental results for various features and classifiers applied. It should be noted that both textual and visual classification modules were developed and tested for the *flood* concept and similar behaviour is expected in the other two use cases of fire and heatwave events.

#### Dataset Description

The datasets, which were used for developing and evaluating both the visual and textual classification modules, are:

- the MediaEval 2017 dataset for the Multimedia Satellite Task<sup>7</sup>
- the beAWARE dataset

Regarding the MediaEval 2017 dataset, it was provided within the context of the Disaster Image Retrieval from Social Media (DIRSM) subtask which goal was to identify all images which show direct evidence of a flooding event from social media streams, independently of a particular event. It should be noted that within the context of DIRSM subtask a set of visual

---

<sup>7</sup> <https://multimediaeval.github.io/2017-Multimedia-Satellite-Task/>

descriptors were also precomputed and provided to the testers which were acc, gabor, fcth, jcd, cedd, eh, sc, cl, and tamura. These descriptors were evaluated during the building of the visual classifier. The dataset comprises of 5,280 images, 1,920 of which are annotated as true and 3,360 as false regarding the flood event. For evaluation purposes the dataset was split into two subsets; a training and a validation set that contained 3,520 and 1,760 images respectively. Table 2 contains the statistics of the MediaEval dataset.

Regarding the beAWARE dataset, it is constructed from the tweets retrieved by the social media crawling module (section 3.3 ). Since the focus in the first version of the social media classification module is on the *flood* concept, the beAWARE dataset used contains tweets in Italian that are related to the flood event. Thus, a significant effort was realised on behalf of the Italian partners of the beAWARE and a considerable number of tweets were annotated to aid in the classifier development. Specifically, 11,931 tweets were annotated, 5,171 of which are annotated as true and 6,760 as false regarding the flood event. Table 3 contains the statistics of the beAWARE dataset. However, it should be noted that the annotation realised by the beAWARE partners refers both to the text and image (if available) of the tweet and thus, it is often that while a tweet is relevant to *flood* event, the image included to be irrelevant, which affects eventually the evaluation metrics.

	Annotation for concept 'flood'		Sum
	True	False	
Train set	1,280	2,240	3,520
Validation set	640	1,120	1,760
Total Records	1,920	3,360	5,280

Table 2: Statistics of MediaEval 2017 dataset

	Annotation for concept 'flood'		Sum
	True	False	
All	5,171	6,760	11,931
Only Text	4,261	5,989	10,250
Only Text (duplicates removed)	4,204	5,859	10,063
Text + Image	910	771	1,681
Text + Image (Image exists)	855	739	1,594

Table 3: Statistics of beAWARE dataset

## Experiments

In order to evaluate the quality of the classification system the metrics that used in mostly are precision, recall, and fscore. These metrics are calculated in every run in order to decide the best performing classification method.

### Social media image classification

In order to find the best performing feature and classifier for identifying images that contain evidence of flood, several features were tested and the parameters of SVM classifiers were tuned in order to maximise their performance. Specifically, as far as the MediaEval dataset is concerned, the following features were tested *acc*, *gabor*, *fcth*, *jcd*, *cedd*, *eh*, *sc*, *cl*, and *tamura* that were provided for the Multimedia-Satellite challenge and the DCNN-based features. Moreover, SVM classifiers were trained for all of these features for different *t* and *g* parameters and results showed that the proposed DCNN feature outperformed most of them significantly. Table 4 contains the evaluation metrics for the MediaEval dataset for the different visual descriptors and SVM classifiers. After inspecting Table 4, we can deduce that the best results are obtained for the DCNN-based features for  $t = 1$  (polynomial function) and  $g = 0.5$  or  $g = 0,03125$ .

Descriptor	SVM Parameters		Precision	Recall	Accuracy	Fscore
	<i>t</i>	<i>g</i>				
acc	1	0,00125	0,5827	0,3359	0,6710	0,4262
acc	1	0,03125	0,5359	0,1516	0,6438	0,2363
acc	1	0,5	0,4830	0,1328	0,6330	0,2083
acc	2	0,5	0,6739	0,0484	0,6455	0,0904
cedd	1	0,00125	0,6427	0,5453	0,7244	0,5900
cedd	1	0,03125	0,6085	0,5391	0,7063	0,5717
cedd	1	0,5	0,5925	0,3953	0,6813	0,4742
cedd	2	0,5	0,8250	0,0516	0,6511	0,0971
cl	1	0,00125	0,6005	0,3500	0,6790	0,4423
cl	1	0,03125	0,6115	0,3641	0,6847	0,4564
cl	1	0,5	0,5957	0,3016	0,6716	0,4004
cl	2	0,5	0,6600	0,5156	0,7273	0,5789
eh	1	0,00125	0,6935	0,4703	0,7318	0,5605
eh	1	0,03125	0,6682	0,4688	0,7222	0,5510
eh	1	0,5	0,6605	0,4469	0,7153	0,5331
eh	2	0,5	0,2500	0,0031	0,6341	0,0062
fcth	1	0,00125	0,6146	0,4609	0,6989	0,5268
fcth	1	0,03125	0,5956	0,4625	0,6903	0,5207
fcth	1	0,5	0,5000	0,2578	0,6364	0,3402
fcth	2	0,5	NaN	0,0000	0,6364	0,0000
gabor	1	0,00125	NaN	0,0000	0,6364	0,0000
gabor	1	0,03125	NaN	0,0000	0,6364	0,0000
gabor	1	0,5	NaN	0,0000	0,6364	0,0000
gabor	2	0,5	NaN	0,0000	0,6364	0,0000
jcd	1	0,00125	0,6465	0,5516	0,7273	0,5953
jcd	1	0,03125	0,6388	0,5250	0,7193	0,5763

jcd	1	0,5	0,6025	0,3719	0,6824	0,4599
jcd	2	0,5	NaN	0,0000	0,6364	0,0000
sc	1	0,00125	0,2381	0,0078	0,6301	0,0151
sc	1	0,03125	1,0000	0,0016	0,6369	0,0031
sc	1	0,5	0,2500	0,0016	0,6352	0,0031
sc	2	0,5	0,3900	0,0609	0,6239	0,1054
tamura	1	0,00125	0,2500	0,0031	0,6341	0,0062
tamura	1	0,03125	0,5246	0,0500	0,6381	0,0913
tamura	1	0,5	0,3913	0,0141	0,6335	0,0271
tamura	2	0,5	0,5455	0,0094	0,6369	0,0184
dcnn-based	1	0,00125	0,8192	0,8000	0,8631	0,8095
dcnn-based	1	0,03125	0,8195	0,8016	0,8636	0,8104
dcnn-based	1	0,5	0,8195	0,8016	0,8636	0,8104
dcnn-based	2	0,5	0,9000	0,0141	0,6409	0,0277

Table 4: Evaluation of visual features and classifiers in MediaEval dataset

As far as the beAWARE dataset is concerned, we tested the DCNN-based feature and the SVM classifier with  $t = 1$  and  $g = 0.5$  that performed best in the MediaEval dataset and the evaluation metrics can be found in Table 5. However, it is evident that the recall achieved by applying the visual classification method in the beAWARE dataset is rather low, which leads to a low F-score as well. In order to discover, the reasons behind such low recall, a more detailed annotation of the images of beAWARE dataset is required. This will reveal whether the low performance is actually false or it is accurate and the visual classifier will need retraining by considering also images of the beAWARE dataset.

