D4.2 Demonstration of social media analytics for localising the origin of wildfire ignition

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 101037247

Project Acronym	SILVANUS
Grant Agreement number	101037247 (H2020-LC-GD-2020-3)
Project Full Title	Integrated Technological and Information Platform for Wildfire Management
Funding Scheme	IA – Innovation action

DELIVERABLE INFORMATION

Deliverable Number:	D4.2
Deliverable Name:	Demonstration of social media analytics for localising the origin of wildfire ignition
Dissemination level:	Public
Type of Document:	Demonstration
Contractual date of delivery:	31/03/2023 (M18)
Date of submission:	31/032023
Deliverable Leader:	CERTH
Status:	Final
Version number:	V1
WP Leader/ Task Leader:	WP4 – T4.4
Keywords:	Social media, Social Media Crawlers, Social Media Analysis Toolkit, Relevance Classification, Event Recognition, Fire Event Detection, Smoke Detection, Concept extraction, RDF Mapping
Abstract:	This deliverable reports on the implementation of a framework for social media analytics, focused on detecting the origin of wildfire ignition. The framework includes the following components: i) Social Media crawlers, ii) Social Media Analysis Toolkit, iii) Fire Events Detection, iv) knowledge base database, and v) Silvanus Dashboard for visualization. A questionnaire was circulated to collect search criteria based on which the Social Media Crawlers retrieve fire-related data from social media. Moreover, the implementation details of Social Media Crawlers for Twitter, Facebook, and websites are explained. Furthermore, the Social Media Analysis Toolkit's textual and visual analysis methods are described. Also, the JSON structure of social media posts and fire events is presented along with the fire events transformation from JSON to RDF mapping and the storing of the

	events in a knowledge base. In addition, the Silvanus Dashboard is introduced as a user interface for retrieving fire events from knowledge base and visualizing them in a map as popups. The document concludes with a summary and future work.

Lead Author(s):	Aris Bozas, Yiannis Kouloglou, Ilias Gialampoukidis
Reviewers:	PLAMEN, SMURD, AMIKOM

Disclaimer

All information in this document is provided “as is" and no guarantee or warranty is given that the information is fit for any particular purpose.

The user there of uses the information at its sole risk and liability. For the avoidance of all doubts, the European Commission has no liability in respect of this document, which is merely representing the authors view.

Document History			
Version	**Date**	**Contributor(s)**	**Description**
V0.1	10/1/2023	Aris Bozas (CERTH) Yiannis Kouloglou (CERTH)	Table of contents
V0.5	31/01/2023	Aris Bozas (CERTH) Yiannis Kouloglou (CERTH)	First round of input
V0.8	10/3/2023	Aris Bozas (CERTH) Yiannis Kouloglou (CERTH) Ilias Gialaboukidis (CERTH)	First Draft for internal review
V0.9	22/3/2023	Aris Bozas (CERTH) Yiannis Kouloglou (CERTH) Ilias Gialaboukidis (CERTH	Document after revision ready for review
V1	27/3/2023	Aris Bozas (CERTH) Yiannis Kouloglou (CERTH) Ilias Gialaboukidis (CERTH	Final version

List of Contributors

Partner	Author(s)
AMIKOM	Kusrini, Arief Setyanto, Arif Dwi Laksito
ATOS	Jose-Ramon Martinez-Salio
HB	Johan Eklund, Sándor Darányi
CERTH	Aris Bozas, Yiannis Kouloglou, Ilias Gialampoukidis
CTL	Georgia Christodoulou, Konstantinos Avgerinakis, Maria Maslioukova, Marios Iakovou, Stelios Kontogiannis
EAI	Simone Martin Marotta
UISAV	Martin Šeleng, Štefan Dlugolinský
ITTI	Dominika Grunwald, Marcin Przybyszewski

List of acronyms and abbreviations

ACRONYM	Description
AI	Artificial Intelligence
API	Application Programming Interface
APP	Application
AWS	Amazon Web Services
BIO	Beginning Inside Outside
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-Separated Values
DCNN	Deep convolutional neural networks
FOSS	Free and Open-Source Software
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
ML	Machine Learning
NER	Name Entity Recognition
NFS	Network File System
NN	Neural Network
NSFW	Not Safe For Work
OWL	Web Ontology Language
ORM	Object Relational Mapping
PO	Pilot
RAM	Random-access memory
RDF	Resource Description Framework
REST	Representational State Transfer
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
WP	Work Package
WPL	Work Package Leader

WSGI	Web Server Gateway Interface
W3C	World Wide Web Consortium

List of beneficiaries

No	Partner Name	Short name	Country
1	UNIVERSITA TELEMATICA PEGASO	PEGASO	Italy
2	ZANASI ALESSANDRO SRL	Z&P	Italy
3	INTRASOFT INTERNATIONAL SA	INTRA	Luxembourg
4	THALES	TRT	France
5	FINCONS SPA	FINC	Italy
6	ATOS IT SOLUTIONS AND SERVICES IBERIA SL	ATOS IT	Spain
6.1	ATOS SPAIN SA	ATOS SA	Spain
7	EMC INFORMATION SYSTEMS INTERNATIONAL	DELL	Ireland
8	SOFTWARE IMAGINATION & VISION SRL	SIMAVI	Romania
9	CNET CENTRE FOR NEW ENERGY TECHNOLOGIES SA	EDP	Portugal
10	ADP VALOR SERVICOS AMBIENTAIS SA	ADP	Portugal
11	TERRAPRIMA - SERVICOS AMBIENTAIS SOCIEDADE UNIPESSOAL LDA	TP	Portugal
12	3MON, s. r. o.	3MON	Slovakia
13	CATALINK LIMITED	CTL	Cyprus
14	SYNTHESIS CENTER FOR RESEARCH AND EDUCATION LIMITED	SYNC	Cyprus
15	EXPERT SYSTEM SPA	EAI	Italy
16	ITTI SP ZOO	ITTI	Poland
17	Venaka Treleaf GbR	VTG	Germany
18	MASSIVE DYNAMIC SWEDEN AB	MDS	Sweden
19	FONDAZIONE CENTRO EURO-MEDITERRANEOSUI CAMBIAMENTI CLIMATICI	CMCC F	Italy
20	EXUS SOFTWARE MONOPROSOPI ETAIRIA PERIORISMENIS EVTHINIS	EXUS	Greece
21	RINIGARD DOO ZA USLUGE	RINI	Croatia
22	Micro Digital d.o.o.	MD	Croatia
23	POLITECHNIKA WARSZAWSKA	WUT	Poland
24	HOEGSKOLAN I BORAS	HB	Sweden
25	GEOPONIKO PANEPISTIMION ATHINON	AUA	Greece
26	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	CERTH	Greece
27	PANEPISTIMIO THESSALIAS	UTH	Greece
28	ASSOCIACAO DO INSTITUTO SUPERIOR TECNICO PARA A INVESTIGACAO E DESENVOLVIMENTO	IST	Portugal
29	VELEUCILISTE VELIKA GORICA	UASVG	Croatia

No	Partner Name	Short name	Country
30	USTAV INFORMATIKY, SLOVENSKA AKADEMIA VIED	UISAV	Slovakia
31	POMPIERS DE L'URGENCE INTERNATIONALE	PUI	France
32	THE MAIN SCHOOL OF FIRE SERVICE	SGPS	Poland
33	ASSET - Agenzia regionale Strategica per lo Sviluppo Ecosostenibile del Territorio	ASSET	Italy
34	LETS ITALIA srls	LETS	Italy
35	Parco Naturale Regionale di Tepilora	PNRT	Italy
36	FUNDATIA PENTRU SMURD	SMURD	Romania
37	Romanian Forestry Association - ASFOR	ASFOR	Romania
38	KENTRO MELETON ASFALEIAS	KEMEA	Greece
39	ELLINIKI OMADA DIASOSIS SOMATEIO	HRT	Greece
40	ARISTOTELIO PANEPISTIMIO THESSALONIKIS	AHEPA	Greece
41	Ospedale Israelitico	OIR	Italy
42	PERIFEREIA STEREAS ELLADAS	PSTE	Greece
43	HASICSKY ZACHRANNY SBOR MORAVSKOSLEZSKEHO KRAJE	FRB MSR	Czechia
44	Hrvatska vatrogasna zajednica	HVZ	Croatia
45	TECHNICKA UNIVERZITA VO ZVOLENE	TUZVO	Slovakia
46	Obcianske zdruzenie Plamen Badin	PLAMEN	Slovakia
47	Yayasan AMIKOM Yogyakarta	AMIKOM	Indonesia
48	COMMONWEALTH SCIENTIFIC AND INDUSTRIAL RESEARCH ORGANISATION	CSIRO	Australia
50	FUNDACAO COORDENACAO DE PROJETOS PESQUISAS E ESTUDOS TECNOLOGICOS COPPETEC	COPPETEC	Brazil

Index of figures

Index of tables

Table of Contents

Executive Summary

This document defines the demonstration of social media analytics for localising the origin of wildfire ignition which focuses on social media sensing and concept extraction. The document follows a certain structure, with an introduction in section 1 that explains the usefulness of social media in monitoring, prevention, detection, and management of wildfire. This section continues with the purpose and scope of the document and a brief presentation of the document structure.

