Start date of the project: **October 1, 2013**
Duration: **30 months**
Organization name of lead contractor for this deliverable: University of São Paulo (USP)
Executive Summary

This document presents deliverable D3.2.2 (Data Analysis Method Description 2) of project FP7-614154 | CNPq-490084/2013-3 (RESCUER), a Collaborative Project, supported by the European Commission and MCTI/CNPq (Brazil), which aims at supporting emergency organizations by crowdsourcing information in crisis situations. Full information on this project is available online at http://www.rescuer-project.org.

Deliverable D3.2.2 provides the second iteration of RESCUER Task 3.2 (Data Analysis) that is concerned with the implementation of the module for automatic analysis of image, video, and text data. The module shall receive data previously prioritized by another module under development in Task 3.1, and it will produce data (labelled images/videos/text) to be aggregated at incident level for the Emergency Response Toolkit (ERTK) of Work Package 4.

List of Authors
Agma Juci Machado Traina – USP
Andreas Poxrucker – DFKI
Glauco Vitor Pedrosa – USP
Gustavo Blanco – USP
Jonathan da Silva Ramos – USP
Jorge Alfonso Kurano – UPM
Jose Manuel Menendez – UPM
María Luisa Arroyo Noblejas – UPM
Jose Fernando Rodrigues Junior – USP
Letícia Pereira Soares Avalhais – USP
Mirela Teixeira Cazzolato – USP
Willian Dener de Oliveira – USP

List of Internal Reviewers
Konstantin Holl – Fraunhofer
Tassio Vale – UFBA
Renato Novais – UFBA
Karina Villela – Fraunhofer
Contents

1 Introduction ..................................................................................................................................... 5
   1.1 Purpose .................................................................................................................................... 5
   1.2 Partners’ Roles and Contributions........................................................................................... 6
   1.3 Document Overview ................................................................................................................ 6

2 Image Data Analysis Component .................................................................................................... 7
   2.1 Identifying Fire with Fire Relevant Regions in Images ............................................................ 7
   2.2 Identifying Smoke with Smoke Colour Description in Images ................................................ 9
   2.3 JSON Image Analysis Content ................................................................................................ 10
   2.4 Identifying Duplicate Images ................................................................................................. 11

3 Video Data Analysis Component ................................................................................................... 13
   3.1 Fire Event Detection in Videos: Overview and Concepts ...................................................... 13
      3.1.1 Problem Formalization ...................................................................................................... 14
      3.1.2 Description of the Proposed Method ................................................................................ 15
      3.1.3 Accuracy Analysis .............................................................................................................. 17
      3.1.4 Time Performance Analysis ............................................................................................... 19
      3.1.5 JSON Video Analysis Content ............................................................................................ 19
   3.2 Crowd Density Estimation ..................................................................................................... 20
      3.2.1 Specification ...................................................................................................................... 21
      3.2.2 Conceptual architecture .................................................................................................... 22
      3.2.3 Results ............................................................................................................................... 22
      3.2.4 Preliminary Analysis of Results .......................................................................................... 29

4 Text Analysis .................................................................................................................................. 30
   4.1 Goals ...................................................................................................................................... 30
   4.2 Model of the Text Analysis Component ................................................................................ 30
      4.2.1 Specification ...................................................................................................................... 30
      4.2.2 Conceptual Architecture .................................................................................................... 32
      4.2.3 Multi-Language Support .................................................................................................... 36
   4.3 Updates on the Text Analysis Pipelines ................................................................................. 37
      4.3.1 Pre-processing ................................................................................................................... 38
      4.3.2 Gazetteer Lookup .............................................................................................................. 40
      4.3.3 Semantic Information Extraction ...................................................................................... 41
1 Introduction

In the context of the Rescuer project, this task aims to timely support decisions by performing automatic processing of image, video, and text. The biggest issue considered here is the overloading of information that might reach the human experts in the command centre. As for example, hundreds of text messages, dozens of images, and too many minutes of video. In the first iteration of this task, presented in document D3.2.1, we introduced the technologies that would be used to implement the expected solutions. In this document, we present details of the implementations that are being integrated to the project.

Currently, we are able to analyse image, video, and text according to software designed and implemented in a modularized set of solutions. Upon new functionalities, the modules can be updated without demanding the system to be rebuilt, simplifying the addition of improved or new features. The integration to the whole system occurs by means of the JSON standard, communicated through the RabbitMQ technology: an architecture vastly discussed and described until now in the project.

1.1 Purpose

According to the RESCUER project, the Work Package 3 – Data Analysis Solutions has four parts: Task 3.1 – Data Fusion and Filtering, Task 3.2 – Data Analysis, Task 3.3 – Data Integration, and Task 3.4 – Data Usage Control. This document concerns Task 3.2, whose goal is to provide multimedia descriptions of crisis situations by means of analytical techniques that deal with image (in charge of USP) aiming at detecting fire and smoke, video (in charge of USP and UPM) also aiming at detecting explosion and crowd, and text (in charge of DFKI). The goal is to monitor fire, smoke, explosion, and crowd as detected from images, videos, and texts in order to assist control centres to identify the places, the time, the victims, and the circumstances of crises. Such monitoring may improve decision-making by augmenting the amount of useful information available for rescuer teams.

More specifically, Task 3.2 has three deliverables:

- D3.2.1 [month 12]: design of the image, video, and text modules; and the definition of how the filtered data is aggregated;
- D3.2.2 [month 20]: second version of the modules, along with the first prototype in preparation for integration with the RESCUER solution;
- D3.2.3 [month 28 according to the original plan, month 36 according to the current plan]: final version of the modules, with a validated prototype prepared for system integration.

Task 3.2, hence, is contextualized in the goals of the RESCUER project concerning automatic data analysis. The goal of the task is to build a data analysis module that should be able to assist the Emergency Response Toolkit (ERTK) by significantly improving situation awareness in a timely manner. The module shall receive data previously prioritized by another module under development in Task 3.1, and it will produce data (labelled images/videos/text) to be aggregated at incident level for the Emergency Response Toolkit (ERTK) of Work Package 4. At the time of this deliverable, the prioritisation and emergency state-related components have not been finalized, and information is
provided to the DAS directly from the Mobile Crowdsourcing Solution. However, this deliverable includes the specification of the emergency state information.

1.2 Partners’ Roles and Contributions

The partners involved in this task are UPM, USP and DFKI. UPM is concerned with video (crowd density), USP with image and video (fire and smoke), and DFKI with text analysis techniques. Each partner involved in this task will instantiate analysis techniques taking into consideration the need for information identified in Task 1.1 (Requirements Engineering) and the type of data to be analysed. Fraunhofer and UFBA have written Section 5 of this document.

1.3 Document Overview

D3.2.2 presents the techniques and architecture for the data analysis modules. The document is structured as follows:

- Section 2 presents the image analysis component;
- Section 3 discusses the video analysis component;
- Section 4 is focused on the text analysis component;
- Section 5 specifies the emergency state information to be considered in the next version of the data analysis components; and
- Section 6 provides the conclusions of this deliverable.
2 Image Data Analysis Component

Figure 1 shows the image analysis component and its communication to the other RESCUER architectural components. One can see the central role of the RabbitMQ platform, the HBase database, and the Amazon S3 media repository. The HBase database is contacted via RabbitMQ, while the Amazon services are contacted straight by the Image Analysis component.

![Figure 1: Analysing individual images](image)

In the next subsections, we explain the algorithms implemented in the Image Data Analysis Component.

2.1 Identifying Fire with Fire Relevant Regions in Images

The current algorithm to provide this functionality is the BoWFire approach. This algorithm was proposed in the paper:
We summarize the essential information about this functionality as follows:

- **BoWFire Method**
  - Status: Finished, integrated in the Image Analysis module.
  - Goal: The goal is to detect the regions of fire on still images using colour and texture information.
  - Brief description: The method consists of three steps: Colour Classification, Texture Classification, and Region Merge. The first step classifies the pixel colour as fire or non-fire through a Naive-Bayes classifier. The second step extracts Regions of Interest (ROI) of the image using a superpixel algorithm and then extracts texture features (LBP) of each region of interest (ROI). To classify the texture features as fire or non-fire regions, BoWFire uses the k-Nearest Neighbors (kNN) classifier. The first two steps occur in parallel to produce images in which fire-classified pixels are annotated. Then, in the third step, the outputs from both classifications are merged into a single output image containing only the fire regions. One of the advantages of the method BoWFire is the capability to dismiss false-positives. This way, it is possible to discard images with fire-like regions, such as sunsets, and red or yellow objects.

BoWFire includes two previous works related to the FFireDt algorithm. The latter is detailed in a paper and in a book chapter, as follows:


  ⇧ [http://rd.springer.com/chapter/10.1007%2F978-3-319-29133-8_2](http://rd.springer.com/chapter/10.1007%2F978-3-319-29133-8_2)

Other intermediate results have been developed and published to support the main functionalities, which are based on content-based retrieval techniques (features extraction, distance functions, metric spaces, metric access methods, and k-nn query and classification):
2.2 Identifying Smoke with Smoke Colour Description in Images

The current algorithm to provide this functionality is the SmokeBlock approach. This algorithm was proposed in the paper:

We summarize the technique as follows:

- **SmokeBlock Approach**
  - Status: Finished, integrated in the Image Analysis module.
  - Goal: The goal is to detect the regions of smoke in still images using colour and texture information.
  - Brief description: The SmokeBlock approach consists of three steps: (i) first, we extract superpixels from the input images. Superpixels are regions that present a similar (homogeneous) color distribution. Then, in the feature extraction step (ii), we extract information regarding texture and colour from each superpixel, generating one feature vector for each one of them. Finally, in the classification step (iii), each superpixel is classified as "smoke" or "not smoke", using the Naive Bayes classifier. Based on this information, the SmokeBlock approach builds a segmented image, composed by the superpixels classified as smoke. The RESCUER requirement states that the smoke colour should be informed. This is addressed by returning the mean colour of each smoke region segmented by SmokeBlock, which is labelled as "Dark Gray", "Light Gray" or "Other".