Descriptor	SVM Parameters		Precision	Recall	Accuracy	Fscore
	$t$	$g$				
dcnn-based	1	0,5	0,7139	0,2738	0,5461	0,3957

Table 5: Evaluation of visual features and classifiers in beAWARE dataset

### ***Social media text classification***

The first method evaluated for text classification uses the Jaccard Similarity Coefficient. Figures Figure 8: Evaluation of Jaccard Similarity method for MediaEval dataset

and Figure 9: Evaluation of Jaccard Similarity method for beAWARE dataset

depict the F-score values for the MediaEval and beAWARE datasets accordingly for different text input and different values of the  $e$  parameter.

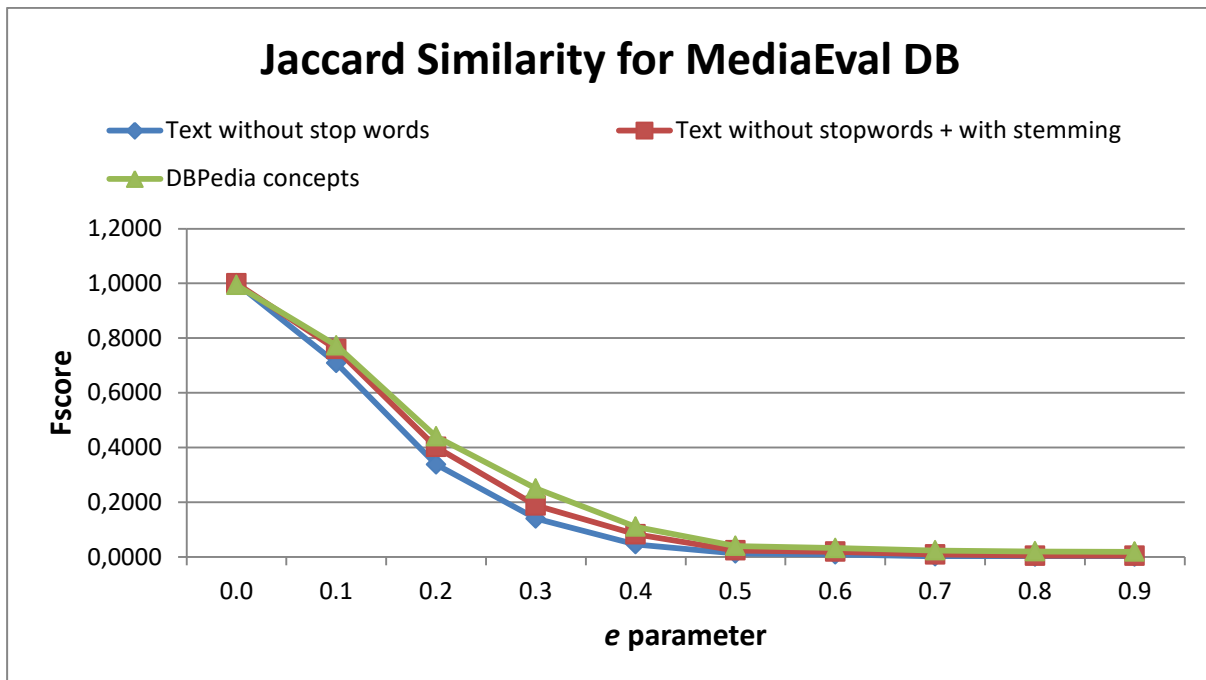


Figure 8: Evaluation of Jaccard Similarity method for MediaEval dataset

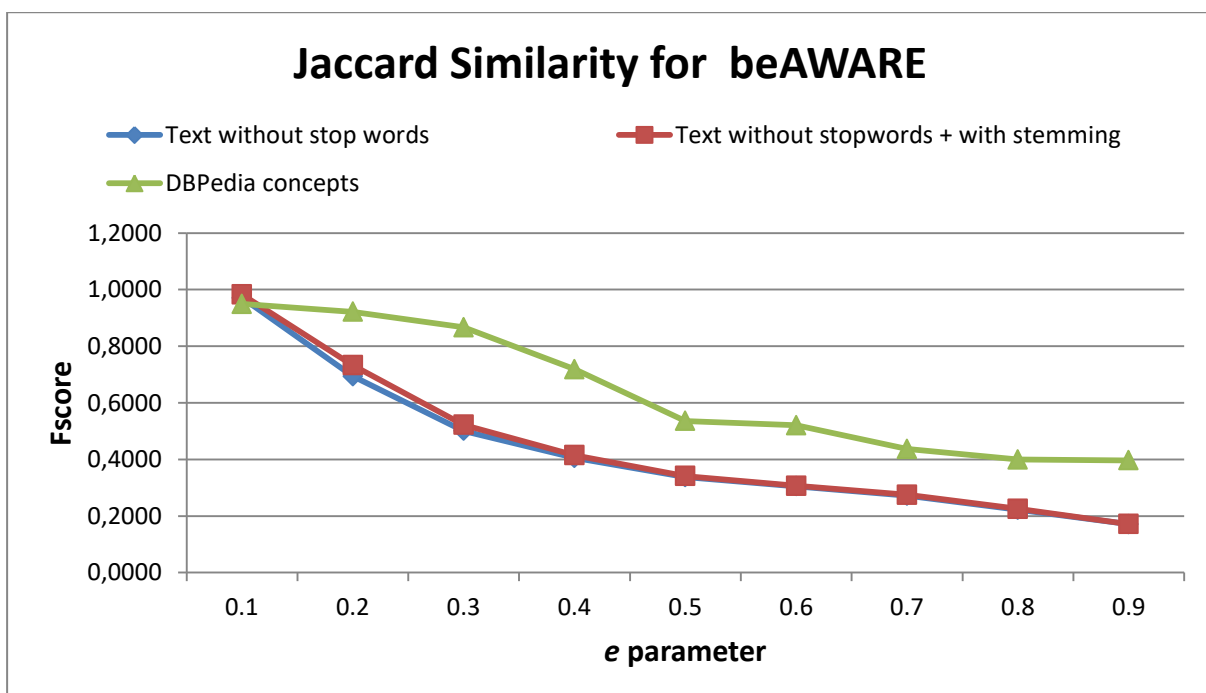


Figure 9: Evaluation of Jaccard Similarity method for beAWARE dataset

After a careful observation of the two figures, we can conclude that for the Jaccard method the use of DBpedia concepts slightly improves the classification performance. However, it should be noted that especially for the MediaEval dataset the Jaccard method is performing satisfactorily.