Section 2 provides a description of the Social Media Sensing Framework, which includes various components such as the Social Media Crawlers, which collect social media posts from Twitter, Facebook, and other websites. The Social Media Analysis Toolkit, which analyzes and enriches these posts with additional knowledge. The Fire Events detection that accepts the enriched social media posts with the help of an API and detects fire events. Then fire events are transformed from JSON format to ontologies that are stored in knowledge base database.

In Section 3, the document explains how a questionnaire was circulated to pilot leaders in order for CERTH to collect their answers to define the search criteria, which are then used by the social media crawlers to collect social media posts based on keywords, Twitter accounts, Facebook pages, and websites.

Section 4.1 provides an in-depth look into the technical details and development of the social media crawlers, which were designed to collect data from Twitter, Facebook, and websites. The section begins with a description of the implementation of the Twitter crawler in Section 4.1, which is a Python implementation that collects tweets in near real-time by performing complex queries in the Twitter API. Section 4.2 describes the implementation of the Facebook crawler, which collects new Facebook posts through a configured webhook. Finally, Section 4.3 describes the implementation of the Web crawler, which obtains digital multimedia content through a web harvester module that downloads images, videos, and text from websites.

Section 5 provides a comprehensive overview of the textual and visual analyses provided by the Social Media Analysis Toolkit. Specifically, Section 5.1 focuses on textual analysis, describing the Relevance Classification and Text Categorization methods used to categorize social media posts based on their relevance and content in section 5.1.1. Additionally, Section 5.1.2 covers the Textual Concepts Extraction methods, which extracts meaningful concepts and entities from social media post texts. Section 5.1.3 describes the Event Recognition method, which identifies significant events mentioned in social media posts. Section 5.1.4 describes the Location Extractions method, which extracts location information from social media post texts. In Section 5.2, the visual analysis components and methods are explained, with Section 5.2.1 detailing the Fire and Smoke Detection in Images method that utilizes computer vision techniques. Section 5.2.2 covers Visual Concept Extraction methods, which use object recognition algorithms to extract meaningful visual concepts and objects from images. Finally, Section 5.2.3 describes the Location Extraction methods, which extracts location information from social media post images.

Section 6 offers a detailed overview of the JSON structure of individual social media posts that are acquired by the Fire Detection module from the Social Media Sensing API. Additionally, it describes the JSON structure used to represent fire events in section 6.1 and how these structures are transformed into RDF semantic representations in section 6.2 and how they are stored in the knowledge database in section 6.3.

In Section 7, the Silvanus Dashboard is introduced as a user web interface for visualizing fire events by acquiring the fire event data from the knowledge database. Early crawling statistics for social media collected from Twitter with the Twitter crawler are presented in Section 8.

Finally, Section 9 concludes with a summary of the document and outlines future work.

1 Introduction

Social media sensing has become an essential tool for monitoring and responding to a wide range of real-world events, including natural disasters such as fires. With the help of advanced algorithms and machine learning, researchers and emergency responders can collect and analyze vast amounts of social media data in real-time, providing crucial insights into the location, spread, and impact of fires.

The vast amount of information contained in social media can be utilized for various purposes and applications. With the increasing use of mobile phones, people who use social media have begun to frequently report various disaster incidents that impact their daily lives. This allows the wider public to participate in disaster monitoring by reporting incidents connected to such events (Saroj et al., 2020). The use of social media sensing for fire management has many advantages. It allows early detection of fires, even in remote or difficult-to-reach areas, by collecting information from a variety of sources. This information can help emergency responders quickly assess the situation, allocate resources, and coordinate their efforts. Additionally, social media data can provide important context and situational awareness, such as the size and direction of the fire, areas affected, and the number of people in need of assistance.

Moreover, social media sensing can also help monitor the progress of firefighting efforts and provide important insights into the effectiveness of different tactics and strategies. This real-time feedback can help responders adjust their approach to optimize the use of resources and ensure the best possible outcomes.

Overall, social media sensing is a powerful tool for disaster prevention, response management, including enhancing situation awareness, promoting emergency information flow, predicting disasters and coordinating rescue efforts (Shan et al., 2019). As more data becomes available, and machine learning algorithms become more advanced, the potential of social media sensing in improving fire management will only continue to grow.

1.1 Purpose and scope

SILVANUS aims to provide a forest management platform that is both environmentally sustainable and resilient to the effects of climate change. The platform will use innovative technologies to prevent and combat forest fires, meeting the needs of efficient resource use and protecting against wildfire threats worldwide. The project will bring together experts from various fields, including environmental, technology, and social sciences, to improve the ability of regional and national authorities to monitor forest resources, assess biodiversity, create more precise fire risk indicators, and raise awareness among citizens through safety campaigns. By establishing synergies between these areas, the project aims to promote sustainable forest management and reduce the impact of forest fires on the environment and communities.

CERTH's involvement in the SILVANUS project is through its leadership of Task 4.4, which focuses on social media sensing and concept extraction. The primary objective of this task is to collect information from social media, primarily Twitter, about possible fire incidents reported by citizens to contribute to early detection of wildfires. CERTH will work with end-users to define specific keywords, accounts, and areas of interest that will be used to retrieve valuable social media data. These criteria will be used to formulate queries for the Twitter API, and relevant tweets will be collected in real-time. In addition to collecting data, this task will also involve the extraction of concepts from both text and visual content to provide a deeper understanding of potential fire events.

1.2 Document structure

The deliverable is structured:

- Executive summary

- *Section 1* – Introduction: Presents an introduction the goals, the scope and an overview of this deliverable.
- *Section 2* – Social Media Sensing Framework: Introduces a Framework explains and presenting the module and their functionality.
- *Section 3* – Collection of social media search criteria: Describes how the search criteria has been defined.
- *Section 4* – Social Media Crawlers: Explains how the social media crawlers are implemented and what technologies are used and how they retrieve data from each platform.
- *Section 5* – Social Media Analysis Toolkit: Illustrates the types of analysis and methods that are used Social Media analysis toolkit and performs in the retrieved social media posts.
- *Section 6* – Fire Events stored to knowledge database: How the fire events are transformed from JSON format to an RDF representation and stored into a graph database
- *Section 7* – Visualization of fire events: Illustrates how the fire events are visualized in the user interface of the SILVANUS Dashboard.
- Section 8 – Crawling Results: Displays the first results and statistics from the social media crawlers.
- Section 9 – Concluding remarks: Provides a summary of the document is pro and outline future work.

2 Social Media Sensing Framework

The Social Media Sensing framework (Figure 1) is designed to collect and pre-process information related to fires from social media. Whenever a social media post is retrieved, the Social Media Analysis Toolkit's components are called separately with individual API calls, and each analysis outcome coming from its components is added to the original JSON of the social media post. The enhanced post is then stored in the mongoDB[1] database in JSON format. Afterward, the posts are fed into a fire event detection module, which detects and generates fire events. The fire events are then mapped from JSON to an RDF representation and stored in a knowledge base database. These events can be easily visualized for the end-users through a semantic representation on the user interface.

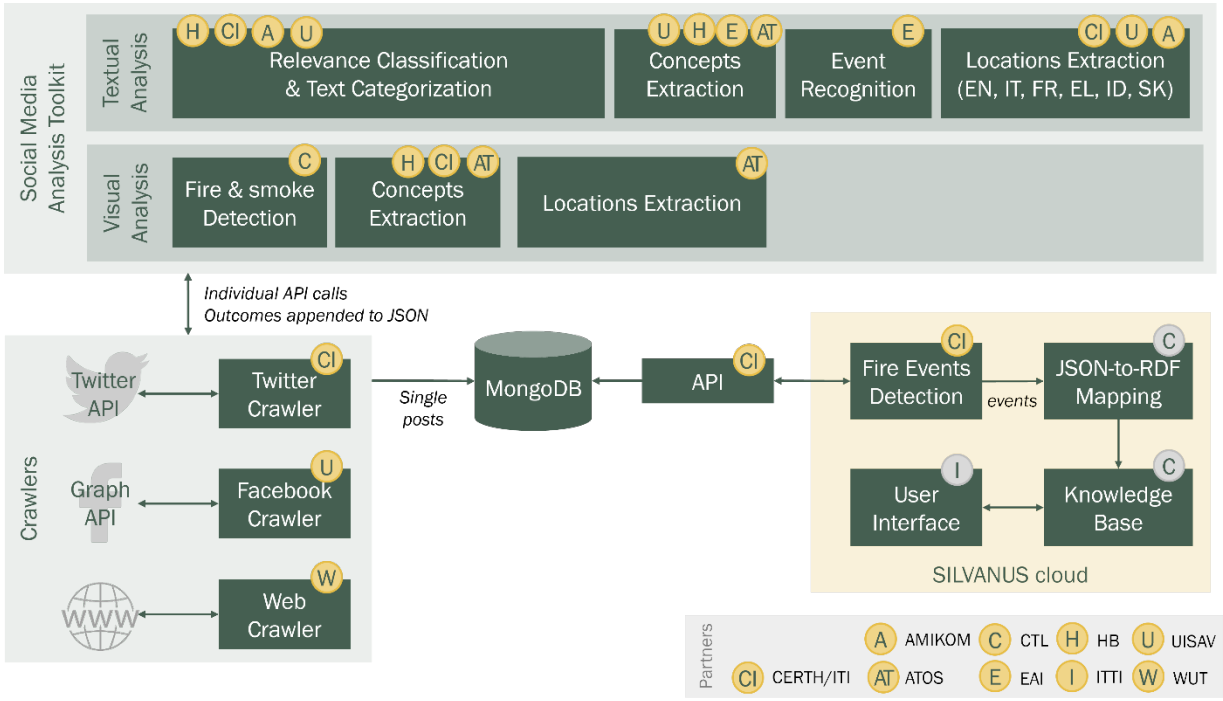


Figure 1: Social media sensing Framework

Social media data are collected from three different sources, i.e., the Twitter by utilizing the Twitter API[2], the Facebook by utilizing the Graph API[3] and by crawling websites that the end users have specified.