### 2.3 JSON Image Analysis Content

As already mentioned, the integration to the system uses the JSON format. Following, we describe the JSON record used by the image analysis component, so that other tasks can use this document as a guide:

```
ImageAnalysis_crowdMessages
{
    "reportTimestamp":"2014-07-04T12:48:06.4147832+02:00",
    "timestamp":"2016-01-13 13:37:58.469",
    "information":{
        "what":"Smoke and Fire",
        "whatId": "3",
        "reliability":"0.5",
        "MBR": [ [1,20,23,11], [2,30,23,112] ],
        "smokeColor": "Light Gray",
        "smokeColorId": "0",
        "observations":"vazio",
    },
    "duplicates": "0",
    "location":{
        "altitude":0,
        "longitude":8.542810682862173,
        "latitude":47.3742753222995
    }
}
```
Details of the fields:

- reportTimestamp: timestamp from the report;
- timestamp: timestamp of the current process;
- what: textual result of image analysis;
- whatId: numeric result of image analysis; the numbers used were:
  - 0 – Nothing
  - 1 – Smoke
  - 2 – Fire
  - 3 – Smoke and Fire
- reliability: reliability of the current analysis; a number between zero and one;
- MBR: regions of interest of the analysed image;
- smokeColor: smoke colour in textual format;
- smokeColorId: smoke colour represented by an id; the numbers used were:
  - 0 – Light Gray
  - 1 – Dark Gray
  - 2 – Other
- observations: possible observations.
- location: set of values of GPS from the original report;
- reportIdentifier: report id;
- quality: quality of the analysis;
- analysedElementIdentifier: image id;
- resolution: resolution of the image;
- comments: any optional comments.

2.4 Identifying Duplicate Images

The aim is to identify images that are exactly the same (duplicates) and/or very similar (near duplicate). Duplicate images might occur due to network, software, hardware, or, even, usability problems, leading the user to upload the same image more than once. Near duplicate images might occur due to sequential shots of the same scene, or due to multiple users observing the same situation from the same perspective. The identification of duplicate images might reduce the network traffic to the ERTK component, the processing load and the information to be interpreted by human specialists. Duplicate and near-duplicate image processing aids the image analysis component, avoiding unnecessary processing of duplicated images.
We perform duplicate and near-duplicate image identification based on their content. We use algorithm pHash [14], which is an open-source algorithm that has shown promising results. In addition, pHash is fast when compared to other content-based methods. In the following, we summarize the current status of this task:

- Status: Finished and integrated in the Image Analysis module.
- Goal: identify duplicate or near-duplicate images.
- Brief description: duplicated or near-duplicated images are summarized using the pHash algorithm. The method first converts the colour image into grayscale level and, then the image is shrunk into a 32x32 resolution. This procedure removes all high frequency and details of the image content. Therefore, resizing or stretching the image does not affect the computation of its pHash value, as required by the algorithm. To compare an image to another, the Hamming distance is computed between their pHash values. If the distance is lower than 3, the two images are duplicated or near-duplicated.

Currently, a new image is compared to all the previous images that arrived in the system. In the next iteration, the comparison shall occur only within the scope of the same incident or report.
3 Video Data Analysis Component

The requirements for Video Data are implemented in the corresponding component. Note that in the current version, the video analysis workflow is analogous to the image analysis’ workflow. However, it is important to highlight that there is no communication between video and image analysis, since all the algorithms and internal processes are distinct. Figure 2 shows this component and its communication with the other RESCUER architectural components. The video analysis is being conducted by USP (fire detection, Section 3.1) and by UPM (crow density estimation, Section 3.2).

![Figure 2: Video analysis component](image)

3.1 Fire Event Detection in Videos: Overview and Concepts

Next, a brief description of the development is presented together with some details about methodology and its performance. We summarize the current status of this task as follows:

- Status: improving performance.
- Goal: Detect events related to fire on videos.
Brief description: Colour is the most representative feature used in fire and flame detection methods. The common approach for these methods relies on algorithms that perform in three steps: 1) selection of candidate regions in frames that have fire-like colour pixels, based only on colour information; 2) motion feature extraction from the candidate regions detected in step 1, on subsequent frames; and 3) classification of the motion feature. We proposed a colour model to perform spatial segmentation that generates the candidate regions for the motion analysis. Our colour model is based on the Hue Saturation Value colour space. After the spatial segmentation, we extract and classify the motion vectors to determine the segments of the video that contain fire.

Since the solution for fire identification has not been published yet, hence, not validated by academic peers, in the following, we provide brief details about the proposed technique for fire identification. The proposed method achieved accuracy up to 80%, which is higher than similar methods found in the literature. The final details of the implementation, as well as further experiments, are being conducted. The corresponding module is already part of the Rescuer system; it shall be updated as new versions of the software are completed.

There are several methods proposed in the literature to identify fire in videos. In general, most of these methods have satisfactory accuracy in answer if a fire scene is occurring in a video. However, there is a major constraint in the existing methods: they assume that the videos were acquired by a static camera. This assumption is a limitation that makes such methods not suitable to RESCUER, which relies on videos generated by mobile devices. Indeed, for these types of data, we have verified that camera motion is almost always present as a visual effect of filming with handheld cameras. In order to overcome this limitation, we have developed a fire detection approach for generic video content. Our main focus, different from the common works regarding fire events, is the detection in terms of segmentation, rather than only classifying cropped videos of an atomic event.

3.1.1 Problem Formalization

We denote a video \( V \) as a sequence of frames such that \( V = \{f_1, f_2, ..., f_L\} \) where \( L \) is the total number of frames. Each frame \( f_i \) is a unique representation of the spatial information in time, being associated with an image \( I_i \) and a timestamp \( t_i \), which is also dependent of the frames per second rate (FPS for short).

**Definition 1:** A segment \( s \) from a video \( V \) is such that \( s = \{f_i, f_{i+1}, ..., f_j\} \) is a sequence of consecutive frames from \( V \).

**Definition 2:** A temporal segmentation of a video \( V \) is a sequence of segments denoted by \( S(V) = \{s_1, s_2, ..., s_n\} \) such that each segment \( s_i \) must initiate before the next segment in the sequence. Therefore, let \( s_i \) and \( s_{i+1} \) be two consecutive segments and \( f_a \in s_i, f_b \in s_{i+1} \) be the first frame of each segment, respectively. Therefore, \( a \leq b \).

**Definition 3:** A semantic (or event) segmentation on video \( V \) is a temporal segmentation of \( V \), denoted by \( \varphi_e(V) \) such that each segment represents the event of interested \( e \) that is likely to be
occurring in the interval of each segment, being the result of an event detection task. Ideally, given $s_i$ and $s_j \in \phi_e(V)$, then $s_i \cap s_j = \emptyset$.

We introduced a semantic segmentation method focused on the detection of fire events. So for each video input $V$, $\phi_{\text{fire}}(V)$ is the result of our method representing the segments where fire was detected.

### 3.1.2 Description of the Proposed Method

We adopted the usual approach presented in correlated works that is first filter information from the spatial domain and then analyse the data from temporal domain. As discussed earlier, this is a well-established foundation to detect fire, because it is capable to reduce computational effort by selecting only regions of interest as potential candidates. In the following we describe the details of our method.

**Spatial segmentation of fire like colour regions**

The spatial segmentation is applied in every frame. The aim of spatial segmentation is to identify subset of pixels that are colour related to fire in the images from each frame. The use of colour information is based on the fact that fire is visually bright red-yellowish in general, which it allows the construction of colour models to captured such pixels at a low computational time cost. We reiterate here the importance of a very efficient spatial segmentation since this task is performed to improve the analysis of motion information, which is a more complex and costly step, besides more discriminant and sensitive to noise.

Our colour model was derived from the HSV colour space, which is a represented geometrically as a conical object. This colour space encodes the hue value into a wheel where the pure colour red is placed at $0^\circ$, green at $120^\circ$ and blue at $240^\circ$, for example, varying according to the amount of intersection of adjacent colours. The dimension S is saturation, encoded in the horizontal plane, representing the amount of mixture of each colours (red, green and blue). The last component is Value, encoded linearly on the vertical axis, representing the overall intensity of colour. Such colour encode scheme allows a more direct analysis of the correlation of each component.

In order to perform an analysis of the components of HSV space in fire coloured pixels, we have selected regions of fire from several images, cropping only inside fire region in order to visualise prospective relations between the components in pixel fire coloured. Figure 3 shows the visualisation of colour distribution for the ROIs showed in Figure 3.a. In Figure 3.b the colour distribution over hue and saturation components is presented. The relative amount of each colour is represented as spheres. In Figure 3.c it is showed two views of saturation and value components. From these views, we can see that value component tends to 4 when the hue is near to $60^\circ$ (yellow) and the saturation has almost an even distribution for the same hue. When hue is near to $0^\circ$ (red), the behaviour is the opposite: the saturation tends to be higher while value is more spread.