Moreover, it is evident that the method has good results for very low values of the  $e$  parameter, i.e. around 0.1 and drops significantly after that value. The main disadvantage of the Jaccard method is that it is rather slow compared to the other methods, given that the new text must be compared against all positively annotated texts in order to determine its relevancy with them. However, since the text classification method is part of the social media monitoring pipeline that is triggered very regularly (usually around 1 second), it is not considered an optimal solution.

In the sequel, the methods using classifiers are evaluated. In all cases three classifiers are tested for a set of parameters, which can be found in Table 6. For the remaining parameters, default values are used. Moreover, regarding the methods using TF and TFIDF representation different  $n$ -gram values and  $\text{min\_df}$  values are considered during text vectorization. The  $\text{min\_df}$  value affects the size of the feature length since it ignore terms that have a document frequency strictly lower than the given threshold when building the vocabulary. Specifically,  $n$ -gram parameter can be either 1 or 2, while  $\text{min\_df}$  can be 0.0001, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01 or 0.02.

Classifiers	Parameters
SVM	Penalty parameter: 0.01, 0.1, 1.0, 2.0, 3.0, 4.0, 5.0 Kernel type: rbf, poly
Naïve Bayes	Additive smoothing parameter: 0.01, 0.1, 1.0
Random Forests	Number of trees in the forest: 10, 50, 100, 200, 500, 1000 Number of features used for best split: auto, log2, sqrt, None

Table 6. Classifier parameters

Table 7 and Table 8 contain the best results of the TF representation method for each different classifier for different text inputs for the datasets MediaEval and beAWARE correspondingly.

n-gram	Text input	Classifier	Precision	Recall	F-score
1	Without stop words	SVM	0,85938	0,83125	0,78740
		Naïve Bayes	0,71406	0,82500	0,74795
		Random Forest	0,90000	0,84886	0,81241
	Without stop words & with stemming	SVM	0,48438	0,75057	0,58546
		Naïve Bayes	0,60313	0,79659	0,68319
		Random Forest	0,76563	0,82727	0,76324
	DBPedia concepts	SVM	0,35781	0,70000	0,46450
		Naïve Bayes	0,50156	0,75000	0,59335
		Random Forest	0,66250	0,75852	0,66614
2	Without stop words	SVM	0,84688	0,82614	0,77986
		Naïve Bayes	0,74844	0,83011	0,76213
		Random Forest	0,89531	0,85227	0,81508

	Without stop words & with stemming	SVM	0,45156	0,74432	0,56226
		Naïve Bayes	0,61094	0,79602	0,68536
		Random Forest	0,75156	0,82727	0,75987
	DBPedia concepts	SVM	0,36406	0,70170	0,47023
		Naïve Bayes	0,50156	0,75114	0,59444
		Random Forest	0,65625	0,76136	0,66667

Table 7. Evaluation of TF representation method for MediaEval dataset.

n-gram	Text input	Classifier	Precision	Recall	F-score
1	Without stop words	SVM	0,86066	0,23299	0,36671
		Naïve Bayes	0,84712	0,47337	0,60735
		Random Forest	0,81124	0,53402	0,64407
	Without stop words & with stemming	SVM	0,87413	0,18491	0,30525
		Naïve Bayes	0,84851	0,45155	0,58943
		Random Forest	0,81081	0,54364	0,65087
	DBPedia concepts	SVM	0,91449	0,19379	0,31980
		Naïve Bayes	0,83220	0,36132	0,50387
		Random Forest	0,77309	0,46117	0,57772
2	Without stop words	SVM	0,85795	0,22559	0,35725
		Naïve Bayes	0,85014	0,44896	0,58761
		Random Forest	0,82090	0,52885	0,64327
	Without stop words & with stemming	SVM	0,87701	0,18195	0,30138
		Naïve Bayes	0,85514	0,42788	0,57037
		Random Forest	0,82659	0,50592	0,62767
	DBPedia concepts	SVM	0,91956	0,18602	0,30944
		Naïve Bayes	0,83795	0,35762	0,50130
		Random Forest	0,74461	0,47226	0,57796

Table 8. Evaluation of TF representation method for beAWARE dataset

The same applies to Tables Table 9 and Table 10 which contain the best results of the TFIDF representation method.

n-gram	Text input	Classifier	Precision	Recall	F-score
1	Without stop words	SVM	0,71120	0,82344	0,76322
		Naïve Bayes	0,76311	0,65938	0,70746
		Random Forest	0,73359	0,89063	0,80452
	Without stop words & with stemming	SVM	0,69727	0,43906	0,53883
		Naïve Bayes	0,77186	0,56563	0,65284
		Random Forest	0,76973	0,74688	0,75813
	DBPedia concepts	SVM	0,66462	0,33750	0,44767
		Naïve Bayes	0,69752	0,48281	0,57064
		Random Forest	0,69581	0,59688	0,64256
2	Without stop words	SVM	0,70572	0,80938	0,75400
		Naïve Bayes	0,76056	0,67500	0,71523



	Without stop words & with stemming	Random Forest	0,74443	0,88750	0,80969
		SVM	0,69825	0,43750	0,53794
		Naïve Bayes	0,77419	0,56250	0,65158
	DBPedia concepts	Random Forest	0,76056	0,75938	0,75997
		SVM	0,66875	0,33438	0,44583
		Naïve Bayes	0,70000	0,48125	0,57037
		Random Forest	0,66831	0,63594	0,65172

Table 9. Evaluation of TFIDF representation method for MediaEval dataset.