- The *Twitter Crawler* (see section 4.1), connects to the Twitter API with credentials, and formulates complex queries in order to fetch tweets related to fire incidents, from Twitter's public data stream.
- The *Facebook Crawler* (see section 4.2) is implemented as a Facebook App is an app that uses the Graph API to read posts on Facebook without explicit permissions. It listens to real-time notifications of changes to specific objects in the Facebook Social Graph called webhooks. The crawler receives notifications in JSON format on newly published posts through a webhook.
- The *Web Crawler* (see section 4.3) is a modular system for web harvesting and content analysis that can handle large volumes of data in a distributed manner using Front-End for site scraping, file analysis, and worker monitoring and Back-End components for task scheduling, database, network file system, and support modules.

[1] https://www.mongodb.com/
[2] https://developer.twitter.com/en/docs/twitter-api
[3] https://developers.facebook.com/docs/graph-api

Every newly retrieved social media post, is enriched with additional knowledge by analysing its textual and visual content (see section 5) through the modules of the Social Media Analysis Toolkit, which are called as REST API services.

The textual analysis components from Social Media Analysis Toolkit are:

- The *Relevance classification* and Text categorization *module* (see section 5.1.1) which categorizing social media posts based on their pertinence to the topic of interest and organizing them into distinct categories based on their content.
- The textual *Concepts Extraction module* (see section 5.1.2) which extracts significant concepts and entities from the text present in social media posts. Such entities may include names of individuals, organizations, geographical locations, and other pertinent details.
- The *Event Recognition module* (see section 5.1.3) which identifies important events mentioned in social media posts from the textual content of the post.
- *Location extraction module* (see section 5.1.4) which consists of web services that automatically extract the location of a social media post from the text content.

The visual analysis components from Social Media Analysis Toolkit are:

- The *Fire and smoke detection module* (see section 5.2.1) which analyses the image(s) of a social media post and returns if it contains fire or smoke or both.
- The visual *Concept Extraction module* (see section 5.2.2) which categorizes new observations into predefined classes and retrieves high-level information (i.e., concepts) from the images associated with the posts.
- The *location extraction from images module* (see section 5.2.2) which extract the location from the image(s) of a post.

Subsequently, the enriched social media posts are stored in the database and the fire event detection module frequently checks the collection of single tweets with the help of the social media sensing API and, if an event is detected, it is stored in a different collection as well as sent to the knowledge base (see section 6). From the knowledge base the Silvanus Dashboard can perform fast queries in order to fetch and visualize the fire events (see section 7).

3 Collection of social media search criteria

The social media content is collected by social media crawlers by utilizing the social media platforms APIs such us Twitter API and Facebook API or crawling the content of various prespecified websites. Nevertheless, before the development of the social media crawlers, it is important to define the keywords/phrases, the Twitter accounts, the Facebook pages that are going to be utilized as filtering parameters in API queries for Twitter and Facebook APIs and the websites that the Web crawler is going to scrap i.e., the search criteria. It is essential these search criteria to be defined with the collaboration of the pilot leaders. Their specialisation in their scientific field leads to an understanding of the task, combined with the expertise of their native language, resulting in a more accurate selection of the search criteria for the retrieval of valuable social media data.

3.1 Circulation of Questionnaire

The first step was to prepare a questionnaire (Table 1) that gathers information about the area of interest, the contact information in order to establish a communication with the end users of Silvanus, the experience they have with the monitoring of social media, how the social data are going to contribute in their pilot case and to define search criteria (keywords, accounts, Facebook pages, websites) related to fire events for crawling in Twitter, Facebook and websites.

1. Contact Information	
1.1. Name	
1.2. E-mail	
1.3. Pilot Use Case	
2. Involvement of social media data ***We aim to collect posts from social media platforms in real-time, analyse the textual and visual information they carry, and promptly detect fire incidents based on social data.***	
2.1. Do you already use/monitor social media in your operations and how?	
2.2. How do you foresee the contribution of social media sensing in your pilot use case?	
2.3. Which phase (A/B/C) do you expect social media sensing to support?	

3. Sensing Twitter ***We aim to develop a Twitter crawler that retrieves tweets in real time according to user-defined search criteria. Twitter offers two options: (1) collect tweets that contain a keyword or a phrase and (2) collect tweets that are posted from a specific Twitter account. The tweets can be in English and/or in the language of the PUC's area of interest.***
3.1. What keywords/phrases (if any) would be of interest for your pilot use case? **(Examples: "forest fire", "smoke plumes", "wildfire", etc.)**
3.2. Which Twitter accounts (if any) would be of interest for your pilot use case? **(Examples: @FireNewsEurope, @WildFires)**
4. Sensing Facebook ***We are also interested in investigating what can be crawled from Facebook, taking into consideration the many limitations. In Facebook there are three main entities: profiles, pages and groups. Since the first type concerns personal pages, we will completely disregard them and focus on pages and groups.***
4.1. Which Facebook pages (if any) would be of interest for your pilot use case? **(Examples: LNU Fire Scanner, Responding Fire)**
4.2. Which Facebook groups (if any) would be of interest for your pilot use case? **(Examples: FireNews Greece, FireRescueVolunteers)**
4.3. What keywords/phrases should posts from the above pages/groups contain? **(Can be same to 3.1.)**

5. Sensing the Web ***Apart from the social media platforms, we will also investigate the collection of information from websites. Custom Web scrappers can be implemented to automatically retrieve posts/articles from specific web pages.***
5.1. Which websites would be of interest for your pilot use case? What particular information should be crawled from these websites? (Please be as specific as possible)

Table 1: *Social media sensing questionnaire*

The questionnaire that is presented in Table 1 was circulated to the Pilot leaders of Silvanus. The pilots that were interested in the social media sensing for fire incidents, in order to contribute to early-stage detection of wildfires in their area of interest were PO1, P02, P03, P07, P09, P10. The area of interest for each Pilot is shown in Table 2.

Pilot	Area of interest
P01	Italy
P02	Slovakia
P03	France
P07	Greece
P09	Australia
P10	INDONESIA

Table 2: Area of interest for each pilot

After the circulation of the questionnaire, the first step was to collect all the answers from Pilot leaders and for each pilot interested in social media sensing identify the phase of Silvanus (A/B/C) the social media sensing is going to support and which of the pilots are interested in Twitter, Facebook or web crawling. The summary of the Pilot leaders' answers is presented in the Table 3 below.

Pilots	Phases			Twitter			Facebook		Web Crawling
	A	B	C	Keywords	Accounts	Pages	Groups / Applications	Keywords	
P01	✔	X	X	✔	✔	✔	✔	✔	✔
P02	✔	✔	✔	X	X	X	✔	X	X
P03	✔	✔	X	✔	✔	✔	✔	✔	X
P07	✔	✔	X	✔	✔	✔	✔	✔	✔
P09	X	✔	✔	✔	✔	✔	✔	✔	✔
P10	X	✔	X	✔	✔	X	X	X	X

Table 3: *Questionnaire summary answers per pilot*

3.2 Definition of search criteria

The second step was the collection of the search criteria (keywords, Twitter accounts, Facebook pages, websites) from the answers of the Pilot leaders, as based on these search criteria the social media crawlers will collect social media data related for fire incidents. The search criteria were defined after multiple rounds of communications. In the first round of communications, every Pilot provided CERTH with a set of keywords, Twitter accounts, Facebook pages and websites in order to start crawling fire related social information. However, some of the keywords were too broad and generic resulting to retrieve social data that were irrelevant to fire related incidents. For this reason, further discussions with the end users continued for the refinement of the initial search criteria. Finally, after a second round of communications, some keywords were replaced or removed as they were found to be too noisy (irrelevant information). The process of the refinement of the search criteria will continue also in the future in order to ensure as much as possible the collection of fire related information from the social media crawlers.

Table 4 displays all the Twitter accounts that are defined by the Pilot leader. The accounts are coloured green to show that were defined in the first round of communications.

Social Media Twitter accounts				
P01 - Italy	**P03 - France**	**P07 - Greece**	**P09 - Australia**	**P010 - Indonesia**
@AlertIncendi	@WildFires	@GSCP_GR	@QldFES	@bmkg
@SOSbochi		@pyrosvestiki		
@SOSincidendi				

Table 4: Search criteria for Twitter accounts

Table 5 presents all the Facebook pages that were defined by the Pilot leaders. The accounts that are coloured green show that were defined in the first round of communications. The account that are blue coloured blue are defined in the second round of communications to enhance and refine the initial search criteria.