Based on this analysis, we proposed a colour model derived from HSV space that is composed by the inequalities below:

$$-10 \leq H \leq 65 \quad (1)$$
\[
\begin{align*}
\sqrt{(1-V)^2 + (1-S)^2} &\leq 0.8 - \frac{H^{0.3}}{100}, & \text{if } H < 0 \quad (2) \\
\sqrt{(1-V)^2 + (1-S)^2} &\leq 0.8 - \frac{H^{0.5}}{2000}, & \text{if } H \geq 0 \quad (3)
\end{align*}
\]

**Figure 3:** Visualization of fire ROIs in HSV colour space: a) ROIs used as input; b) Distribution along the Hue component wheel and Saturation axis; c) Views of Saturation and Value

Note that the values in the hue interval were expanded to allow the detection of a more relaxed range of possible fire pixels that may not be represented in the sample we used to visualise and derive the model. So, for each pixel \( p_i \in f_n \), if it satisfies the inequalities 1 and 2 (or 3 depending on H), then pixel \( p_i \) is a candidate pixel.

**Temporal segmentation of motion information**

Apply the spatial segmentation only is not sufficient to answer where there really is fire pictured in the frame. This is because the spatial segmentation model also captures pixels that are fire like coloured but are false positives. In this step, we use the region segmented previously to extract the motion information. As our aim is to produce an event segmentation, the motion flow between frames can contribute to it by describing the movement in candidate regions. So, given two consecutive frames with candidate regions \( f_{i-1} \) and \( f_i \), interested points are tracked and for each successful match, it produces the transition motion information \( T_{i,j} \), which is the histogram of oriented optical flow.

The transition motion histograms are classified using a k-Nearest Neighbour (KNN) classifier. We have trained a model using a dataset manually annotated of each frame regarding to the existence of fire in each frame. This is a simplistic and relatively fast approach to detect fire motion in transitions. Although this can be robust enough for videos with fixed camera, the videos with motion introduced by camera movements are negatively affected since this motion easily supress the motion around the interested region. This leads to an inaccurate representation of the motion in the ROIs, which in turn leads to classification error.
To attenuate this problem, our first attempt was using a background removal algorithm. This algorithm is commonly used in object tracking systems. The background is detected by performing a difference operation between the frames, where the moving regions are kept to the tracking. But this approach failed because for our videos, the critical segments with camera motion were detected as foreground.

A second approach was designed aiming at using compensation in the ROIs motion flow. We have proposed a technique for global compensation that used the complement region of the spatial segmentation, assuming it as the background and estimated the average motion for it. Then, for each optical flow vector in the candidate region, we subtracted the vector the average motion. The final representation of the motion feature is the histogram of oriented optical flow, \( H_{flow} \), quantizing into 32 bins the extracted directions of the new set of compensated flow vectors. In this representation, we do not consider the magnitude of the vectors, but only the orientations, since the objective is to model fire transition flow regarding the motion disorder over the candidate regions. After we computed the histograms flow, we use the Support Vector Machine (SVM) classifier, which has shown to provide more accurate classification results than the KNN classifier, to state whether or not a given motion pattern is a fire transition.

This led to an improvement in our classification results. The extra processing time in introducing the compensation motion was not high. Figure 4 shows three frames and its motion flow. The first frame is a ROI with fire where the motion on its edge is represented by the direction of the camera motion. The next two frames illustrate the motion flow in the dense approach.

![Figure 4: Motion flow detected from ROI and background camera motion](image)

### 3.1.3 Accuracy Analysis

There are no standard benchmark datasets for fire detection publicly available. Related works usually validate their methods using short video clips from static cameras. In order to evaluate the proposed method, we used two datasets with videos of uncontrolled scenes. The first dataset, FireVid, was obtained from YouTube through a crawler tool. This dataset contains 27 videos that were selected and manually annotated. The videos have diversity regarding to their spatial resolution, bitrate, duration and quality. For this dataset a total of 83,675 frames were given one of the following labels: (“fire”), (“smoke”), (“fire and smoke”), and (“none”). We used the “ignore” label to segments that have transition effects or a very fast movement that causes blur for example, since these frames are irrelevant and do not contribute to our analysis. The label “none” was assigned to every frame that does not contain fire nor smoke.
We also used the videos from the RESCUER dataset. The same methodology from FireVid was used to label the RESCUER dataset, which consists of 60 videos from emergency simulations at COFIC. The total number of frames manually labelled was 29,826. We compared our method using two baseline works from the literature: Celik et al. [1] and Di Lascio et al. [2].

Measures taken from the experiment are a weighted average of the results for the prediction of fire and the prediction of non-fire frames transitions. This means that the positive and negative scores are computed from the classification of each motion histogram, i.e., it takes into account not only one frame each time, but the flow information between the previous to the current frame. For each processed video, we computed the confusion matrix for the classification of all the motion features; we calculated three standard measures: Precision, Recall, and F1-measure.

The results obtained with the FireVid dataset are showed in Table 1. Considering the precision score, our method was about 13% more precise than the method of Celik, and nearly tied in precision when compared to the results of Di Lascio, with 89.16% and 89.17%, respectively. Our method has obtained the best recall score with 63%, being 3.37 times superior to Celik and 24% superior to Di Lascio. Regarding F1-score, SPATFIRE overcame both Celik and Di Lascio, with 74.3%, 30.48%, and 65.2%, respectively.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celik</td>
<td>79.16</td>
<td>18.87</td>
<td>30.48</td>
</tr>
<tr>
<td>Di Lascio</td>
<td>89.17</td>
<td>51.37</td>
<td>65.18</td>
</tr>
<tr>
<td>Our method</td>
<td>89.16</td>
<td>63.70</td>
<td>74.31</td>
</tr>
</tbody>
</table>

Table 2 shows the performance results of the three methods with respect to the experiment performed in the RESCUER dataset. The proposed method overcame both the concurrent approaches. Our method achieved a precision rate of 94.4%, while Di Lascio achieved the second best precision with 90.5%, and Celik achieved the lowest rate of 78.6%. The recall rate achieved by our method, 73.6%, was 37% higher than that of Celik and 31% higher than Di Lascio. By analyzing the F1-score results, one can confirm the overall better performance of our method, with 82.73%, against 69.2% obtained by Di Lascio, and 63.8% obtained by Celik.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celik</td>
<td>78.53</td>
<td>53.75</td>
<td>63.82</td>
</tr>
<tr>
<td>Di Lascio</td>
<td>90.44</td>
<td>56.09</td>
<td>69.24</td>
</tr>
<tr>
<td>Our method</td>
<td>94.40</td>
<td>73.62</td>
<td>82.73</td>
</tr>
</tbody>
</table>
3.1.4 Time Performance Analysis

We also collected time processing information in the experiments (average) to evaluate whether our method can be used for real time tasks or not. The setup of the machine used in the experiments was: processor of 2.5 GHz Intel Core i5, a memory of 8 GB (1600 MHz) under OS X System (10.11.4).

The FireVid dataset consists of videos varying resolutions from 320×240 to 600×336, FPS varying from 10 to 30 with different encodings. The execution time revealed an average rate of 2,120 frames processed per minute in this dataset. Considering one video with FPS=30, 1800 frames are exhibited per minute. The ratio between the time execution and video duration was, in average, 65%, with a maximum of 88%.

The RESCUER videos dataset has balanced amount of videos with low resolutions (320×240) and high resolutions (1920×1080). For the videos with higher resolutions and frame rates, the execution time was superior to the video duration. To attenuate this problem, the frames were resized so that the maximum dimension was at most 600 pixels. The resize operation, although has its cost, was effective to satisfactorily reduce the execution time without harming the accuracy of the method. Using the resize, the average ratio of execution in relation to the duration was around 86%. From this experimental analysis, we can conclude that our algorithm is suitable to be used for real time applications since the time to process each video was lower than the duration of the video.

3.1.5 JSON Video Analysis Content

Following, we describe the JSON record used by this video analysis, so that other tasks can use this document as a guide:

```
VideoAnalysis_crowdMessages
{
    "reportTimestamp":"2014-07-04T12:48:06.4147832+02:00",
    "timestamp":"2016-01-13 13:37:58.469",
    "information":{
        "what":"Smoke and Fire",
        "whatId": "3",
        "reliability":"0.5",
        "segments_smoke": [[1,20], [2,30]],
        "segments_fire": [[4,20], [11,233]],
        "observations": ".",
    },
    "location":{
        "altitude":0,
        "longitude":8.542810682862173,
        "latitude":47.3742753222995
    },
    "reportIdentifier": "113245",
}
```
Details of the fields:

- reportTimestamp: timestamp from original report;
- timestamp: timestamp of the current process;
- what: textual result of image analysis;
- whatId: numeric result of video analysis; the numbers used were:
  - 0 – Nothing
  - 1 – Smoke
  - 2 – Fire
  - 3 – Smoke and Fire
- reliability: reliability of the current analysis. A number between zero and one;
- segments_smoke: segments from video where smoke were detected;
- segments_fire: segments from video where fire were detected;
- observations: possible observations;
- location: set of values of GPS from original report;
- reportIdentifier: report id;
- quality: quality of the analysis;
- analysedElementIdentifier: image id;
- comments: any additional comments.

3.2 Crowd Density Estimation

Another key feature of the video analysis component is the density estimation of people in the scene. This density is obtained estimating the number of people frame by frame, adding all the estimates, and then dividing by the number of frames analysed. Thus, this procedure takes into account that the number of people changes among frames due to moving people and occlusions. It is also considered that there is no prior knowledge of the real dimensions of the scenario (due to the lack of information) and that there is no calibration of the mobile cameras used in the scenario of the incident. Moreover, people’s density may not be homogeneous and perspectives can be different, resulting in subjects appearing in the video at different distances with different sizes. We have considered an approach (which relates the aspect ratio of a person with the resolution of the image) to overcome these shortcomings and still obtaining a useful estimate of the crowd density. In this section we illustrate some of the challenges on density estimation together with some initial results of the solution adopted in the project.
3.2.1 Specification

The types of views that will be considered for video analysis of crowd aspects are frontal view and panoramic view. In the first case, if there are people ‘hidden’ behind the forefront, it will be impossible to determine the correct number of people in the scene due to total occlusions. In the case of the panoramic view it will be possible to estimate the number of people more accurately because of the expected fewer number of occlusions.