n-gram	Text input	Classifier	Precision	Recall	F-score
1	Without stop words	SVM	0,24186	0,60738	0,37727
		Naïve Bayes	0,50444	0,70267	0,62526
		Random Forest	0,52071	0,70940	0,63797
	Without stop words & with stemming	SVM	0,20488	0,59356	0,33144
		Naïve Bayes	0,48521	0,69613	0,61094
		Random Forest	0,52256	0,71086	0,63995
	DBPedia concepts	SVM	0,20118	0,60129	0,32910
		Naïve Bayes	0,32988	0,64552	0,47497
		Random Forest	0,71006	0,59608	0,63085
2	Without stop words	SVM	0,23151	0,60556	0,36597
		Naïve Bayes	0,48558	0,70104	0,61499
		Random Forest	0,55806	0,71395	0,65737
	Without stop words & with stemming	SVM	0,19970	0,59102	0,32442
		Naïve Bayes	0,46043	0,69194	0,59512
		Random Forest	0,54882	0,71158	0,65173
	DBPedia concepts	SVM	0,17825	0,59285	0,29854
		Naïve Bayes	0,37426	0,65504	0,51331
		Random Forest	0,70932	0,59572	0,63040

Table 10. Evaluation of TFIDF representation method for beAWARE dataset.

Regarding the word2vec methodology several runs were realised for different vector dimension (i.e. 100, 200, 300, 400, 500), words window (i.e. 2, 3) and training algorithm (i.e. 0, 1) parameters. Tables Table 11 and Table 12 contain the best results of the word2vec method for the MediaEval and beAWARE datasets correspondingly. For all cases, the highlighted rows are the ones with the best results for each table. The sizes of the of the corpora used are 6,600 records for the mediaEvalEnglishFloods\_corpus, around 830,000 records for the beAwareEnglishFloods\_corpus and 15,000 for the beAwareItalianFloods\_corpus. It is evident that for the case of the MediaEval dataset where two different corpus are used, the bigger corpus with around 830,000 records achieves better performance.

Text input	Corpus	Vector dimension	Words windows	Training algorithm	Precision	Recall	Fscore
text with	mediaEvalEnglishFI	100	3	1	0,75	0,745	0,751

stop words removed	oods_corpus				835	31	77
text with stop words removed	beAwareEnglishFloods_corpus	200	3	0	0,79341	0,82813	0,81040
text with stop words removed + stemming applied	mediaEvalEnglishFloods_corpus	100	3	1	0,76167	0,71406	0,73710
text with stop words removed + stemming applied	beAwareEnglishFloods_corpus	200	3	0	0,77647	0,82500	0,80000
DBPedia concepts	mediaEvalEnglishFloods_corpus	100	2	1	0,75455	0,77813	0,76615
DBPedia concepts	beAwareEnglishFloods_corpus	[100 – 500]	[2,3]	[0,1]	0,86667	0,02031	0,03969

Table 11. Evaluation of word2vec representation method for MediaEval dataset.

Text input	Corpus	Vector dimension	Words windows	Training algorithm	Precision	Recall	Fscore
text with stop words removed	beAwareItalianFloods_corpus	100	3	1	0,95855	0,06842	0,12772
text with stop words removed + stemming applied	beAwareItalianFloods_corpus	[100 – 500]	[2,3]	[0,1]	nan	0,00000	0,00000
DBPedia concepts	beAwareItalianFloods_corpus	[100 – 500]	[2,3]	[0,1]	nan	0,00000	0,00000

Table 12. Evaluation of word2vec representation method for beAWARE dataset.

The best runs from all tables along with information concerning the text representation parameters and the classifier parameters can be found in Table 13. After a careful observation, we can deduce that for the MediaEval dataset the best performing method is the TF method while for the beAWARE is the TFIDF method. The performance of the two methods is comparable within the same dataset, since for the beAWARE dataset the Fscore of TF method is 0,65 and the Fscore of the TFIDF method is 0,657 while for the MediaEval dataset the Fscore of TF method is 0,815 and the Fscore of the TFIDF method is 0,809. However, we observe a significant difference in the Fscore between the two datasets. This is most probably due to the special characteristics of the Twitter text such as the limited length, the use of non-standard terms and possible grammatical errors which require more advanced processing and representation techniques. However, for the current version of the

system and given that the dataset of interest is the beAWARE, we will proceed with using the TFIDF method for the text classification module, as the best performing examined approach. Finally, as far as the word2vec method is concerned, while it works satisfactory for the MediaEval dataset, it fails in the beAWARE dataset. A possible explanation for such low performance is a low quality of the corpus used for the word2vec representation as well a rather small size that does not cannot produce a good vector space. Thus, the use of a larger corpus of around 100,000 records will be examined in the next version.

Method	Dataset	Text Input	Parameters	Classifiers & Classifier parameters	Precision	Recall	Fscore
TF	beAWARE	text with stop words removed + stemming applied	n-gram = 1 min_df = 0,001 features length = 1313	Random Forest {Features num for best split: auto Number of trees: 1000}	0,81081	0,54364	0,65087
TF	MediaEval	text with stop words removed	n-gram = 2 min_df = 0,003 features length = 1068	Random Forest { Features num for best split: auto Number of trees: 200}	0,74804	0,89531	0,81508
TFIDF	beAWARE	text with stop words removed	n-gram = 2 min_df = 0,001 features length = 2762	Random Forests { Features num for best split: auto Number of trees: 200}	0,79968	0,55806	0,65737
TFIDF	MediaEval	text with stop words removed	n-gram = 2 min_df = 0,003 features length = 1068	Random Forest { Features num for best split: auto Number of trees: 500}	0,74443	0,88750	0,80969
word2vec	beAWARE	text with stop words removed	corpus = beAwareItalianFloods_corpus vector dimension = 100 words window = 3 training algorithm = 1	SVM {Penalty parameter: 5.0 Kernel type: rbf}	0,95855	0,06842	0,12772
word2vec	MediaEval	text with stop words removed	corpus = beAwareEnglishFloods_corpus vector dimension = 200 words window = 3 training algorithm = 0	SVM {Penalty parameter: 5.0 Kernel type: rbf}	0,79341	0,82813	0,81040

Table 13. Best parameters from TF, TFIDF, word2vec text classification methods.