Social Media Facebook pages			
P01 - Italy	**P03 - France**	**P07 - Greece**	**P09 - Australia**
ASSET - Agenzia Sviluppo Ecosostenibile Territorio Regione Puglia	Official Facebook page PUI	Πυρκαγια Ενημερωση	Queensland Fire and Emergency Services - QFES
Regione Puglia		dasarxeio.com	
Protezione Civile Regione Puglia			
Tepilora - Parco Naturale Regionale			
RAS - Regione Autonoma della Sardegna			
Comune di Torpè			
Comune di Posada			
Comune di Bitti			

Ceas Casa Delle Dame Posada			
CEAS Montalbo di Lodè - Centro Educazione Ambientale e alla Sostenibilità			
CEAS Torpè Centro Educazione Ambientale Porta del Parco			

Table 5: Search criteria for Facebook page

Finally, the Table 6 shows all the search criteria keywords defined by the Pilot leaders. The keywords with the green colour were defined in the first round of communications and found to be useful in the collection of social media posts by the crawlers. On the other hand, the red colour keywords were defined in the first round of communications but were deleted or replace with other keywords in order to reduce the noise of the retrieved social media posts by the crawlers. Lastly, the blue colour keywords are defined in the second round of communications to enhance and refine the initial search criteria.

Social Media Keywords (Twitter/Facebook)							
P01 - Italy		P03 - France	P07 - Greece		P09 - Australia	P010 - Indonesia	
English	Italian	English	English	Greek	English	Indonesian	English
forest Fire	incendio boschivo	wildfire	wildfire	φωτιά	bushfire	kebakaran hutan	forest fire
arson	incendio doloso	forest fire	forest Fire	πυρκαγιά		karhulta	wild land fire
fire risk	rischio incendio	industrial accident	pyrocumulus	Δασική πυρκαγιά(ς)		darutat asap	smoke emergency
smoke	fumo	disasters	evacuation alert	πυκνός καπνός		Kabut asap	smoke fire
hazard	pericolo	explosion	wildfire alert	προληπτική εκκένωση		sesak nafas	urban fire
			forest restoration	μέτωπο πυρκαγιάς		kebakaran lahan	forest fire risk
			WUI fire	φυσικές καταστροφές		resiko kebakaran hutan	Forest fire emergency
			fire resilience	κλιματική αλλαγή		resiko kebakaran hutan	smoke
			fire weather	αναδάσωση			breath difficulty
			fire catastrophe	αναδασώνουμε			
			smoke Evoia/Evia	καπνός Εύβοια			
			wildfire(s) Greece	φυτεύομε νέα δέντρα			
			forest Fire(s) Greece	κίνδυνος πυρκαγιάς			
			wildfire(s) Evoia/Evia	υψηλός κίνδυνος πυρκαγιάς			
			forest fires Evoia/Evia	εναέρια μέσα			

			wildfire/fire evacuation	πυροσβεστικενα εναέρια μέσα			
			wildfire/fire warning	canadair			
			wildfire(s) Europe	αγροτοδασική πυργκαγιά			
			forest fire Europe	μαίνεται η φωτιά			
			catastrophe	δυνατοί άνεμοι πυρκαγιά			
			smoke	υψηλές θερμοκρασίες πυρκαγιάς			
			Greece	Εύβοια			
			Evoia/Evia	καπνός			
			evacuation				
			warning				
			Europe				
			global warming				
			climate change				
			climate crisis				

Table 6: Social media search criteria keywords for Twitter and Facebook.

4 Social Media Crawlers

Social media crawlers are important because they enable individuals and organizations to gather and analyse large amounts of data from social media platforms. With the massive amounts of content being shared on social media every day, it can be difficult to manually gather and analyse relevant information. In SILVANUS project the social media crawlers automate the process of data collection for fire related information in social media.

This section provides an in-depth look into the technical details and development of our social media crawlers, which are designed to collect data from Twitter, Facebook, and websites. The implementation of Twitter crawler is described on section 4.1, the implementation of Facebook crawler on section 4.2 and the implementation of Web crawler on section 4.3.

4.1 Implementation of Twitter Crawler

Twitter is a popular platform used by millions of unique users for various purposes, one of which is reporting disasters, such as fire incidents, which can be used under the Silvanus project. For this reason, CERTH developed a Twitter crawler that consumes tweets in near real-time from the Twitter API. The Twitter crawler was developed with the Python programming language and using Python's Tweepy library, which provides easy connection to the Twitter API endpoints. More specifically, Tweepy[4] establishes an open connection to the filter stream endpoint[5] that provides access to Twitter's public data stream and then the crawler can collect tweets in near real-time by performing complex queries containing multiple stream rule sets[6] on the filter stream endpoint.

To access Twitter's filtered streaming endpoint, you need a Twitter account, a developer account on the Twitter portal[7], at least Essential access (the lowest level of access to Twitter's API) and a bearer token[8]. In Twitter API v2 a bearer token allows developers to have a more secure entry point for using Twitter's APIs. A bearer token can be created from the Twitter portal with a developer account and at least Essential access. In addition, CERTH applied for and obtained Elevated access (level of access to the Twitter API) from Twitter in order to have higher platform limits[9] on the Twitter API.

The Twitter crawler with the help of the Tweepy library forwards the bearer token created by the Twitter developer portal to the Twitter authorization service, which uses the OAuth 2.0 security model. More specifically, the OAuth 2.0 authorization framework allows a third-party application to gain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an authorization interaction between the resource owner and the HTTP service, or by allowing the third-party application to gain access on its own behalf. When the request to use the Twitter API is approved, the crawler establishes an open connection to the Twitter filtered stream endpoint. To collect tweets, the crawler must specify in the filtered stream endpoint the stream rule sets, the predefined search criteria (keywords/phrases, accounts) and the endpoint's query parameters[10]. In turn, the filtered stream endpoint will return the newly

[4] https://www.tweepy.org/

[5] https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/introduction

[6] https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/api-reference/post-tweets-search-stream-rules

[7] https://developer.twitter.com/en/portal/dashboard

[8] https://developer.twitter.com/en/docs/authentication/oauth-2-0/bearer-tokens

[9] https://developer.twitter.com/en/docs/twitter-api/getting-started/about-twitter-api

[10] https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/api-reference/get-tweets-search-stream

published tweets that match these rules, search criteria and contain the information that was specified in the query parameters.

Regarding the query parameters the crawlers utilizes four query parameters that are introduced below:

- **Tweet.fields**: This parameter allows the crawler to have access in information about content of the tweet (e.g., id, timestamp, lang, text, media).
- **Media.fields:** This parameter allows the crawler to have access in additional information about the media (e.g., media type, media URL, media id).
- **User.fields:** This parameter allows the crawler to have access in additional information about the user who posted the tweet (e.g., username).
- **Expansion:** This parameter allows the crawler to request additional data objects related to the original returned Tweets in each returned tweet object (e.g., A list of unique identifiers of the media that are attached in this tweet).

Table 7 presents the query parameter that the crawler uses and the fields selected to be returned in each tweet.

Query Parameter	Attribute Name	Description	Type
tweet.fields	id	A unique identifier for the tweet	String
	created_at	The time and date when the tweet was created	Date (ISO8601)
	lang	The language of the tweet	String
	text	The text of the tweet	String
	attachments	Specifies the type of attachments (if any) present in this Tweet (image, video)	JSON Object
	referenced_tweets	A list of Tweets that this Tweet refers to. For example, if the parent Tweet is a Retweet, it will include the related Tweet that referenced to by its parent.	Array
	entities	Contains tags that are extracted from the text and have special meaning	JSON Object
	possible_sensitive	Indicates if this tweet contains sensitive information	Boolean
media.fields	type	The type of the media contained in the tweet (animated_gif, photo, video)	String
	url	A direct URL to the media file on Twitter.	String
	media_key	A unique identifier for the media contained in the tweet	String

user_fields	name	The username of that the user defined in Twitter profile	String
expansions	attachments.media_keys	List of unique identifiers of the media attached to this Tweet	String
	referenced_tweets.id	The unique identifier of the referenced Tweet.	String

Table 7: Tweet's JSON attributes coming from Twitter API

As mentioned, the crawler retrieves data that match a set of stream rules that was applied in the filtered stream endpoint. A stream rule can contain one or more types of operators[11]. The crawler uses four types of stream rule operators that are presented below:

- **"Keyword"**: This operator fetches tweets that match a word within the text of a tweet (e.g., Retrieves tweets that contains the keyword "wildfire").
- **"Exact phrase match"**: This operator fetches tweets that match an exact phrase within the text of a tweet (e.g., Retrieves tweets that contains the phrase "forest fire").
- **"from:account"**: This operator fetches tweets that match a particular user (e.g., Retrieves tweets from a specific Twitter user1).
- **"lang:code"**: This operator fetches tweets that match classified by Twitter as being of a particular language (e.g., Retrieves tweets that the text is in English language).

Moreover, multiple stream rule operators can be concatenated together to create a single stream rule by the following tools:

- **AND logic**: A space between the operators will result in Boolean "AND" logic.
- **OR logic**: An OR between them will result in boolean "OR" logic.
- **Not logic, negation**: Prepend a dash (-) to a keyword (or any operator) it will result in boolean "NOT logic".
- **Grouping**: You can use parentheses to group operators together.

For example, the rule (wildfire ("windy day" OR "hot day")) (from:User1) (lang:en) will fetch tweets from the Twitter username @User1 in English language and contain the keyword "wildfire" and the exact phrases "windy day" or "hot day" in the tweet text.