The approach that has been developed is specific for detecting and counting people at medium and close distances, because it is based on edge orientations and not in textures, as people in the far field usually appear blurred in the videos. Lighting conditions should be good enough in indoor and outdoor places (scenes recorded with daylight at central hours of the day, and preferably on a sunny day).

The scene can be a fixed or be a moving scene. In the latter case if the movement of the camera is high, probably different perspectives will be recorded, with many occlusions. In this case, the average of people over frames can be quite different, so it is recommended to have a fixed scene for a better estimation of the number of people in the scenario.

The solution for the detection of people in the video is based on the paper [15], which uses HOG (Histograms of Oriented Gradient) and SVMs (Support Vector Machines). In our context the challenging task is to detect and count people in crowded scenes with a mobile camera instead of using CCTV cameras as in the researches that have been conducted so far.

Resolution and video length

The video resolution used for make tests are varied: 320x240, 640x360, 960x540, 1920x1080, among others. The video length is established as ten seconds, due to scalability and manageability.

Connection to RabbitMQ and AMQP protocol

The video analysis component for crowd aspects depends on the RabbitMQ platform to download the videos which have been previously sent by the user of the mobile application. Videos are analysed and afterwards the information extracted is uploaded to the ERTK. To make these connections and to manage communications among them, a library (called 'messagebroker') has been implemented using the AMQP (Advanced Message Queuing Protocol) protocol.

JSON messages and cURL

JSON is a lightweight format for data exchange. The JSON message is the container where the information of the report and the information extracted from the analysis are stored. These JSON (JavaScript Object Notation) messages contain several data, including for example the Amazon URL where the videos recorded by the users are stored. This information is used by the component to download the video from the URL using cURL (command line tool and library for transferring data).

Libraries

The libraries used for the video analysis component for crowd aspects are: OpenCV, RabbitMQ, cURL, rapidJSON, AMQP and 'messagebrokerlib' (link with the RabbitMQ platform).
3.2.2 Conceptual architecture

The diagram in Figure 5 briefly shows how the video analysis component for crowd aspects is integrated with RabbitMQ. Basically, this analysis component uses a specialized library to connect to RabbitMQ.

![Figure 5: High-level architecture of the Video Analysis Component & Communication](image)

The video analysis component for crowd aspects is composed of the diagram blocks presented in Figure 6. The figure depicts the processing steps that are applied to every video data that enters the component.

![Figure 6: Block’s diagram of the video analysis component](image)

3.2.3 Results

In this section, we present some of the results achieved with the video analysis implementation. In particular, we report on two video sequences with frontal view and three video sequences with panoramic view.
The dataset used for testing the video analysis component for crowd aspects is composed of 98 videos of different qualities (with distinct resolutions, lengths, bit rates, frame rates and codecs). They are taken from near to a middle distance. Here we only show the results obtained for some of them.

The sequences consist of crowded people scenes and were used for preliminary check of the good functioning of the video analysis component for crowd aspects. These sequences range from a tiny crowd to a massive crowd (heavy occlusions). People appear really close together in some videos and more sparse in other video sequences. They are mostly outdoor scenes with different lighting (taking at night and during the day), and recorded with diverse camera angle views (top views, frontal views, etc.). People appear at distinct distances from the camera (walking, running, riding a bicycle, among others) and are taking different and diverse directions. Sometimes people appear blurred, and at times they appear really next to the camera, which provokes a total occlusion for some seconds.

We present below the tests conducted with the mentioned five videos sequences; other tests with more populated videos are being considered.

Thereupon some frames of the five sequences are shown in order to showcase how the component works. The red rectangles in the frames indicate detected people. The corresponding tables show what the density estimation does quantitatively: it detects people over the frames of a video and calculates the estimated number of people in each frame, after that the average number of people in the video is calculated, in addition to the median and the maximum and minimum number of people detected in the sequence.

Frontal view

Video sequence '022800513'

Frame 54

Frame 55

Frame 89

Frame 132
Figure 7: Video sequence of people crossing the street

As it can be seen, in the sequence of Figure 7, there are people walking on a zebra crossing, a person riding a bicycle, and people riding motorbikes who are correctly detected. Detection windows (red rectangles) represent people detected in the frame, sometimes as a single person and sometimes as a group of people if they are really close together. Sometimes some false positive detections appear and a person can be present in two different detection windows. There are also partial occlusions, but they do not prevent the detection of people, all the people are correctly detected.

<table>
<thead>
<tr>
<th>People average estimation</th>
<th>Median</th>
<th>Maximum number of people</th>
<th>Minimum number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>20</td>
<td>38</td>
<td>8</td>
</tr>
</tbody>
</table>

This table represents different values obtained from the video analysis component, first column tells the number of people estimated in average through all the video, the second one gives the median number of people obtained for the scene, and the last ones the maximum number of people found in the whole scene and the minimum number of people estimated in the scene respectively. As it can be seen the results are very similar to the ground truth (the people who really appears in the scene). The same table is shown for every sequence with its respective obtained values.

Video sequence '044171254'
In the sequence of Figure 8, there are people walking towards/from the camera. There are several occlusions due to the fact that this is a frontal view. People in the back cannot be seen (total occlusions) and that is why they cannot be detected logically either. There are also one false detection (frame 89 and 99) and detections of parts of the body (frame 49, frame 180). The person who is dressed in red in the left side is detected in some frames and not detected in other frames as he is in the border of the image and appears incomplete. In general, people are correctly detected, but there is a missing detection in frames 49 and 99.

<table>
<thead>
<tr>
<th>People average estimation</th>
<th>Median</th>
<th>Maximum number of people</th>
<th>Minimum number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>19</td>
<td>44</td>
<td>0</td>
</tr>
</tbody>
</table>

This table represents different values obtained from the video analysis component. The number of people estimated in average through all the video is close to the ground truth, the median number of people obtained for the scene gives a robust estimation, and the maximum number of people found in the whole scene and the minimum number of people estimated in the scene is similar to the maximum and minimum number of people who appears in the most populated and the less populated frames. In detail, the minimum number of people is zero due to a bus that crosses the view field of the camera and completely covers the scene in some frames.
Panoramic view

Video sequence '1778613'

Figure 9: Video sequence of people in a panoramic perspective

In Figure 9, the video sequence presents a relatively panoramic view. In this scene some people appear at a medium distance and some of them further at the back. As it can be seen, detection windows only appear in the areas where there are people (meaning there are not false positives), these red windows enclose individuals as well as groups of people who are very close together. In this sequence, very good results are achieved. People at the back and some at the medium distance appear blurred and partially occluded, but they are correctly detected. Some people appear in more than one detection window and there are only one or two false positives.

<table>
<thead>
<tr>
<th>People average estimation</th>
<th>Median</th>
<th>Maximum number of people</th>
<th>Minimum number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>78</td>
<td>157</td>
<td>43</td>
</tr>
</tbody>
</table>
In this table we show the values obtained for this video sequence, the people estimated in average and using the median are really close to the reality as it can be seen from the frames presented. In some frames the number of people raise till more than one hundred and in some of them the number of people decrease to dozens, these values correspond almost to the ground truth.

**Video sequence '3452204_031_c'**

Figure 10: Video sequence of people in a police occurrence

Figure 10 shows a video with a panoramic view. This is a scene with a lot of people running, fleeing and hiding, so people appear in a wide variety of positions and forms. Despite of these difficulties, people are correctly detected and the number of people in the video is correctly estimated. For instance, in frame 325 there is a woman lying on the floor and she is detected; and in frame 206 there is a crouched person and he is also detected. In frame 415 there are some false positives and some missing detections due to the complexity of the scene and the dim lighting.
This sequence shows good results regarding the number of people in average in the scene, and also in the maximum and minimum number of people estimated in the most and the least populated frames.

**Video sequence '879-38_l'**

In the panoramic view of Figure 11 there are a lot of people crossing a street in multiple directions. In this scene, there are several occlusions due to the crowd. Some people are included in two different detections and a few detections are missing, mostly in the borders of the image. People which appear with a lot of occlusions have been detected demonstrating the good performance of
the algorithm. Detections are really good, because people who appear in different perspective views, partially occluded and almost totally occluded, are detected.

<table>
<thead>
<tr>
<th>People average estimation</th>
<th>Median</th>
<th>Maximum number of people</th>
<th>Minimum number of people</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>28</td>
<td>54</td>
<td>14</td>
</tr>
</tbody>
</table>

The table above shows the results obtained for the video sequence '879-38_l'. In this table we can see how the algorithm estimates with precision the average number of people over the whole sequence and how it also calculates the maximum and minimum number of people detected in the frames of the scene with precision, taking into account that in most of the frames people does not appear entirely.

### 3.2.4 Preliminary Analysis of Results

As commented before a variety of videos have been tested in order to demonstrate the accuracy and the proper functioning of the implemented video analysis component for crowd aspects. These video sequences range from very crowded scenes to less crowded scenes. There are sequences with good lighting and dim lighting. There are sequences taken with frontal and perspective views, with total and partial occlusions, with people at different distances from the camera, with people walking and riding a bicycle, standing up and crouching or lying, head-on, sideways and backwards. There are scenes with the crowd walking in the same direction as well as in distinct directions, with people walking to the camera or in any other direction. There are scenes with people that appear blurred (especially at the back of the scene as for instance in Figure 9). There are sequences taken in indoor and outdoor places (artificial light and sunlight), despite indoor sequences not being shown here. And finally there are people moving and still (for instance Figure 8 and Figure 10), as well as video sequences taken with a moving or a fixed camera.