### 3.7 Integration of the social media module into the beAWARE system

The social media monitoring framework is a module that runs constantly and makes use of real-time data. For reasons of demonstration, we have developed a single page web

interface<sup>8</sup> that serves to initiate the framework’s procedure at a specific time, for a specific dataset. As it can be seen in Figure 10: Screenshot of the demonstration tool

, the left side contains a set of simulated tweets that have been proposed by beAWARE user partners. These tweets were actually posted in Twitter and successfully crawled. To initiate the process of the social media module, the “Insert to DB” button has to be clicked. After the workflow (described in Section 3.1 ) is completed, the tweets that were estimated as relevant and were sent to BUS will be displayed on the right side. Clicking the “Empty the DB” button will clear the consumed tweets and it will be able to repeat the demonstration.

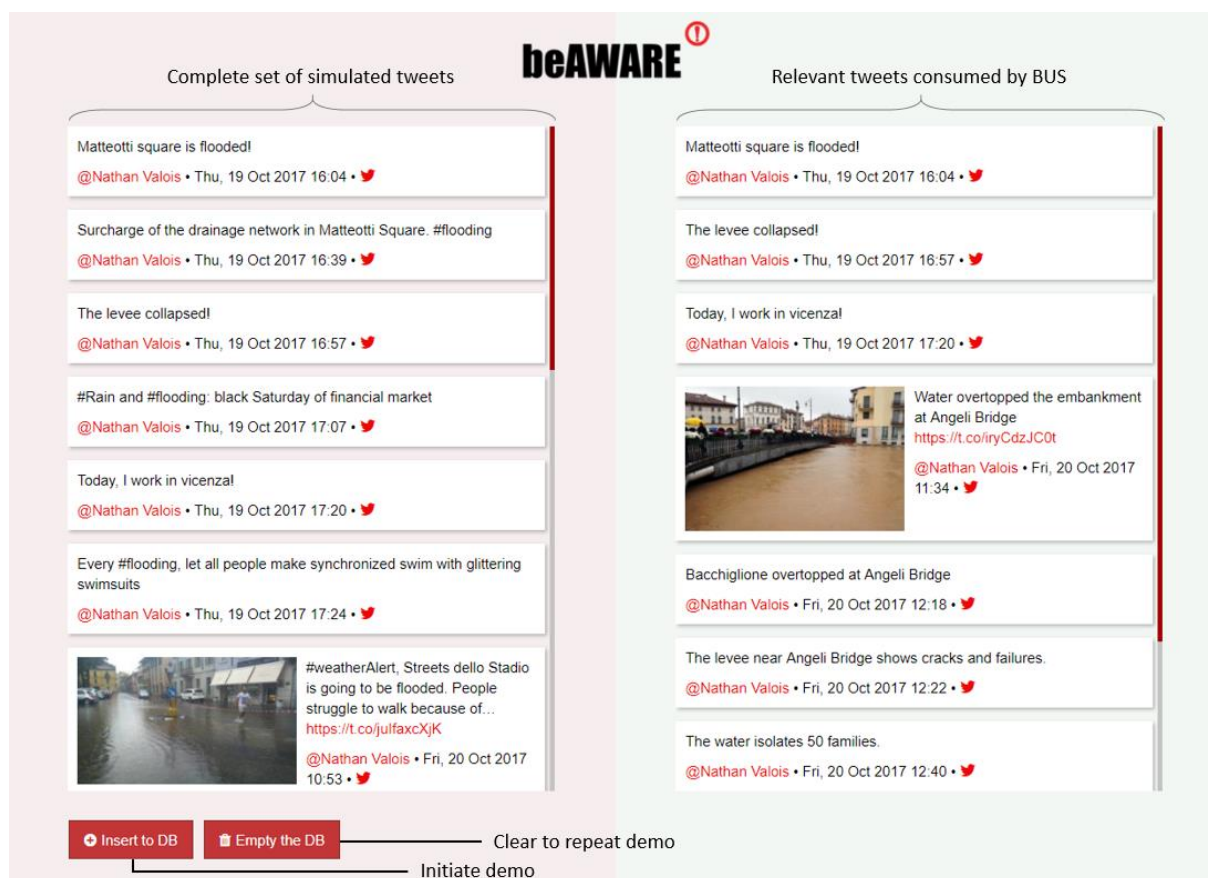


Figure 10: Screenshot of the demonstration tool

<sup>8</sup> [mklab-services.iti.gr/beAWARE\\_demo/](http://mklab-services.iti.gr/beAWARE_demo/)

## 4 MULTIPLE SENSING PLATFORMS

Situational awareness is crucial to make decisions. One source of information in a crisis situation is the sensor network that is available in the area of operation. For a decision support system to be able to use the data from these sensors, it needs a unified way to access not just the data from a multitude of different types of sensors, but also the meta-data about these sensors.

### 4.1 Sensor Types

There are several ways to classify sensors from the point of view of the beAWARE platform:

- **Online / Offline:** Online sensors perform measurements and transmit their readings to the system without human intervention. Offline sensors require human intervention before their readings become available to the system.
- **Remote / in-, ex-situ:** Remote sensors observe the subject from a distance, while in-situ and ex-situ sensors need to be located at the same location as the observed subject. In the case of in-situ measurements the sensor can be brought to the subject, in the case of ex-situ measurements the subject, or a sample of the subject, has to be brought to the sensor (usually, for an analysis in laboratory).
- **Machine-interpretable data / Non machine-interpretable data:** Machine-interpretable data can be processed by the system without human intervention. Non machine-interpretable data requires a human to interpret the data.

In the beAWARE platform, not all combinations of the above classifications are present. So far, the following four classifications of sensors have been identified to be most relevant for the beAWARE platform:

- Online, in-situ, machine interpretable (e.g. Weather stations, water level gauges)
- Offline, in-situ, machine interpretable (e.g. Manual water level measurements)
- Offline, remote, machine interpretable (Satellite / drone based sensors, etc.)
- Offline, in-, ex-situ, Non machine interpretable (Photos, observation notes, etc.)

Besides these actual sensors, mathematical models can also be seen as virtual sensors. Depending on the way the models are integrated into the beAWARE platform, they can be classified as online or offline. An example of an online, virtual sensor is the weather forecasts

software of the Finnish Meteorological Institute (FMI). Also the image and video analysis components can be seen as virtual sensors.

## **4.2 Data Types**

The classes of sensors described above can produce a range of different types of data that are stored in different ways. The main types of data relevant for the beAWARE project are:

- Time series
- Geospatial Coverages (one-off or series)
- Images and video

### **4.2.1 Time series**

Time series are measurements of the same observed property (OGC, 2011), for instance the temperature, taken at a more-or-less regular interval, for instance every hour. They can be taken by a sensor with either a fixed location (fixed-point), or by a moving sensor (moving point).