Tweets are collected by the Twitter crawler from Twitter API as JSON strings. Each tweet is represented by a JSON string with the attributes shown in Table 7. For each newly received tweet the crawler applies a two-step pre-processing before it is stored in the database. The first part involves the extraction of the useful information of Twitter's initial JSON attribute values to a new polished JSON string. The second part of the processing of the original JSON involves the addition of attribute-value pairs that refer to the outcomes of the various knowledge extraction analyses or additional information that is beneficial for the Silvanus project. This action filters the excess and unnecessary information. Thus, the new JSON is smaller and needs less storage, which enhances the database's query retrieval speed.

In the first step of pre-processing the first attribute created in the new JSON string is the attribute *created_at* in the initial tweet JSON and it is transformed to the attribute *timestamp* in the new JSON. Additionally, from the attribute *referenced_tweets* in the initial tweet JSON the crawler can infer if the tweet is a quote or a retweet that are represented by the attributes *is_quote*, *is_retweet* in the new JSON. Also, from the attribute *lang* in the initial JSON converted to attribute *language* in the new JSON. Furthermore, the attributes *type* and *url* that are contained in the attribute *attachments* in the initial JSON,

[11] https://developer.twitter.com/en/docs/twitter-api/tweets/filtered-stream/integrate/build-a-rule

*c*ontains information about the media in Tweet, are modified to *media_type*, *media_url.* The attributes *id* and *text* from the original JSON remained the same in the new JSON too.

In the second step of pre-processing, a new attribute is added to the new JSON called *platform* in order to determine the platform from which the social media post originated, in this case “twitter”. In addition, the attribute *matched_criteria* is created in the new JSON to contain the keywords or the accounts that are found in the content of the tweet and match with the search criteria. The JSON object array attribute *extracted_locations* contains the detected locations found after the localization analysis (see section 5.1.4). The attribute visual_concepts is an array attribute that the labels of an image of a tweet are stored after the visual_concepts extraction analysis (see section 5.2.2). In the future all the results of the analysis that are occurring in the Social Media Analysis Toolkit will be in the new JSON in similar way.

Attribute Name	Description	Type	Example Value
id	A unique identifier for each tweet	String	"1474011868100448259"
platform	The platform that the post originates from	String	"Twitter"
timestamp	Date and time of the publication of the post, in the following format: YYYY-mm-ddTHH:MM:ssZ	String	"2022-09-07T15:00:00Z"
text	The text of the tweet	String	"The Bolt Creek wildfire in Washington state has grown to an estimated 7,600 acres"
is_quote	Whether the twet is a quote	Boolean	true/false
is_retweet	Whether the tweet is a retweet	Boolean	true/false
media_url	The URL(s) of the media that are attached in the tweet	String Array	["https://pbs.twimg.com/tweet_video_thumb/E-mfl0aXIAE72Jm.jpg", ...]
media_type	The type of the media that are attached in the tweet	String	"image"
language	The language of the tweet's text	String	"en"
visual_concepts	The concepts that can be extracted from the tweet's media	String Array	["fire", "person", "daylight", ...]
matched_criteria	A JSON object that contains the attributes	JSON Object	{"keywords": [" fire", "arson"], "account": "@user1"}

	"keywords" and "account"		
keywords	The keywords found in the text and matched with search criteria keywords	Array	
account	The account that posted the tweet and is matched with the search criteria accounts	String	
extracted_locations	The locations extracted from the post's text, represented as JSON objects with the attribute "placename", "geometry" and "crs"	JSON Object Array	[{ "placename" : "Triest, Trieste, Friuli-Venezia Giulia, 34121-34151, Italy", "geometry" : {"type" : "Point", "crs": "WGS84", "coordinates" : [{"lat": 13.7772781,"lon": 45.6496485}]}}, ...]
placename	The name of a location	String	
crs	The type of the coordinates reference system that is used.	String	
geometry	A JSON object with the attributes "type" and "coordinates"	JSON Object	
type	This always has value "Point" to support GeoJSON	String	
coordinates	The pair of the coordinates of a location contained in the attributes "lat" and "lon"	JSON Object Array	
lat	The longitute coordinates	Double	
lon	The latitude coordinates	Double	

Table 8: Tweet's JSON attributes that are stored in MongoDB

The newly formed JSON is stored in the MongoDB database after the two-step pre-process ends. There are two separate MongoDB collections, one for each Pilot: P01, P02, P03, P07, P09, P10 (Table 8). Thus, a check is performed after the crawling of a tweet, in order to identify for which of the Pilot the tweet has been crawled for based on the *matched_criteria* attribute of the new JSON. For example, if in the *matched_criteria* attribute the *keywords* field contains the word "bushfire" then it is inferred from the search criteria (Table 6) that the tweet was collected for P09. To sum up, all the attributes of the new JSON string are displayed in JSON format below.


```
[{
  "id": "1602861355400413184",
  "platform": "twitter",
  "is_retweet": false,
  "is_quote": false,
  "text": "An out-of-control bushfire is threatening lives and homes in Jurien Bay, with a huge effort underway to contain it. https://t.co/PUesYMcJJa https://t.co/HC1KkuYIle",
  "media_url": [
    "https://pbs.twimg.com/media/Fj6B6JWXoAIJkyo.jpg"
  ],
  "media_type": "image",
  "language": "en",
  "timestamp": "2022-12-14T03:01:41.000Z",
  "extracted_locations": [
    {
     "placename": "Jurien Bay, Shire Of Dandaragan, Western Australia, 6516, Australia",
      "crs": "WGS84"
      "geometry": {
        "type": "Point",
        "coordinates": [{
          "lat":115.0406027,
          "lon":-30.3040478
        }]
      }
    }
  ],
  "visual_concepts": [
    "Explosion_Fire",
    "Outdoor",
    "Clouds",
    "Sky",
    "Airport_Or_Airfield",
    "Military_Aircraft",
    "Vehicle",
    "Daytime_Outdoor",
    "Military_Base",
    "Airplane_Landing"
```

```
  ],
  "matched_keywords": {
    "keywords": [
      "bushfire"
    ],
    "accounts": []
  }
}]
```


Table 9: An JSON example of tweet stored in mongodb by the Twitter Crawler

Twitter Policy Risks

It is important to refer to the to the potential legal, financial, and reputational consequences that may arise due to the Twitter policies[12], like all social media platforms, must navigate a complex web of legal and ethical issues related to user-generated content, including hate speech, harassment, and the spread of misinformation. Failure to appropriately address these issues can result in a range of consequences, including lawsuits, regulatory fines.

For a Twitter crawler, the policy risk primarily involves the potential violation of Twitter's terms of service and the risk of account suspension or legal consequences. Twitter's terms of service prohibit automated access to its platform for the purpose of scraping or collecting data, unless the crawler has been authorized by Twitter through its official API. In our case all the data are collected through an endpoint of the official Twitter API.

Twitter's data storage policy, which dictates the amount of time data can be stored in a database, has no restrictions on how long tweets can be stored and analyzed. Previous guidelines limited companies to storing data for six months (Article I.F.2.b.ii), but this is no longer the case.

Twitter's data usage policy only authorizes pre-processed data obtained from the content of the tweets, such as text and images. All of the analyses performed in the Social Media Analysis Toolkit by CERTH are on the textual and visual content of the tweets, and no potentially harmful information is generated. Therefore, we comply with Twitter's policy.

To comply with European GDPR[13] regulations and avoid publishing any private information about Twitter users, such as addresses or personal information, CERTH does not provide, visualize, or store any user information such as usernames, addresses, or other personal data.

Overall, using a Twitter crawler can be in line with Twitter's policy risks as long as the developer follows the API guidelines and terms of service. Additionally, developers must use any data collected from Twitter responsibly and ethically, in compliance with applicable laws and regulations.

4.2 Implementation of Facebook Crawler

Within the SILVANUS project, Facebook is used as one of the sources of the social media activity (Figure 2). The choice of this platform was conditioned by several factors. First of all, it was the target language and geographic area of the pilot P02. Another important factor was the outcome of a representative survey carried out by Go4insight agency, which recognized Facebook as the most used social media network in Slovakia. The survey was conducted on a representative sample of 1,000 respondents aged 15-79 in March 2022. According to the survey, Facebook is used multiple times per day by more than 46% of the

[12] https://developer.twitter.com/en/developer-terms/agreement-and-policy

[13] https://gdpr.eu/

respondents. Twitter, the next social media network that enables sharing similar content as Facebook was reported to be used only by 1% of the respondents. YouTube and Instagram, which are relatively frequently used in Slovakia, are however platforms aimed at different content. While YouTube is a video sharing platform of various content, Instagram is a photo and video sharing application mostly for visual advertising.