The algorithm has been tested in a wide variety of video sequences, ranging from simple to complex and in all of them the results achieved are remarkable (as it can be seen from the tables above). Henceforth, the results obtained attest the good functioning of the implemented video analysis component for people detection at close-up and middle distances.

There exist some open issues, such as people detected twice, body parts detected, false positives and missing detections. However, the prevalence of these issues is not so high, and they will be studied in depth to mitigate their effects in the final results.

Summing up, what the density estimation does is to detect people over the frames of a video and calculate the estimated number of people in each frame, after that the average number of people in the video is calculated, in addition to the median and the maximum and minimum number of people detected in the sequence. Furthermore, if there are more than four people per square meter the risk of congestion alarm is set to true. More detailed information related to the video analysis implementation will be given in deliverable D3.2.3.
4 Text Analysis

4.1 Goals

In this chapter, we present the progress made in the text analysis component. In summary, we will:

- present the conceptual model of the text analysis component and its contributions to the RESCUER platform (Section 4.2);
- present the current state of the text analysis pipelines for English and Portuguese and describe the subcomponents used for their realization (Section 4.3);
- report our progress towards the implementation of a deployable text analysis component which is integrated to the overall RESCUER platform prototype (Section 4.4);
- discuss the progress concerning the evaluation of the text analysis component (Section 4.5);
- give an overview of our work in the upcoming final period of the RESCUER project (Section 4.6)

4.2 Model of the Text Analysis Component

4.2.1 Specification

Conceptually, as shown in Figure 12, the text analysis component can be seen as a black box that gets a free text message from a user as an input (probably together with some metadata such as the time the message was created or the GPS position of the device when the message was sent – message 1), applies a series of pre-processing and processing steps to extract semantic content, and returns a structured representation of the relevant information (message 3.1).

In the context of disasters, the text analysis module extracts information that is important in every emergency (e.g., type of incident, number of injured people, incident location). It should be flexible enough to account for aspects, which are only relevant for certain types of incidents (e.g. type of chemicals leaked during an accident in an industrial park). Additionally, the module can use context information, i.e. what is already known about the incident, in order to look only for those aspects for which information is not available or not sufficient for example. The following subparagraphs list a number of outputs the text analysis module may provide:
Figure 12: Input and output of the text analysis component

Text normalization

The text analysis component transforms an input text into a normalized representation by replacing known abbreviations and “internet slang” (expressions people use in social media like Twitter or Facebook, or message services like SMS or WhatsApp). This applies especially to emergencies in which people are overwhelmed and stressed by the situation and in which they may care least about orthography. These abbreviations and slang expressions may be hard to read and interpret for somebody who is not used to them, which could take valuable time from the ERTK users in the command and control centre to “decode” the information in the text. By including normalization as a pre-processing step, the text analysis component can replace those abbreviations and slang expressions with the corresponding correct words increasing readability and understandability. This functionality does not depend on the current context or any other semantics of the report text, but is purely syntactic.

What, where, who, and when phrases

The extraction of what, where, who, and when phrases from texts is the most basic semantic analysis functionality of the text analysis component. Although these phrases are only simple plain text assigned with the type of information they contain, this may nevertheless be a useful feature. If visualised in different colours by human operators at the command and control centre for example, it
may speed up the recognition of relevant information in a certain message or make it possible to skip a message in which the analysis component could not identify any important aspects.

**Incident time**

A user reporting an incident may provide information about the time an incident happened. An example may be a phrase like an incident happened “five minutes ago”, which can be used to automatically correct the incident time assumed by e.g. the timestamp when the report was sent out. This information can be extracted from when phrases and will provide a more exact estimation of the time of the incident.

**Incident locations**

Similar to the time, the free text description may provide more detailed information about the location of the incident. In context of an industrial park for example, a user may observe that smoke is coming from “production hall A”. The text analysis module may in turn identify this phrase if the specific keyword (here “production hall A”) is provided in a keyword list (see 4.4.1, step 8) and may directly link it to a GPS position if e.g. a lookup table that maps building/area/region names to GPS coordinates is available.

**Number of injured and dead people**

An essential aspect of every emergency is the number of human casualties. In case a user reports about this aspect in free text, the text analysis module can extract the quantities in case they are explicitly provided. A quantity may thereby be a concrete number which can be directly extracted from the text and added to the analysis result or may be expressed as an (subjective) estimation like “some”, “a lot of”, or “many”. In the latter case, the respective term requires interpretation and only a discrete level like “Low”, “Medium”, and “High” can be provided as analysis result given a mapping of quantity keywords to these levels.

**Further incident-dependent Information**

Some information that could be extracted by the text analysis module only makes sense in the context of certain incidents. Identifying the colour of smoke or finding indications of the extend of a fire for example are valid in case a fire incident is considered, but do not make sense in case of a report about a fight during a football game. It is thus necessary to apply a suitable processing depending on the current context, e.g. the type of the incident. Additionally, it is important that the analysis result data structure can be easily extended to accomplish for new additional results.

### 4.2.2 Conceptual Architecture

In order to ensure a proper integration with the RESCUER system architecture, it is necessary to define a conceptual architecture of the text analysis component, which, in particular, specifies the interface of the component as well as input and output data structures. Figure 13 shows the conceptual architecture. With respect to the specification in the previous section, the text analysis component gets an instance of the CrowdMessage data structure as input (we chose the name here
CrowdMessage referring to the data structures defined in the RESCUER platform demonstrator in order to avoid introducing an unnecessary amount of data structure names), which has the following attributes:

```
CrowdMessage {
    identifier: Unique id of the report identifying the text to analyse
    timestamp: Timestamp of the point in time the report was sent
    location: GPS position the report was sent from
    freeTextDescription: Free text description of the incident
}
```

Similarly, putting the semantic information described above together into a common structured output representation yields a generic high-level data structure `AnalyzedResult` containing the results of text analysis as attributes:

```
AnalyzedResult {
    identifier: Reference id of the message the analyzed text belongs to. Taken from the corresponding CrowdMessage instance
    originalText: Raw text of the original CrowdMessage instance
    normalizedText: Text of the original CrowdMessage instance with abbreviations and slang expressions replaced by the orthographically correct word
    extractedWPhrases: {
        what: Plain text phrases about incidents and attributes
        where: Plain text phrases about locations
        who: Plain text phrases about involved person and their attributes
        when: Plain text phrases containing temporal information
    }
    incidentAttributes: {
        types: Discrete type(s) of the identified incidents which can be defined based on e.g. an underlying ontology
        timestamp: Incident time from free text or timestamp of original CrowdMessage, if no meaningful time information was contained in the text
        locations: GPS locations from free text or location from original CrowdMessage if no meaningful location information was contained in the text
    }
}
```
**injuredPeople**: Reported number of injured people. Can be either a concrete number or a discrete level e.g. “none”, “some”, “many” depending on what a user wrote in the message. In case nothing is reported in the text, the field remains empty.

**deadPeople**: Reported number of dead people. Can be either a concrete number or a discrete level e.g. “none”, “some”, “many” depending on what a user wrote in the message. In case nothing is reported in the text, the field remains empty.

*Further incident-dependent information of interest*

---

**Figure 13: Conceptual architecture of text analysis component**

Internally, the text analysis component consists of three types of subcomponents, which are executed sequentially when a new message to analyse comes in. First, the *Language Identification* subcomponent determines the language of the free text description of the message to analyse. This is explicitly necessary, as processing texts in different languages requires different processing resources like keyword lists, etc. If the language cannot be identified successfully, an empty report with an error code could be returned, for example. Otherwise, the message is processed by a suitable *Text Analysis Pipeline* subcomponent (EN=English, PT=Portuguese in Figure 13).

A Text Analysis Pipeline applies a series of processing steps (see Section 4.3 for more details) and creates annotations carrying semantic information, which are expressed in terms of an ontology. An ontology is a formal way of expressing knowledge about a certain domain by defining types (classes) of entities, possibly associated with attributes, and their interrelations. For RESCUER, the minimal domain scope that needs to be covered by an ontology is the what, where, who, and when information characterizing an emergency. A very simple example is shown in Figure 14. The so
created semantic annotations may in turn carry a number of attributes related to the respective classes defined in the ontology. With respect to the ontology in Figure 14, the semantic information contained in a phrase like “seven injured and 4 dead” for example could be expressed by two annotations \{who=victim, injured=true, count=7\} and \{who=victim, dead=true, count=4\}, respectively. The term injured for example is thereby assigned the class victim with attribute injured from the ontology. In a similar way, textual phrases containing time and date (“when”), location (“where”) and incident (“what”) information can be annotated in such a way that the final Information Extraction step can work on a formal representation of the underlying free text description. At the moment, we are investigating existing emergency ontologies to be integrated to the demonstrator implementation. In case no suitable one is available, we aim at defining our own one, which will be presented in the final deliverable.

In the Information Extraction step, it may be additionally be possible to use context information i.e. what is already known about the incident. Exploring possible ways to realise this will be part of our efforts during the last period of the project.

Figure 14: Very simple example ontology showing a description of an incident in terms of its type (“what”), location (“where”), involved persons (“who”), and time information (“when”)
Figure 15: (Simplified) technical architecture of the text analysis component

Figure 15 depicts a more technical view of the text analysis component. In this figure, the Information Extraction step from Figure 13 is assumed to be part of the Text Analysis Pipelines. The component offers an interface ITextAnalysisModule specifying a single method analyze which gets a CrowdMessage instance as input and returns an AnalyzedReport structure containing the extracted semantic information in a structured way. First, the Language Identification subcomponent determines the language of the free text description of the message to analyse. The message is then processed by a suitable Text Analysis Pipeline component calling its analyse method defined by the interface ITextAnalysisPipeline. As before, the method expects the message to analyse as input, performs the actual information extraction, and returns the analysis results in form of an AnalyzedReport structure. Using this architecture, it is easily possible to realize a multi-language text analysis component through the addition of pipelines implementing the ITextAnalysisPipeline interface for every relevant language. Adapting the processing steps of the pipelines (essentially, parts of the semantic pre-processing and information extraction – see Section 4.3) and extending the AnalyzedReport data structure allows the customization of the output to whatever attributes are important in a specific application of the RESCUER platform.