Fixed-point time series are generated by, for instance, the automatic water-level gauges, maintained by the Alto Adriatico Water Authority (AWAA) in the Vicenza region, or the national weather stations available throughout Europe. These sensors are located in-situ, and generate data at a fixed time interval.

Moving-point time series are generated by sensors that can be hand-held, or mounted on a vehicle or drone, and have a different location for each measurement. The GPS receivers of the first-responders are an example of this type.

Both types of time series are typically stored in servers that implement the OGC Sensor Observation Service standard (SOS; OGC, 2012), or the OGC SensorThings API standard (Liang, 2016; OGC, 2016).

### **4.2.2 Geospatial Coverages**

Geospatial coverages are typically generated by satellite-, aircraft- or drone-based sensors, or by mathematical models for spatial interpolation. They are images, where each pixel represents a measured (or processed) value of an observed property for a certain geographical region. Depending on the exact sensor type, a pixel can represent a region of a few square meters, up to many square kilometres. Typically, geospatial coverage data is served using a Web Map Service (WMS; OGC, 2006) or Web Coverage Service (WCS; OGC, 2012-2) like Geoserver (GeoServer, 2017).

#### 4.2.3 Images, video and text

This class covers all the non-machine-interpretable data types. Examples are photos, videos and text messages sent by first-responders or the public. The beAWARE platform will not directly be able to interpret these data, and first stores the files as they are. The image and video analysis components will extract as much information from the files as possible, but the end-user may still want to review the original. The system can present the availability of media files at relevant points in the platform, depending on the available metadata. For instance, if the geolocation of the recording is available, the platform can show the availability of the data as icons on a map. If the data consists of image files, the platform can display the image, with any automatic analysis results once they are available.

### 4.3 Pilots

The different pilots in the beAWARE project cover different types of extreme weather events and thus employ different types of sensors. Since the pilots are still in development, this initial list of relevant sensors is subject to change.

#### 4.3.1 Flood Pilot

The most relevant sensors for the flood pilot are the water-level sensors in the different rivers in the pilot area, and the weather stations recording precipitation, since they reflect the current situation. These are in-situ, on-line sensors that generate machine-interpretable time-series data and they are always available, not just in an emergency situation.

The FMI makes forecasts of the weather, and AWAA makes forecasts of the water-levels in the different rivers in the pilot area. The models used to create these forecasts can be seen as virtual sensors that generate machine-interpretable data. The water-level forecasts are time-series, while the weather forecasts are coverages that can be converted to time-series if needed. These models run automatically and are also available when there is no emergency situation.

First responders and the general public can send messages, with images and video, to the beAWARE system. When these data arrive in the system, the system cannot directly use this data. These raw data is therefore classified as off-line, remote, non-machine-interpretable, while the analysed results are on-line, virtual, machine-interpretable. On one side, this data is analysed first by the image/video/text analysis components to extract the important information and to make it available for the next analysis steps. This generated result can be classified as machine-interpretable data. On the other side, the raw data can be directly displayed to a user, to allow next to the automated evaluation a manually triggered action.

#### **4.3.2 Fire Pilot**

An important indicator for fire-risk is the current and predicted weather. Important sensor data for this pilot are therefore data from weather stations (on-line, in-situ, machine-interpretable, time-series) and the weather forecast (on-line, virtual, coverages/time-series, machine-interpretable). A high temperature, combined with a low humidity and little precipitation increases the risk of fire.

Next to the risk of a fire, the fast detection of existing fires is very important to allow a quick containment. A possible way to detect these is by using static cameras that constantly record the area of interest (on-line, remote, non-machine-interpretable) and analyse the data from those cameras using video analysis software (on-line, virtual, machine-interpretable).

Messages sent by citizens or first responders are also important, since the static cameras don't cover the overall area. This non-machine-interpretable data is handled the same way as in the Flood pilot.

#### **4.3.3 Heatwave Pilot**

Also for the Heatwave pilot the weather situation and the weather forecast are important sensor inputs, the same as for the other two pilots. Where for the Fire pilot a low humidity increases the risk of fire, for the heatwave pilot a high humidity increases the severity of a heatwave. The messages, images and videos sent by first responders and the general public are also handled the same way as in the Flood and Fire pilot.

Another relevant type of information is how crowded different public buildings that have air-conditioning are. This allows a systematic guidance of people to those places. Until now, it's unclear if pertinent data can be gathered using online sensors. However, it is possible to collect such data by using messages/images sent by first responders and citizens using the mobile app; like in all cases this offline, non-machine-interpretable data first needs to be analysed to extract machine-interpretable data.

### **4.4 OGC SensorThings API Data Model**

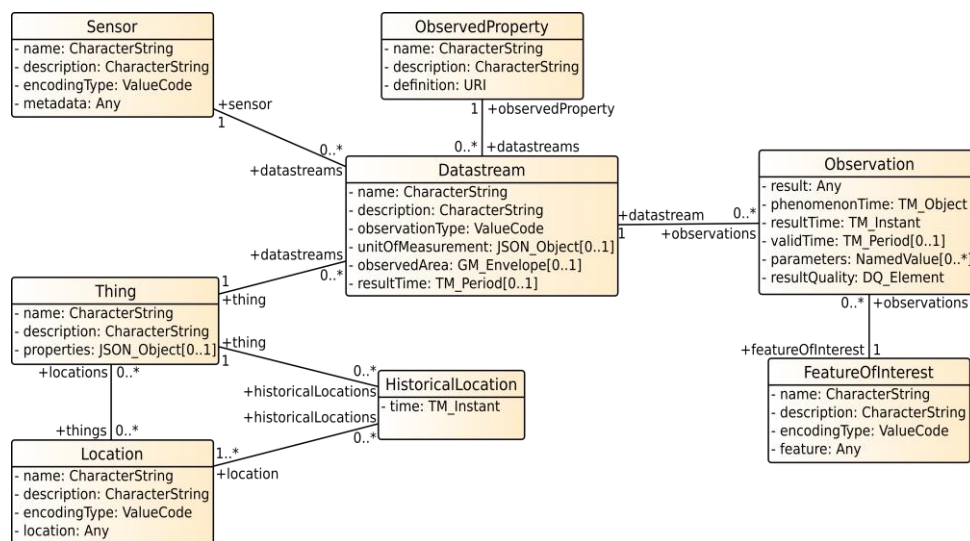
The model used to describe the sensors and their metadata in the beAWARE ontology is based on the data model described in the SensorThings API standard. The OGC SensorThings API standard (Liang, 2016; OGC, 2016) defines both a data model and a REST-API to access the data. It can be described as Sensor Web Enablement for the Internet of Things. It is a modern, light-weight REST API designed for storing and requesting sensor data, with advanced filtering options. The data model of this standard is based on OGC/ISO



Observations and Measurements model (OGC, 2011), which is a standardised model to describe a sensor.