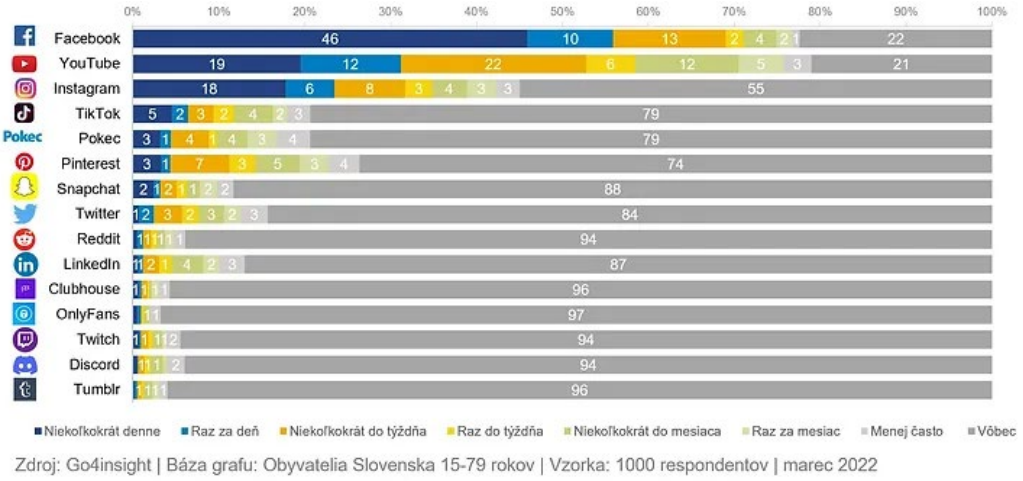


Figure 2: The most used social networks in Slovakia in 2022. Source: Go4insight

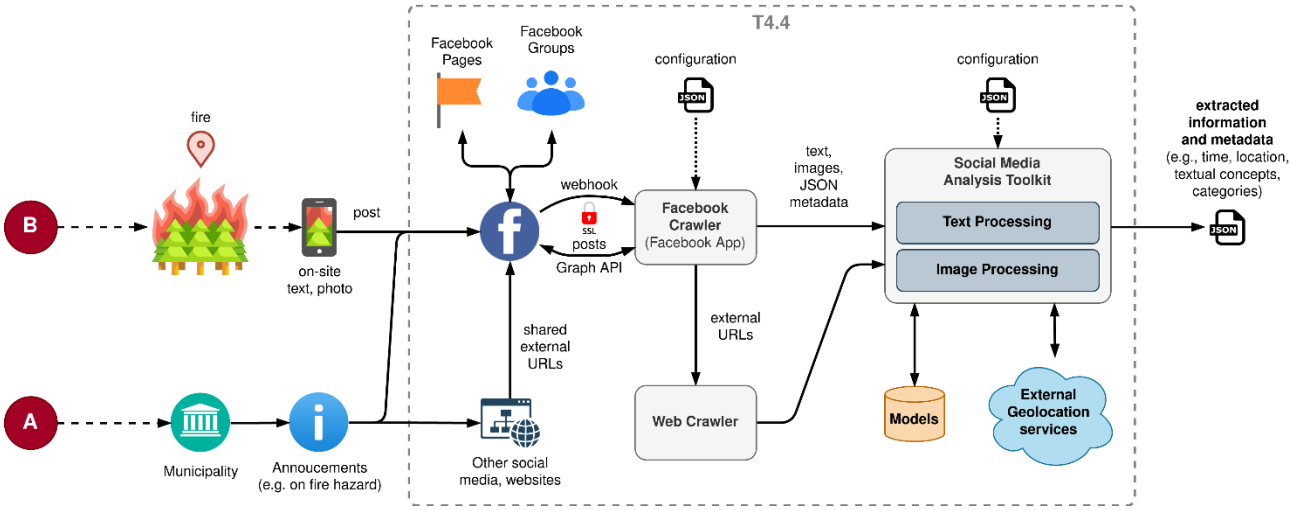


Figure 3: Facebook as a source of social media activity in the context of SILVANUS

Due to the latest strengthening of data and privacy protection rights and regulations (e.g. driven by GDPR), as well as due to technical limitations of classic crawling techniques with respect to Web 2.0, Facebook crawler is implemented as a Facebook App using the official Graph API. The role of the application is set to Business in order to be able to request certain permissions and use APIs required to read published posts. There is also a FORT Pages API[14], which allows approved Facebook Research Partners to get data about public Pages and public Posts on public Pages where steps have been taken to reduce the exposure of personal information. However, Facebook is not accepting new partner applications at this time.

In general, reading published posts can be allowed to the app either by authorization mechanisms or by specific permissions. Relevant authorization mechanism is the Page Public Content Access feature, which allows the application access to Pages Search API (for searching the pages; e.g., voluntary fire brigades) and public data for Pages (e.g. public posts and comments). The allowed usage for this feature is to provide aggregated, anonymized public content for competitive analysis and benchmarking. This feature enables to

[14] https://developers.facebook.com/docs/fort-pages-api/overview

read posts even when lacking explicitly granted permissions *pages_read_engagement* and *pages_read_user_content*, which are the second option to reading posts.

In order to fully exploit these capabilities a successful App Review process of the crawler application must be carried out by Facebook before it can access live data. In addition, the application must be put to live mode as well as installed into target groups by their administrators or it must gain explicit permissions from users and pages of which the crawler is reading posts. Successful review of the crawler application requires passing several conditions, which at this moment we are not fully covering. However, we have implemented a proof-of-concept solution as a test application that is capable of reading posts from a private group (SILVANUS_SK) to which we can re-share posts on fire incidents that were found in the Social Graph. The limitation is that the crawler can read posts re-shared by a single user, otherwise it is functioning as a live application.

The principle of the crawler integration and operation within the project is depicted in Figure 3 Crawler is deployed as a server listening to real-time notifications of changes to specific objects in the Facebook Social Graph called webhooks. In particular, the crawler is subscribed to notifications on a new post shared in a group or posted by a page or a user; i.e., Page/feed[15] and User/feed webhooks. Webhooks are sent using HTTPS and a valid TLS/SSL certificate is required for the crawler (self-signed certificates are not supported). The crawler is implemented in Python and uses facebook-sdk[16] library for making Graph API requests and processing responses.

Crawler operation

Prior to crawler operation, the crawler app must be either installed in the target Facebook groups by their administrators or the crawler must be explicitly allowed by pages or users to read their posts.

1. The crawler receives a notification in a JSON format (Figure 4) on a newly published post (Figure 5) through a configured webhooks: Page/feed and User/feed.
2. Reading a post using the schema in Table 10.
    a. Post ID is read from the notification and a new request to read the target post is send to Facebook using the Graph API.
    b. Detailed information on the post is received in a JSON format, however striped of personal data (e.g. post author).
    c. Additional information about the post is requested from the Social Graph; i.e., attachments such as URLs of the photos, external links, text of the original post.
3. URL fetcher retrieves the content of the external links if attached to the post.
4. retrieved data is aggregated in a JSON file (Table 11) and sent for text and image analysis.

[15] https://developers.facebook.com/docs/graph-api/webhooks/reference/page/#feed
[16] https://github.com/mobolic/facebook-sdk

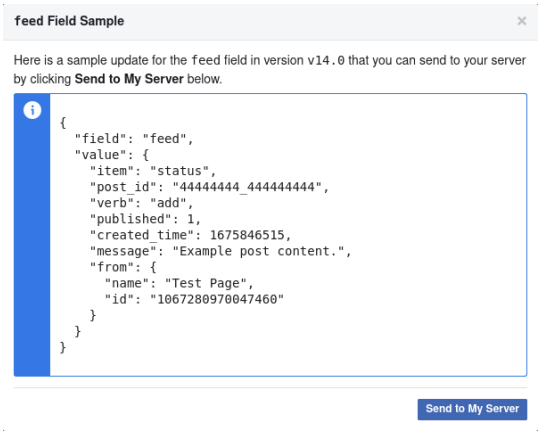


Figure 4: Sample notification payload

```
'post': {
  "id": "{id}",
  "fields":
"id,attachments,caption,created_time,description,from,full_picture,link,message,messa
ge_tags,name,object_id,place,shares,source,type",
  "edges": {
    "attachments": {
      "id": "{id}/attachments",
      "args": {
        "fields":
"description,description_tags,media,media_type,subattachments,target,type,unshimmed_u
rl,url"
      },
      "target": {
        "id": "{id}",
        "args": {
          "fields":
"id,album,alt_text,alt_text_custom,backdated_time,backdated_time_granularity,can_back
date,created_time,event,from,height,icon,images,link,name,name_tags,page_story_id,pic
ture,place,target,updated_time,width"
        }
      }
    },
    "comments": {
      "id": "{id}/comments",
      "args": {
        "fields":
"id,parent{id},created_time,from,message,message_tags,reactions,attachment",
        "order": "reverse_chronological",
        "filter": "stream",
        "summary": "1"
      }
    },
    "reactions": {
      "id": "{id}/reactions",
      "args": {}
    }
  }
}
```