4.2.3 Multi-Language Support

In Deliverable D3.2.1, we presented a pipeline for processing English texts. Based on these descriptions, it should be easy to see that it is hardly possible for us to provide a general-purpose text processing pipeline which can process messages written in any arbitrary language. This is not possible for the reason alone that every language has its own vocabulary (effecting the gazetteer lists) and grammar (effecting POS tagging and chunking stages), which in turn influences the rules used during semantic annotation.

Instead, we contribute with a general architecture of a text analysis component. In this architecture, the actual processing pipeline is abstract in the sense that it is defined only by its interface (input: crowdmessage instances sent from users/the crowd; output: generic analysedresults structure as described in Section 4.2) with the actual processing pipeline highly

36
depending on the language under consideration. In the next section of this document, we will present our progress.

4.3 Updates on the Text Analysis Pipelines

In the previous Deliverable D 3.2.1, we presented the concept of a text processing pipeline exemplarily and explained some of the basic steps which are commonly applied during information extraction. Figure 16 shows the current sequence of processing steps, which we implemented in both the English and Portuguese pipeline. In this section, we aim to present our progress towards the realization of both analysis pipelines.

In summary, steps 1 to 8 (tokenizer, normalization, sentence splitter, number tagger, measurement tagger, POS tagger, chunker, gazetteer) are syntactic and semantic pre-processing steps adding annotations and tags, which are necessary for the final information extraction procedure in step 9 (semantic information extraction). The gazetteer lookup, in particular, is one of the key subcomponents as it introduces the actual semantics to the text, therefore linking detected keywords to entities and concepts of the underlying ontology.

As described in D 3.2.1, the English pipeline is implemented using the GATE framework [6]. We reuse ready-made components and models of the various GATE plugins whenever possible as they have been developed and tested for years using large amounts of training and evaluation data. For Portuguese texts on the other hand we use a Python library called NLTK (http://www.nltk.org/) that provides easy-to-use natural language processing interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, etc. Both pipelines have been successfully integrated to the RESCUER platform demonstrator (see section 4.5). The following sections will shortly explain the individual steps in more detail and will present the components we used for their realization.
4.3.1 Pre-processing

The first step of the pipeline is *tokenization*, which denotes the task of splitting text into logical text units called tokens. The corresponding processing component is called a *tokenizer*. Tokenization is applied to distinguish individual words, numbers, or punctuation marks in a text. In the English pipeline, we use the English *Twitter Tokenizer*, which is available in *GATE* as part of the *TwitIE* text processing plugin [3]. As the name implies, this tokenizer is optimized for Twitter texts written in English, but it can also be applied to texts written in general social media language.

The next step is *normalization* of the given text, which denotes the task of transforming the text into some canonical form. In our context, normalization is the task of identifying and correcting abbreviations as well as “internet slang” expressions, which are commonly found in social media and SMS language. For our implementation, we use the *Tweet Normalizer* of the aforementioned *TwitIE* text processing pipeline. It can deal with special tokens such as hash tags and emoticons (which are unlikely to appear) and, in particular, abbreviations which are typically found in English social media texts (e.g. “thru” -> “through”, “b4” -> “before”).
A sentence splitter identifies individual sentences in texts. In the English pipeline, we use the sentence splitter of the ANNIE plugin, which is based on finite-state transductions to deal with special cases like abbreviations (e.g. the dot in “Dr.” usually comes from the abbreviation of the title and does not indicate the end of a sentence). Sentence splitting is a necessary pre-processing step for the POS tagger.

As the name implies, a number tagger finds numbers in texts. This is necessary in order to extract quantitative information such as “5 injured people” or “one hundred meters away” from emergency messages. A suitable implementation for English is provided as a plugin in the GATE framework creating number annotations based on tokens identified by the tokenizer. It is capable of finding numbers composed of individual digits (e.g. “5”, “100”) as well as numbers written-out as possibly several words (e.g. “five”, “one hundred”).

The measurement tagger identifies measurement units in texts (“seconds”, “meters”) and combines them with the number tags added by the number tagger to create measurement annotations. Using the output of a measurement tagger, it is possible to obtain quantitative information from texts like “100 meters” or “5 minutes”. As before, we use a GATE plugin providing the required functionality. It is able to find most common physical units written out in various ways, for example “s”, “sec”, and “second” are all recognized as the unit second. Furthermore, it assigns the type of measurement, e.g. temporal, which is highly relevant when looking for temporal phrases in the actual information extraction step.

POS (“part-of-speech”) taggers determine the lexical class of each word of a sentence, which depends on its syntactical and morphological behaviour. Example classes, which are common in almost every language, are noun and verb. In European languages, other categories such as adjective, adverb, or preposition exist. Another factor influencing the part-of-speech of a word is the semantic context it is used in. Depending on the context, the English word “help” for example may either be a noun or a verb. In our current realization of the pipeline, we use the implementation of the Hepple POS tagger [4] available with GATE which is an adaptation of the rule-based Brill tagger [5]. It adds part-of-speech annotations in form of a category feature to the token annotations extracted in step 1. The tagger thereby assigns tags according to the Penn Treebank Project [7] which defines one of the most popular standards for POS tags.

Chunking or shallow parsing denotes the process of finding the constituents of a sentence, e.g. noun phrases or verb phrases. In contrast to parsing, it does not create a complete parse tree, i.e. it does not identify subject, predicate etc. of a sentence but only sequences of words that belong together and form a “noun phrase” for example. For our purpose, full parsing makes no sense, as we are only interested in finding constituent phrases. Therefore, the overall sentence structure is not required. Furthermore, chunking is less complex with respect to runtime than full parsing. The first is preferable in the RESCUER context, as messages should be processed as fast as possible. For our pipeline, we use the OpenNLP chunker [Apache OpenNLP] available as part of the OpenNLP plugin in GATE. The OpenNLP chunker is based on maximum entropy models with a well-trained model for English language being available in GATE.

In the Portuguese pipeline, we use the NLPnet POS tagger (http://nilc.icmc.sc.usp.br/nlpnet/). NLPnet (Natural Language Processing with neural networks), which performs part-of-speech tagging and semantic role labelling. Most of the architecture is language independent, but some functions were especially tailored for working with Portuguese. NLPnet uses the Punkt Sentence Tokenizer to split sentences of the text and the RegexpTokenizer to get all the words of the text. NLPnet POS
tagger also works as a "Number Tagger" and "Measurement Tagger". The "normalization" step is being developed following the algorithm proposed by [12] and, for that reason, it will not be explained in this document.

4.3.2 Gazetteer Lookup

Gazetteers perform look up of words or word phrases from a text from a list of predefined keywords and can assign an arbitrary set of annotations to identified keywords. They are commonly used to find names of entities such as names of people (e.g. “Peter”, “Andy” etc.) or places (e.g. “Allianz Arena”, “Frankfurt Airport”, “Empire State Building”), but can be applied to find any given word as long as it is defined within the list of keywords. The gazetteer lookup is one of the key steps of the text analysis pipeline, which introduces the actual semantics to texts. In particular, it provides the link between keywords spotted in a text and an underlying ontology. With respect to the ontology in Figure 14, for example, keywords like “fire”, “flames”, “burning”, or “burn” can all be mapped to the incident type “fire”.

To create the gazetteers, we use a keyword-based approach based on a free, large lexical database of words called WordNet (https://wordnet.princeton.edu/). Wordnet groups nouns, verbs, adjectives and adverbs into sets of cognitive synonyms called synsets, each one expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. In addition, Wordnet is available in multiple languages (e.g., Portuguese, Spanish, Italian, French, etc.) as part of a project called Open Multilingual WordNet (http://compling.hss.ntu.edu.sg/omw/). As starting point, we use a collection of "seed words" from a set of primary gazetteer lists (lists which have been defined manually) and then extend them using WordNet. At the moment, we have defined nine categories. They are: attack (e.g., terrorist_attack, bombard, strike, shoot, etc); explosion (e.g., explosion, bombing, blast, fulmination, etc.); fire (e.g., fire, burn, open_fire, flame); smoke (e.g., smoke, fume, steam); medical (e.g., medical_care, first_aid, heart_attack, breathless, etc.); miscellaneous (e.g., gas, gasoline, fuel, chemical, etc.); observations (e.g., patrol, confused, hopeless, desperate, etc.); person_affected (e.g., victim, casualty, crowd, survivor, etc.); and person_malicious (e.g., terrorist, attacker, bad_guy, gunman).

Given a list of "seed words", the next step is to find the proper meaning of each word. For example, the word "attack" has nine different meanings as a noun and three different meaning as a verb (see e.g. the online WordNet (http://wordnetweb.princeton.edu/perl/webwn). Then, it is chosen the first meaning as noun: "(military) an offensive against an enemy (using weapons)" and the first meaning as verb: "launch an attack or assault on; begin hostilities or start warfare with". The notation of the synset is "attack.n.01" and "attack.v.01", respectively, where "attack" is the lemma, "n" means noun, "v" means verb, and "01" refers to the first meaning.