The data model of the OGC SensorThings API consists of 8 entities, with their properties and relations (see Figure 11). The entities are:

- **Thing:** A virtual or physical object. Depending on the use case this can be the object being observed, such as a river or river section, or the sensor platform, such as a satellite or weather station.
- **Location:** The locations of Things. These can be geographic locations, encoded as points or areas, or symbolic locations, like “Crossing of road X and street Y”
- **HistoricalLocation:** the link between a Thing and a Location, with the time indicating when the Thing was in a certain Location.
- **Sensor:** The meta-data of a sensor that generates data. This could be a real sensor, or mathematical model generating a prediction.
- **ObservedProperty:** A property of the feature of interest that is being observed by a sensor. For instance, the water level or the air temperature.
- **Datastream:** a collection of Observations of one ObservedProperty, made by one Sensor, and linked to one Thing.
- **Observation:** a measurement made by a Sensor.
- **FeatureOfInterest:** The geographic area or location for which an Observation was made. This can be the same as the Location of the Thing, which is often the case for in-situ sensing. In the case of remote sensing, the feature of interest can be different from the location of the Thing, depending on what is chosen as the Thing. The feature is a geographical point or a polygon encompassing an area or volume, usually encoded in GeoJSON.



**Figure 11: the OGC SensorThings API data model**

The relations between these entities are also defined by the data model. Most relations are one-to-many: An Observation must have one FeatureOfInterest and one Datastream, while a Datastream and FeatureOfInterest can have zero or more Observations. A Datastream must have one ObservedProperty, one Sensor and one Thing, while a Thing, ObservedProperty and Sensor can have zero or more Datastreams. A HistoricalLocation must have one Thing, while a Thing can have zero or more HistoricalLocations.

The relations of Location are a bit more involved: a Thing can have zero or more Locations, but these Locations must all be different representations of the same physical location (e.g. a geospatial location, represented by GPS coordinates, and a symbolic location). A Location can have zero or more Things.

Each time a Thing is linked to a new Location (or set of Locations) a new HistoricalLocation is generated that tracks the time when the Thing was at this Location. A HistoricalLocation also has the restriction that if it has more than one Location, these Locations have to be different representations of the same real-world location.

When applied to beAWARE, for example to a water-level gauge in a river section, the Thing could be the river section of which the sensor is measuring the water level. The Thing would have a location, with a polygon describing the layout of the river section. Since river sections usually do not move much, there would be only one HistoricalLocation. The Sensor entity would describe the exact properties of the water-level gauge, like brand, type and accuracy. The ObservedProperty would be named “water level” and contain an exact reference to the water level entry in the knowledge base. For this set of Thing, Sensor and ObservedProperty there would be a Datastream, grouping the water-level observations for this sensor in this river section. Each value measured by the sensor would be stored as an Observation. Since

the sensor is static, each Observation is linked to the same FeatureOfInterest, which has the exact coordinates of the Sensor.

An important feature of the OGC SensorThings API is that it is possible to request data from related entities in a single query. For example, in a single request, one can fetch a set of Things, including the Datastreams belonging to those Things, and for those Datastreams the ObservedProperty, and the last Observation. This makes it very easy to write data visualisation tools, since it is possible to fetch all relevant data in one request, instead of having to make many separate, asynchronous requests.

For a temperature sensor located at the same spot as the water-level gauge, the same Thing, Location and FeatureOfInterest entities would be used. Only new Sensor, ObservedProperty and Datastream entities would need to be added.

## **4.5 Mapping the sensors onto the Ontology**

In order to make the beAWARE platform able to offer a coherent view of all data available for a given vulnerable object (like living being, infrastructure or possessions), the platform needs to know which data is available and how it is relevant for the vulnerable object. To achieve this, all information about vulnerable objects and their exposed risks have to be described in the beAWARE ontology, as will be described in detail in D4.2 - Semantic representation and reasoning. In order to enable the capability of the platform to combine information about the vulnerable objects with sensor data, the information related to a sensor has to be mapped to the same ontology. At this point, we would like to mention that only the sensor metadata is mapped to the ontology. This allows referencing from the ontology to the SensorThingsAPI server. The measurements themselves are stored inside the SensorThingsAPI server and are accessible through this relation.

The representation of sensor data and metadata in the ontology requires that the entities of the SensorThingsAPI are mapped to concepts in the ontology and are linked to the other relevant concepts. The result of this mapping is that most of the entity types in the data model of the SensorThingsAPI are represented by a concept in the beAWARE ontology. Since the ontology is still being developed, the exact mapping might still change.

The concept “Thing” in ontologies is reserved, meaning that it is impossible to name a concept “Thing”. Therefore, the entity “Thing” in the SensorThings API is mapped to the concept “Asset Representation” in the ontology. An Asset like a river is monitored by a SensorThing. The location of a “Thing” in the SensorThingsAPI is represented by the concept with the same name in the ontology. Historical locations won’t be used in our case. The entities “ObservedProperty” and “Datastream” are represented by the “Parameter” and

“DataSet” concepts respectively. Since the ontology will not store any observation itself, those measurements are reference by the “Service Endpoint” concept, which provides an URL to the datastream in the SensorThingsAPI server. Those references allow an easy navigation between the ontology and the data in the SensorThingsAPI server.

#### **4.5.1 Time series**

This type of data and its metadata can be directly modelled according to the SensorThings API data model. For this type of data, only the metadata (such as Sensor type, observed properties, etc.) is stored in the ontology. The observations themselves are not stored in the ontology, but in a separate server that implements the SensorThings API. Since this server will also have to store metadata as specified by the SensorThings API, there will be some duplication of metadata between the ontology and the SensorThings API server. An example of time series data is the water level at some point in the river. It contains measurements of the past, the current value as well as the predicted value in the future.

#### **4.5.2 Geospatial Coverages**

For this type of data, only the metadata will be stored in the ontology, the data itself is usually made available through a Geoserver instance, using WMS or WCS. To link the metadata in the ontology to the data, the Observation instance in the ontology could contain a URL, which points to the data stored in the Geoserver instance. For example, the weather forecast matches this category.