Table 10: Schema for reading a post from the Facebook Social Graph

Attribute Name	Description	Type	Example (see Figure 4)
id	ID of the post	String	"581713980054014_730229218535822"
created_time	The time the post was initially published. Ref: https://schema.org/dateCreated	DateTime in ISO 8601 format	"2022-08-16T12:56:42+0000"

updated_time	The time when the post was created, last edited or the time of the last comment that was left on the post. Ref: https://schema.org/dateModified	DateTime in ISO 8601 format	"2022-08-17T03:43:46+0000"
message	The status message in the post. Ref: https://schema.org/articleBody	String	"VINEYARD FIRE I keep my fingers crossed for the firefighters fighting the vineyard fire near Pekna cesta. I believe that they will be able to put out the fire and that the nearby forest will not be affected."
caption	The caption of a link in the post (appears beneath the name).	String	"aktuality.sk"
description	A description of a link in the post (appears beneath the caption). Ref.: https://schema.org/description	String	"On Tuesday afternoon, firefighters battled flames in a Bratislava vineyard. The police closed the section between Pekna cesta and Vinohrady."
picture	A thumbnail of the picture scraped from any link included with the post. Ref.: https://schema.org/thumbnailUrl	URL	"https://external.fbts10-1.fna.fbcdn.net/emg1/v/t13/6221475771127048 97?url=https%3A%2F%2Fimg.aktuality.sk%2Ffoto%2Fpo-iar-vinohradu-v-bratislave%2FMTIwMHg2MzAvZmlsdGVyczp3YXRlcm1hcmsoaHR0cDovL2xvY2FsaG9zdDo4MS9pbWcvYWt0dWFsaXR5X3dhdGVybWFya192MiwyNCwyNCwwKS9odHRwOi8vbG9jYWxob3N0OjgxL3B1bHNjbXMtdHJhbnNmb3Jtcy8x%2FuxqktkpTURBXy8zNTE5NTI1ZWVjMjVlNWMxNzE1OWVmODY5ZmEwOTM0Mi5qcGeSlQMAzQGFzQOEzQH6lQLNB9AAwsM%3Fst%3DTSLv3o2-D4C48xL5JLr1OjRwvV91yy2LAyIh873bR3w%26ts%3D1660600800%26e%3D0&fb_obo=1&utld=aktuality.sk&stp=c0.5000x0.5000f_dst-emg0_p130x130_q75&ccb=13-1&oh=06_AbHt0wlU-yoSq-LlyhuoR3ULuCWNLmXdv6skO0h6NUJ_uw&oe=63E9D6CE&_nc_sid=6ac203"
full_picture	If the photo's largest dimension exceeds 720 pixels, it is resized, with the largest dimension set to 720. Ref: https://schema.org/image	URL	"https://external.fbts10-1.fna.fbcdn.net/emg1/v/t13/6221475771127048 97?url=https%3a%2f%2fimg.aktuality.sk%2ffoto%2fpo-iar-vinohradu-v-bratislave%2fMTIwMHg2MzAvZmlsdGVyczp3YXRlcm1hcmsoaHR0cDovL2xvY2FsaG9zdDo4MS9pbWcvYWt0dWFsaXR5X3dhdGVybWFya192MiwyNCwyNCwwKS9odHRwOi8vbG9jYWxob3N0OjgxL3B1bHNjbXMtdHJhbnNmb3Jtcy8x%2fuxqktkpTURBXy8zNTE5NTI1ZWVjMjVlNWMxNzE1OWVmODY5ZmEwOTM0Mi5qcGeSlQMAzQGFzQOEzQH6lQLNB9AAwsM%3fst%3dTSLv3o2-D4C48xL5JLr1OjRwvV91yy2LAyIh873bR3w%26ts%3d1660600800%26e%3d0&fb_obo=1&utld=aktuality.sk&stp=dst-emg0_q75&ccb=13-1&oh=06_AbGgQB_x_wn4HxQKK6nOBBJCHrTuPVPQ4WKzVUIOJPnEaA&oe=63E92E0E&_nc_sid=5f3a21"

link	The link attached to this post.	URL	"https://www.aktuality.sk/clanok/eR7ry4Q/poziar-plamene-zachvatili-vinohrad-v-bratislave/"
name	The name of the link, if attached to the post. Ref: https://schema.org/name	String	"Fire: Flames engulfed a vineyard in Bratislava"
permalink_url	The permanent static URL to the post on www.facebook.com. Ref.: https://schema.org/url	URL	"https://www.facebook.com/735832858152841/posts/588660979536697"
object_id	The ID of any uploaded photo or video attached to the post.	String	"231489075684458"
shares	Number of times the post has been shared.	JSON Object	"{'count': 6}"
type	A string indicating the object type of this post; i.e. link, status, photo, video.	String	"share"
status_type	Description of the type of a status update; i.e. mobile_status_update, created_note, added_photos, added_video, shared_story, created_group, created_event, wall_post, app_created_story, published_story, tagged_in_photo	String	"shared_story"

Table 11: Schema for reading a post from the Facebook Social Graph

Figure 5: Sample public post posted by a member of the city council for the Municipality of Nové Mesto, Bratislava.

4.3 Implementation of Web Crawler

The developed web crawling platform for SILVANUS project purposes is structured as a modular system designed to perform two main operations: web harvester to obtain digital multimedia content (e.g., digital images, videos, text) and inspecting the downloaded content with various analytical tools. Moreover, it is able to execute these activities for a significant volume of data in a distributed manner, i.e., separate tasks are accomplished concurrently by the Workers placed on the different nodes. From the logical perspective, the architecture is composed of two main parts: Front-End and Back-End. The Front-End is a Web Application that allows the user to interface with different modules:

- *Site scraping* ('Harvester') module responsible for downloading digital multimedia content.
- *File analysis* ('Analyzer') module, which role is to perform various types of analyses on the content obtained by the Harvester.
- *Flower* supervisor for *Workers* monitoring and control.

On the Back-End, there are: a task scheduling mechanism, Worker modules, Database, and Network File System (NFS) used to store the obtained multimedia files, as well as auxiliary support modules, which will be described later in this paper. Below we present details on the most important components of the developed web crawler.

Front-End

The Web Application serves as a management plane for the end-system user. It is an application written in Python running on an Apache webserver with WSGI extensions enabled to allow the processing of Django-generated content. It allows the ordering of Harvester or Analyzer tasks, which are then performed on the Back-End by specialized Workers based on Celery.

By default, upon opening the Web Application, a Flower supervisor (a web-based control interface for the Celery-based task Workers) is shown, which provides a status overview pane, displaying the number, as well as the current status of workers and tasks that are currently being processed. It also provides the ability to stop a given worker if that becomes necessary.

The Web Application provides four main functions to the system user, who can:

- *Start browsing*: this allows displaying stored websites, tabulated information about files stored on the Back-End, all files selected for analysis, and a brief overview of previous analyses, with the option to display more details for a given file.
- *Setup web crawler*: to perform harvesting on the list of websites desired by the user, firstly the information about this website needs to be stored in the developed system. Then, tasks for Harvester Workers can be ordered for the websites selected from the system. Thus, there are two main options that can be used: (i) the addition of a new website to the system if the desired one is not present or (ii) scheduling file scraping from a given website. To add a new website to the system, the user must first prepare a semicolon-separated CSV file containing a list of website names (by which the site will be identified to the user in the system), as well as a start URI (the website, where the image collection is to begin from). To schedule a Harvester Worker task, a CSV file must be uploaded containing a website name (as stored in the system) with one site per CSV line. At this stage, the Harvester module of the Front-End parses the user input, then the tasks are prepared and passed on to the Back-End for processing.
- *Select file(s):* this part of the system allows the user to choose files previously obtained by the Harvester that will be subjected to the analysis. The selection of files can be performed incrementally to create the desired subset of all multimedia files stored in the system. The results of filtering can be used as input to the Analyzer module or serve as a base for further refinement of the query. In the latter case, the filtering steps can be repeated many times, using any combination of filters, until the desired set of files for processing is obtained by the user. Currently, there are four available filtering methods: by file extension, file size, file ID in the Database, or acquisition date. After selecting the desired file set, the Analyzer part of the Web Application can be accessed.
- *Setup Analyzer*: it enables the user to select a desired analysis from the list and order the analysis to start. The system offers various types of analyses, implemented as plugins. Each plugin is responsible for one analytical algorithm. Additionally, during setup, the plugin that will be used for file analysis needs to be selected. At this stage, the Analyzer module is responsible for the initial processing, and creation of tasks to be executed by the available Workers, which are then put into a task queue to be run when resources are available. Upon completion of the processing, the results of the performed analyses are stored in the Database.

Back-End

The main components of the Back-End are: the Celery task distribution module, Harvester, and Analyzer Celery-based Workers, and data storage containing PostgreSQL Database and NFS.

Celery is a task execution mechanism that allows for tasks to be distributed and run across multiple nodes, and it is most commonly implemented in Python. In addition, it has been designed so that any task can be run on any number of remote nodes without the need to manually control the task assignment process, scheduling, or reporting back to the result database. This framework is used extensively in our task distribution module, which consists of two main components:

- *Redis Database*, which is responsible for storing the submitted tasks, until there are Workers available to process them.

- *Task Broker* - Celery tasking module, which assigns tasks to available Workers of a given type (Harvester or Analyzer). It is communicating with Celery Workers over the Internet. Celery modules are also in constant communication with the Flower supervisor for control and monitoring purposes.

The control node of the developed system stores the Web Application, task queue, and Database. On the contrary, the Celery Worker code is stored and executed on a Worker node. All nodes are also connected to the NFS to store and access downloaded multimedia files. Currently, the whole system consists of two machines. The first one plays the role of the controlling node, but also of the small node executing Workers' tasks (i.e., the role of the Worker node). The second one functions as a purely remote Worker node and allows running many more Harvester or Analyzer Workers at the same time.