Next, we extend the gazetteer lists using the concept of hyponymy. Hyponymy expresses the most specific meaning of a word unlike the general meaning (hypernymy). For example, "fire.n.02" is a hyponym of "attack.n.01" and "operation.n.05" is the hypernym. Using the correct meaning of the word helps us to retrieve more words related to the same context. For example, in the context of emergencies it is not correct to retrieve words related to the synset "fire.n.09" which have the meaning of "intense adverse criticism" (e.g., "Clinton directed his fire at the Republican Party").

The procedure follows the steps:
1. Based on the work in [13], we use WordNet to explore five levels of possible synonyms. According to the authors, the synonyms of the seed words represent the level 1 and the synonyms of the hyponyms of this level represent the level 2. The procedure continues descending in the WordNet hierarchy and taking synonyms, until reaching the fifth level. Following the previous example, we found "fire.n.01" in the level 1 and "crossfire.n.02" in the level 2. Therefore, we save the synset "crossfire.n.02". The POS (e.g., "n") and the meaning (e.g. "02") could be useful in the future for text disambiguation algorithms. Note that it is important to save the "category" related to the seed word that each word belongs to, because it will be used in the handcrafted rules (e.g., "crossfire.n.02" belongs to the seed word "fire.n.01", which is related to the category "fire").

2. Remove the repeated items once all the synonyms have been obtained.

3. Remove the noisy items. For example, the lemma "charge" is very ambiguous and depends on the context.

For the Portuguese pipeline, all the words are stored in an in-memory database called REDIS (http://redis.io/), which makes word search fast and efficient. In case of the English pipeline, we use the ANNIE gazetteer that expects the keyword lists as plain text files with one entry per line. To link keywords to concepts of an underlying ontology, keywords of different categories are stored in separate lists, e.g. “incident_fire.lst” for incident keywords related to fire or “attribute_person.lst” for keywords describing attributes of interest of persons such as “injured” or “in panic”. All lists to be used by the ANNIE gazetteer need to be specified in a configuration file “lists.def”, which is loaded during the initialization phase of the component. In this file, it is also possible to specialise the annotations the gazetteer will add defining an arbitrary number of attributes to add to a matching word or phrase.

4.3.3 Semantic Information Extraction

In the last step, semantic annotations are created based on all the annotations from previous processing steps. The primary basis for this is the detection of what, where, who, and when phrases. From these, further high-level context-dependent and independent information can be extracted in subsequent processing steps. In other words, the output of this step is a representation of the information contained in the free text description, which is language independent and specified with respect to an underlying ontology. In the simplest case, the results of the gazetteer lookup step may already be sufficient for creating a w-phrase, e.g. “stadium” or “factory” may have already been tagged as a “location” producing a where annotation, or “flames” and “detonation” may have already been tagged as an incident which assigns the ontology classes “Fire” and “Explosion” as an attribute of a what annotation, respectively. More comprehensive annotations covering sequences of words can be created using grammars which usually consist of a set of rules. Additionally, properties like quantities can be added to these w-annotations using output of e.g. the number tagger.

As explained in the previous deliverable D 3.2.1, there are basically two possible approaches on extracting semantic information from texts: 1) Rule-based systems use a set of handcrafted rules to extract information of interest, in which the creation of those rules is usually done by an expert of the domain; 2) Learning-based approaches use statistical models, which are trained using a large
corpus of annotated texts to identify information of interest. Due to the lack of a corpus of messages and the very specialized application domain, we created extraction rules manually.

In GATE, rules can be specified and implemented using the Java Annotations Pattern Engine (JAPE), which is a version of the Common Pattern Specification Language (CPSL) [9]. A single JAPE rule is basically a regular expression providing finite state transductions over annotations. In other words, a JAPE rule defines a pattern in terms of annotations and attributes of these annotations, which is matched with a given input. A JAPE grammar consists of a set of so-called phases which, in turn, consist of a set of JAPE rules to be executed during that phases. A detailed introduction at this point would go beyond the scope of this deliverable. Detailed information can be found in [10]. Exemplarily, we will present a rule for extracting what phrases. In the simplest case, a “what” annotation is assigned to any keyword or phrase, which was annotated with a majorType incident by the gazetteer. Depending on the context the RESCUER system is applied in, different incidents may be considered and the gazetteer can be initialized with several incident-specific keyword lists to account for this. Nevertheless, all incident keyword lists need to be defined with a majorType incident in order to work with our system. Additionally, any such keyword may be preceded by an arbitrary number of incident related attributes (such as “dense black smoke”) which are also of interest for the people in the command centre and which, again depending on the context of the report (e.g. the type of incident), may be added to the generic attributes of the analysis result structure. The rule we defined to account for these two cases can be seen below. In summary, it defines an incident as any arbitrary length sequence (including zero, indicated by the *) of lookups of majorType attribute or number annotations followed by exactly one lookup annotation of majorType incident.

Rule: Incident

\[
\{\text{Lookup.majorType="attribute"} | \text{Number.type="numbers"}\}^* \text{\{Lookup.majorType="incident"\}}
\]

:label

--> :label.What = {rule = "Incident"}

Similarly to English, semantic annotations for Portuguese are created based on all the annotations from previous processing steps. To detect the w-phrases (what, where, who, and when) we created a set of rules based on the gazetteer lists and POS tagging. Basically, to detect what, where and who, we get all the unigrams, bigrams, trigrams, and tetragrams of the tokens (words) provided by NLPnet, then we lemmatize each token. Given a text, we extract all the [1-4]-grams and look if some n-gram exists in the gazetteer list. For example, taking into consideration the text "Houve um ataque terrorista em Bruxelas esta manhã, há 200 pessoas feridas e pelo menos 10 mortos" (in English, "There was a terrorist attack in Brussels this morning, there are 200 injured people and at least 10 dead people."), the algorithm looks in the gazetteer list and finds 3 coincidences: the 2-gram "ataque terrorista" (terrorist attack) and the 1-gram "feridas" (injured) and "mortos" (dead). We can identify that "terrorist attack" is a what-phrase because the n-gram
“terrorist attack” belongs to the category “attack”. In the same way, we can identify who-phrases like "200 pessoas feridas" ("200 injured people") and "10 mortos" ("10 dead people") using NLPnet tags (to get numbers and nouns) and because the words “injured” and “dead” appear in the category "person_affected". To detect where-phrases we use only NLPnet tags. Following the previous example, we detect the phrase "em Bruxelas" ("in Brussels"). To detect when-phrases we use Python Regex (regular expression) rules and time lexicon.

Given the set of w-phrases (which include all the information of every preceding annotation steps as well as ontology related information) found in the text, the final step is to extract further high-level information filling the AnalysedResult data structure. For this purpose, the annotations within a when phrase can be used to correct the timestamp of the incident by subtracting the identified amount of time from the original timestamp for example. In a similar manner, the location can be corrected from where annotations. Additionally, the generic fields of the structure can be filled by applying an appropriate processing of w-phrases, which, of course, depends on the concrete case.

4.4 Towards an Implementation of a Multi-Language Text-Analysis Module and its Integration to the RESCUER Demonstration Platform

This section is intended to give a short overview of the current state of development towards a deployable text analysis component that can be integrated into an overall project prototype system. A full description of the implementation architecture is intended to be presented in D 3.2.3. Deliverable 1.5.1 defines the data structures for input (crowdmessage) and output (analyzedresult), which implement the identically named conceptual data structures for the text analysis component presented in section 4.2.

Figure 17 depicts the component architecture of the text analysis component as well as the flow of messages. It basically consists of subcomponents, which use an internal RabbitMQ broker for communication. This has two advantages: First, the additional internal broker makes it possible to easily extend the component with a new pipeline for a new language. Second, different technologies, programming languages can be used to realize different pipelines.
The **Language Identification** component on the one hand identifies the language of reports coming from the RESCUER mobile application. It polls *crowdmessage* instances which contain free text to analyse from the respective queue of the RESCUER RabbitMQ broker (topic “crowdmessage.true.*.*.*”) and forwards the *crowdmessages* to the internal broker once the language has been identified (topic “crowdmessage.true.*.*.*.language” where *language* is an identifier for the detected language), which stores them in a separate queue for each language. In our current implementation, we use the algorithm proposed by [11] called "LDIG" (Language Detection with Infinity-Gram). LDIG is a prototype of language detection for short message service, such as Twitter messages, with 99.1% accuracy for 17 languages (e.g., English, Portuguese, Spanish, Italian, French, etc.).

The **Text Analysis Pipeline** components implement the processing pipelines for different languages. They subscribe to the internal RabbitMQ broker polling the respective queue that contains messages in the language they can process (e.g. topic “crowdmessage.true.*.*.*.en” for English or “crowdmessage.true.*.*.*.pt” for Portuguese). For the final prototype of the RESCUER platform, we are working on implementations of analysis pipelines for English and Portuguese.

The architecture of both text analysis pipelines is multi-threaded to ensure maximal scalability. The core of the implementation is a threadpool executing a number of *Worker* class instances. Every Worker manages all the resources necessary to receive and process a *crowdmessage* instance, and to publish the *analyzedresult* back to the global broker. In essence, those resources are:

1. A *Channel* instance over which it can communicate with the internal RabbitMQ broker to poll *crowdmessage*, which have been identified to contain free text in English.
2. A *Channel* instance over which it can communicate with the global RabbitMQ broker to publish the results and query context information.
3. An instance of the respective text processing pipeline. Once a message has been analysed, the resulting report is published to the global broker with topic “analyzedresult.true.*.*.*”.

As of April 2016, the integration of the text analysis component to the RESCUER prototype was done successfully. As modifications of the current text processing pipelines do not affect the architecture of the overall component, we can change the pipeline independently and without affecting any other component of the prototype system.