#### **4.5.3 Images and Video**

These sensor data will be stored as files in the platform. Since these data come from mobile devices of the public and of first responders, and are not inherently geo-referenced, it is important to define a clear way to organise these data and add relevant metadata like location, who sent the file, and which tasks they are relevant for. Such files are represented by the media item concept in the ontology, which has a reference to the stored file in the platform. By analysing those media files with analytical components, machine-readable information is generated. This contains for example the detection of vulnerable objects inside the media files which is represented by the concept with the same name.

## 5 CONCLUSIONS

In this report, we have presented the first version of the social media monitoring tool, as well as the sensor data wrappers of the beAWARE platform. Regarding the social media information, we are connected to Twitter's Streaming API that allows real-time feeds and is preferable to continual requests from Twitter's REST API. Moreover, keyword-based filtering of tweets is the most fitting option to the framework's goal at this stage. The list of search keywords and accounts had to be reconsidered because some keywords had been delivering only unrelated content. At the storage of data, we indexed tweets using the JSON format provided by Twitter Streaming API, as it is found to be optimal for storing tweets in a Mongo database and allows interoperable solutions. However, we had to enrich it with additional fields to improve efficiency in the overall flow of data in the further stages of data analysis.

Experiments on two collections of data, involving textual and visual information has shown that the best performing method in the case of text classification is the use of the TFIDF method for text representation and Random Forests as classifier. The other methods that were tested were the TF method together with Random Forests, the word2vec with SVM and the Jaccard Similarity Coefficient. The Jaccard Similarity Coefficient performed rather well but was not selected eventually for the text classification module since it is rather slow compared to the others. Moreover, as far as visual classification is concerned, the DCNN-based features together with an SVM classifier were selected as the best performing method among other simple color-based or edge-based features.

Regarding the management of sensor data in the beAWARE platform, we described the different types of sensor data relevant to the beAWARE use cases, and the different approaches taken for storing these different types of sensor data. The initial plans for linking the sensor data to the ontology have also been laid out, but these need to mature more, and will be described in detail in future deliverables.

Future work includes expanding the import mechanisms for the different sources of sensor data used in the beAWARE pilots, adding mechanisms for threshold detection and automatic data processing to the sensor data platform. Future work in social media monitoring involves the addition of a multimodal clustering module to support the visualisation of incoming tweets in the PSAP, further exploration on the on the data fusion of visual and textual information and optimal text representation and feature extraction method. The plan is also to examine the enhancement of the crawling process with location-based search and burst detection for the identification of specific events.

## 6 REFERENCES

- Aggarwal, C.C. and Zhai, C. eds., 2012. Mining text data. Springer Science & Business Media.
- Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L., 2008. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3), 346–359.
- Chandrashekar, G., Sahin, F., 2014. A survey on feature selection methods. *Computers & Electrical Engineering*, 40 (1), 16-28.
- Daiber, J., Jakob, M., Hokamp, C., & Mendes, P. N., 2013. Improving efficiency and accuracy in multilingual entity extraction. In: *Proceedings of the 9th International Conference on Semantic Systems* (pp. 121-124). ACM.
- GeoServer, 2017, <http://geoserver.org/>, November 2017
- Imran, M., Castillo, C., Diaz, F. and Vieweg, S., 2015. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)*, 47(4), p.67.
- Jégou, H., Douze, M., Schmid, C. and Pérez, P., 2010. Aggregating local descriptors into a compact image representation. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 3304-3311). IEEE. San Francisco, CA.
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., 2014. Caffe: Convolutional architecture for fast feature embedding. In: *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678). ACM.
- Jun Yan, 2009. Text Representation. *Encyclopedia of Database Systems*, 3069-3072
- Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems* (pp. 1097–1105).
- Liang, S.; Huang, C. & Khalafbeigi, T., 2016. OGC SensorThings API, Open Geospatial Consortium: Wayland, MA, USA.
- Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 91–110.
- Markatopoulou, F., Mezaris, V., & Patras, I., 2015. Cascade of classifiers based on binary, non-binary and deep convolutional network descriptors for video concept detection. In: *Proc. of the IEEE International Conference on Image Processing 2015* (pp. 1786–1790).

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

OGC, 2006, OpenGIS Web Map Server Implementation Specification, ISO 19128, <http://www.opengeospatial.org/standards/wms>

OGC, 2010, OpenGIS Web Feature Service 2.0 Interface Standard, ISO/DIS 19142:2010, <http://www.opengeospatial.org/standards/wfs>

OGC, 2011. Observations and Measurements, ISO/DIS 19156:2011, <http://www.opengeospatial.org/standards/om>

OGC, 2012. OGC Sensor Observation Service Interface Standard, <http://www.opengeospatial.org/standards/sos>

OGC, 2012. OGC WCS 2.0 Interface Standard, <http://www.opengeospatial.org/standards/wcs>

OGC, 2016. OGC SensorThings API Part 1: Sensing, <http://www.opengeospatial.org/standards/sensorthings>

Perronnin, F., Sanchez, J., Mensink, T., 2010. Improving the Fisher kernel for large-scale image classification. In: 11th Eur. Conf. on Computer Vision: Part IV (pp. 143-156). Springer-Verlag.

Pittaras N., Markatopoulou F., Mezaris V., Patras I., 2017. Comparison of Fine-Tuning and Extension Strategies for Deep Convolutional Neural Networks. In: Amsaleg L., Guðmundsson G., Gurrin C., Jónsson B., Satoh S. (eds) MultiMedia Modeling. MMM 2017. Lecture Notes in Computer Science, vol 10132. Springer, Cham

Qiu, G., 2002. Indexing chromatic and achromatic patterns for content-based colour image retrieval. Pattern Recognition, 35, 1675-1686.

Selvaperumal, P., & Suruliandi, A., 2014. A short message classification algorithm for tweet classification. In: 2014 International Conference on Recent Trends in Information Technology (ICRTIT) (pp. 1-3). IEEE.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CORR, arXiv technical report.

Song, G., Ye, Y., Du, X., Huang, X., & Bie, S., 2014. Short text classification: A survey. Journal of multimedia, 9(5), 635-644.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Vedaldi, A. and Lenc, K., 2015. Matconvnet – convolutional neural networks for matlab. In: Proceeding of the ACM International Conference on Multimedia.

Vijaymeena, M. K., and Kavitha, K., 2016. A Survey on Similarity Measures in Text Mining. Machine Learning and Applications: An International Journal, 3 (1), 19-28.