4.5 Evaluation

4.5.1 Creating a Corpus of Test Messages

The evaluation of the text analysis component turned out to be difficult for one main reason: There is no corpus of annotated emergency text messages available which have been collected under circumstances similar to what we consider in the RESCUER project (reporting person is close to the incident and is probably stressed). The correct way to proceed would be thus to collect a reasonably large corpus during a real emergency, which is, of course, beyond the scope of this project. As a work-around, we use messages collected from Twitter (“tweets”). Therefore, we use a crawler for Python called tweepy (http://www.tweepy.org/). To filter the tweets we use the gazetteer list created as described in section 4.3. Basically, we normalize and lemmatize the tweets and obtain all the [1-4]-grams. Next, the algorithm looks for all the [1-4]-grams that are stored in the gazetteer list. If a tweet contains at least three [1-4]-grams, the tweet is chosen. Note that the algorithm is likely to choose noisy tweets. Human verification is thus necessary to sort them out. Additionally, it requires significant effort to annotate the collected corpora and we need to see to what extend the creation of a large corpus will be possible for a final evaluation which will be part of the final Deliverable.

4.5.2 Temporal Evaluation Using the Current Implementation Prototype

Using the approach described in the previous subsection, we have created a small corpus of test messages (500 in English and 500 in Portuguese). The used tweets were not related to one special event, but were collected from the live Twitter stream using the filtering approach described above. All of them are between 30 and 148 characters in size with a mean message length of 127. Table 3 summarizes the message characteristics.

<table>
<thead>
<tr>
<th></th>
<th>min</th>
<th>max</th>
<th>mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>OriginalText</td>
<td>30</td>
<td>148</td>
<td>127</td>
<td>17.60</td>
</tr>
</tbody>
</table>

With this corpus, we did a first evaluation of the runtime of the text analysis component. As test platform we used a 2.5 GHz quadcore MacBook Pro with 16 GB RAM. The results are depicted in Figure 22. The mean processing time per message is 40 ms (with 20 ms standard deviation) including
the detection of the language it is written in, the actual text processing, as well as the communication overhead due to the use of the internal RabbitMQ broker. The numbers show that component in its current state fulfils the requirements defined in deliverable D7.4, which specifies an average processing time of 100 ms per message.

Table 4: Runtime statistics of test message corpus in ms

<table>
<thead>
<tr>
<th>Component</th>
<th>mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>LanguageDetector</td>
<td>6.31</td>
<td>1.77</td>
</tr>
<tr>
<td>TextAnalysis</td>
<td>19.74</td>
<td>14.22</td>
</tr>
<tr>
<td>RabbitMQ</td>
<td>14.36</td>
<td>4.03</td>
</tr>
<tr>
<td>Total</td>
<td>40.41</td>
<td>20.02</td>
</tr>
</tbody>
</table>

4.6 Next steps

During the rest of the project, we will focus on the following points:

- Investigation of suitable ontologies for emergencies and incidents and integration to the prototype of text analysis component;
- Exploration of the usage of context in the text analysis component;
- Refinement of the prototype of the text analysis component;
- Evaluation of the text analysis component.
5 Emergency State Information

This chapter presents the specification of the emergency state information to be kept by the RESCUER platform. This information is expected to be used by several RESCUER components, such as by the data prioritization components (as e.g. specified in D1.1.4: System Specification, Section D.5) and the data analysis components (to be addressed in the next version of this deliverable). Deliverables D1.2.3: System Architecture 3 and D1.5.3: Integrated Platform Demonstrator 3 are going to provide the details about which component holds the emergency state information, how it is represented computationally, and how to obtain it through the JSON-RabbitMQ technology.

<table>
<thead>
<tr>
<th>Information</th>
<th>Range of Values</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>incident_types.INFORMED</td>
<td>{Fire, Gas Leak, Explosion, Environmental, Vandalism, Crush, Mass Collapse, Stampede, Shooting}</td>
<td>All reports</td>
</tr>
<tr>
<td>incident_type.CONFIRMED</td>
<td>{Fire, Gas Leak, Explosion, Environmental, Vandalism, Crush, Mass Collapse, Stampede, Shooting}</td>
<td>WF Incident Report</td>
</tr>
<tr>
<td>incident_type.DETECTED</td>
<td>{Fire, Stampede}</td>
<td>Image/Video Analysis</td>
</tr>
<tr>
<td>flames.CONFIRMED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>smoke.DETECTED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>light_grey_smoke.CONFIRMED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>light_grey_smoke.DETECTED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>dark_grey_smoke.CONFIRMED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>dark_grey_smoke.DETECTED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>other_color_smoke.CONFIRMED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>other_color_smoke.DETECTED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>turmoil_panic.CONFIRMED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>turmoil_panic.DETECTED</td>
<td>[Yes</td>
<td>No]</td>
</tr>
<tr>
<td>injured.CONFIRMED</td>
<td>Yes, No, Unknown</td>
<td>WF Incident Report</td>
</tr>
<tr>
<td>number_injured.CONFIRMED</td>
<td>Number, Unknown</td>
<td>ERTK User / WF Incident Report (Follow-up)</td>
</tr>
<tr>
<td>incident_position.DETECTED</td>
<td>Lat, Long</td>
<td>GPS position of a civilian or supporting force</td>
</tr>
<tr>
<td>Information</td>
<td>Range of Values</td>
<td>Source</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>incident_position.INFORMED</td>
<td>Lat, Long</td>
<td>Manual entry of a civilian or supporting force (map or text)</td>
</tr>
<tr>
<td>incident_position.CONFIRMED</td>
<td>Lat, Long</td>
<td>GPS position or manual entry of WF (map or text)</td>
</tr>
<tr>
<td>incident_place.INFORMED</td>
<td>Name of Affected Company or Event Name</td>
<td>Manual entry of a civilian or supporting force (map or text)</td>
</tr>
<tr>
<td>incident_place.CONFIRMED</td>
<td>Name of Affected Company or Event Name</td>
<td>Manual entry of WF (map or text) / ERTK User</td>
</tr>
<tr>
<td>incident_date</td>
<td>dd/MM/AAAA</td>
<td>First report / ERTK User</td>
</tr>
<tr>
<td>incident_starting_time</td>
<td>hh:mm</td>
<td>First report / ERTK User</td>
</tr>
<tr>
<td>fatalities.CONFIRMED</td>
<td>Yes, No, Unknown</td>
<td>ERTK User</td>
</tr>
<tr>
<td>number_fatalities.CONFIRMED</td>
<td>Number, Unknown</td>
<td>ERTK User / WF Incident Report (Follow-up)</td>
</tr>
<tr>
<td>leaked_material.DETECTED</td>
<td>Name of the leaked material</td>
<td>Text Analysis</td>
</tr>
<tr>
<td>leaked_material.CONFIRMED</td>
<td>Name of the leaked material</td>
<td>Text Analysis of WF Incident Report / ERTK User</td>
</tr>
<tr>
<td>evacuation.CONFIRMED</td>
<td>Yes, No, Unknown</td>
<td>Follow-up questions</td>
</tr>
<tr>
<td>road_blocks.CONFIRMED</td>
<td>Yes, No, Unknown</td>
<td>Follow-up question</td>
</tr>
</tbody>
</table>

The notion of CONFIRMED used in D1.1.4: System Specification, Section D.5, which had the meaning of “confirmed by data analysis or reliable source (workforce or ERTK user)”, is refined in Table 5 in DETECTED, with the meaning of “computationally detected”, as e.g. by image/video/text analysis or GPS position; and CONFIRMED, with the meaning of “informed by a workforce (WF Incident Report or WF Status Report) or by an ERTK user and thus already confirmed”. In addition, the emergency state information also includes the notion of INFORMED, with the meaning of “informed by a civilian or supporting force” and thus demanding confirmation.

All references to CONFIRMED in D1.1.4: System Specification, Section D.5 should be understood as DETECTED or CONFIRMED as defined in this document.
6 Conclusions

In this deliverable, we discussed the problems to be solved by the Data Analysis components, including data in the form of image, video, and text. Concerning images, we have an advanced version of the software architecture already in production. We are still working on improvements concerning smoke detection, since the fire detection was significantly better. Concerning video, we have achievements related to the detection of fire in videos, which is already in production. We are working on the detection of smoke, and on the estimation of crowd density, which poses several challenges, but whose initial solution has presented good results. Concerning text, we already reached the production stage integrated to the Rescuer platform. The next challenge is to have a multi-language (English and Portuguese) component.

We stress that these last periods of work have focused in the integration of the components in the current Rescuer platform. Due to these efforts, several issues were solved concerning integration, communication between partners, and requirements. Some of the results are already in production and are part of the demonstration platform headed by the integration team. Still, some issues have to be solved. The most important of them is the use of the context information provided by the emergency state. The emergency state was initially defined in D1.1.4: System Specification, but specified in detail in parallel to the writing and review of this document. As a consequence, in this document, we only present the specification of the emergency state information, which will be taken into consideration in the next version of the data analysis components. Making use of emergency state information in the data analysis components demands not only the conceptual notion of what a context is, but also how to represent it computationally, how to communicate it through the JSON-RabbitMQ technology, and from which component to acquire it. The next steps to this regard will be considering the emergency state information as additional information to the data analysis algorithms.

Next iteration

By the time of this document, the demonstration of the Rescuer solution is already taking place. This first prototype will raise many issues and point out improvements to be implemented in the next iteration. Accordingly, the next deliverable D3.2.3 shall present the final solutions of each Data Analysis component, its integration, and results in the production environment.
Bibliography

Glossary

Abbreviations

**CBIR** Content-Based Image Retrieval
**DT** Decision Trees
**FEM** Feature-Extractor Method
**FV** Feature Vector
**IBL** Instance-Based Learning
**MLP** Multilayer Perceptron
**NB** Naïve Bayes
**ROI** Region of Interest
**SVM** Support Vector Machine