A Worker node is a Linux-based machine running Celery Worker modules responsible for executing Python code for either harvesting files from websites or running the analytical plugin code. As mentioned earlier, there are two types of Workers: Harvester Workers and Analyzer Workers. For the best performance, the worker types should not be mixed across the same node at run time.

The Workers can process tasks asynchronously with respect to one another, acquiring the tasks and returning the results directly to a PostgreSQL Database and in the case of Harvester Workers, additionally storing the acquired files on the NFS. A task that is successfully executed is erased from the Redis Database upon receiving a completion signal from the Celery Task Broker. If the Worker abnormally terminates, the task is reassigned to any other available Worker and restarted.

The Workers are instances of standalone programs written in Python executed per task on a given node. API provided by the Celery framework is used to run the code of Harvester and Analyzer Workers as Celery tasks. The exact number of Workers depends primarily on the number of available CPU cores and RAM available on a given host. Details on how the Harvester and Analyzer Workers are built are described next.

Harvester Workers

The Harvester Worker is responsible for accessing web pages, downloading digital multimedia content, and storing the content in our system. More precisely, the Harvester is assigned a task to begin the crawling process on the given starting URL address. Then, it has to recursively follow all links available on the scraped website, up to the given (configurable) depth limit. The multimedia files and their metadata from all accessed URLs are saved in the system.

To implement automated web crawling, we decided to use the Scrapy framework. Scrapy is a highly efficient and tested web scraping tool, capable of resolving various types of web content via the use of an XPath language. This framework has many desirable features, such as high modularity, the ability to smartly adjust the rate of request to avoid flooding the queried system with requests, or HTTP-aware navigation, which includes automatic resolution of relative URLs and automatic handling of invalid URLs or requests. All the above-mentioned functionalities are performed with very little system overhead, allowing for the efficient scraping of many websites without overloading the system.

However, most modern websites typically employ JavaScript to generate HTML code, including links to other web pages and multimedia files, and cannot be gathered correctly without rendering the whole contents. Due to this fact, we also added Splash to perform JavaScript execution before Scrapy reads the page content. Splash is implemented in LUA, a flexible JavaScript-aware proxy used for page rendering, and invoked as HTTP API requests.

Splash is a single-core application and, therefore, lacks load-balancing capabilities. As a result, it can be easily overloaded with many concurrent requests from Worker processes. However, to correctly handle multicore workloads, such as many parallel Harvester Workers needed in our system, it requires multiple

load-balanced instances. To solve this issue, multiple instances are run in parallel and virtualized into a single instance by a highly stable and performant proxy broker, i.e., HAProxy.

Moreover, sufficient headroom must be provided for Splash proxy instances so that they are not overloaded with too many concurrent requests. As the documentation suggests, a good starting ratio is a 50% allocation, which can then be adjusted after running system stress tests in a particular deployment scenario. Splash instances, as well as HAProxy broker, are run under the Docker composer modules. All three core modules used by the Harvester Worker, i.e., Scrapy-based harvesting mechanism, Splash, and HAProxy mediator.

As mentioned above, the implementation of web crawling by the Harvester Worker uses extensively the Scrapy framework. It is designed in a modular way, as the Scrapy documentation suggests. Consequently, the Scrapy module of the Harvester Worker consists of three integral parts cooperating with each other:

- *Scrapy spider*: responsible for crawling sites on the basis of the given starting URL, interfaces with the Splash renderer via an endpoint exposed to it by HAProxy. The spider is capable of smartly extracting desired elements from the page, which in our case means digital multimedia files of various types and links to other web pages. Once a multimedia file is identified by the Scrapy spider, it is passed on to a second submodule (i.e., the file pipeline). Moreover, the spider follows the extracted links to other websites recursively up to the set depth limit.
- *Scrapy file pipeline*: it is responsible for file classification, rejection of unwanted files, as well as storage of files on the NFS. Lastly, the file metadata is stored in the database as reflected in the Django ORM.
- *Scrapy-Django interposer*: it is a representation of metadata obtained by Scrapy, which is then mapped to Django ORM objects. It is responsible for proper integration with Django ORM, thanks to which the information extracted by Scrapy can be saved directly into the Database.

Analyzer Workers

The job of the Analyzer Worker is to perform the selected type of analysis on the given multimedia file. This responsibility is modularized in two ways. Firstly, there is a plugin mechanism, which allows to easily extend the set of offered types of analyses. A plugin is mainly an implementation of an algorithm performing a given type of analysis on the multimedia file. It can be either: (i) our own implementation of some algorithm written in Python and coupled with the rest of the system, (ii) the code that only executes some external algorithms available in a rich library of Python modules and is capable of storing the results in our Database in the correct format or, (iii) completely external program (which must be compatible with the system architecture of the host machine) which is only invoked from our platform with a set of necessary command-line arguments in a newly created process and which results are then intercepted by our system. In this case, the program can be written in any programming language and there is no need to have its source code. The plugin only executes the binary and collects its results. The exact output depends on the invoked program, so in every case, it is necessary to decide how the plugin will collect the results from the external binary. The data can be obtained from either the standard output of the program or from the files (e.g., CSV) created by this program. In some cases, it is useful to use both sources, as each of them contains a bit different information.

In every case, the plugin is responsible for acquiring the multimedia file metadata from the Database and the file itself from the NFS. Then, the plugin runs the given analytical algorithm (either our own implementation, library code, or an external binary) on the given digital media file. Finally, it has to translate the output of the program into a common format understood by the rest of the system and store the results in the Database.

There is also a second way in which the analytical part of the developed system is modularized. The analysis of each separate file with one selected plugin (indicated by the user on the Analyzer setup) is directed to

the Analyzer Worker and processed as a separate task, i.e., each Worker task is responsible for performing one given type of analysis on one given file. Both solutions mentioned above, that is, the plugin mechanism and the fact that each analysis for each file is executed separately, allow for easier scaling of the developed platform.

Operational Flow

The control flow between different components of the harvesting system consists of five main steps which are briefly described below:

Step 1: The user orders either a harvest or analysis work on the system. The input from the user in the correct form (i.e., CSV with websites to be added, CSV with websites to be harvested, or files to be analyzed along with the plugin which is to be used for analysis) is read by the Web Application. The Harvester or Analyzer submodules of the Web Application prepare and put the tasks in the Redis Database.

Step 2: Tasks are waiting in the Redis Database until there is a ready Worker of the correct type able to process them. Then, they are assigned to the first available Worker dependent on the type of work needed to be done (so a Harvester Worker will not accept the Analyzer type of tasks and vice versa).

Step 3: Once a Worker is available, it consumes the task.

Step 4: Depending on the type of task: (i) if a Harvester task is required, the Scrapy spider (via Splash and HAProxy support modules) acquires the URI from the task argument and scrapes the target domain for multimedia content recursively. Upon completion, if successful, the downloaded files are stored in the NFS, and their metadata are saved in the PostgreSQL Database, if not, the task terminates without introducing changes to the database or the NFS; (ii) if an Analyzer task is required, the matching multimedia files are read from the NFS on the records in the PostgreSQL Database, and the results are stored in the same Database.

Step 5: Upon completion of the task, Celery Task Broker and Flower supervisor are signaled that the task is finished. The task is removed from the Redis Database.

In the case of an abnormal condition occurring during task execution, for instance, the crash of one or more of the Worker nodes, the Flower supervisor, in conjunction with the Task Broker, reassigns all tasks of the said Worker to another available Worker instance automatically (if available; otherwise the task is held in a queue until a Worker becomes available). It is worth mentioning that tasks explicitly terminated via the Flower supervisor are not reassigned, as the action is taken upon explicit user input only, and thus interpreted as an intended abort.

The most important advantage of the developed system is its flexibility. Firstly, the crawling and analyzing processes can be distributed across many machines, which allows performing many tasks concurrently in order to speed up the whole procedure (either harvesting, analyzing, or both). Secondly, the analytical part of the system is modular and easily extendable. This is possible thanks to the plugin mechanism, which provides an easy way to incorporate various analytical algorithms from different sources into our system.

5 Social Media Analysis Toolkit

This section provides a comprehensive overview of the analyses carried out by the Social Media Analysis Toolkit using textual (see section 5.1) and visual (see section 5.2) analysis components. In each component is described the methods that are used and the input and output of the analysis.

The collected social media posts from Twitter, Facebook, and Web crawlers are pushed via individual API calls to the components of the Social Media Analysis Toolkit. These components meticulously analyze the visual and textual content of each post, augmenting the social media post metadata with enhanced knowledge. This enriched metadata can then be utilized by the Fire Events Detection module for the detection of fire events and to provide valuable insights to SILVANUS Dashboard users.

5.1 Textual Analysis

This subsection provides an overview of the textual analysis provided by the Social Medi Analysis Toolkit that process textual information collected by the Social Media Crawlers. The analysis methods employed by these components are as follows:

I. Relevance Classification and Text Categorization: Involves categorizing social media posts based on their relevance to the topic of interest and classifying them into different categories based on their content (see section 5.1.1).
II. Textual Concepts Extraction: Extracts meaningful concepts and entities from the text of social media posts, such as names of people, organizations, and locations, and other relevant information (see section 5.1.2).
III. Event Recognition: Identifies significant events mentioned in social media posts, e.g., as fires (see section 5.1.3).
IV. Location Extraction: Extract location information from social media posts text (see section 5.1.4).

5.1.1 Relevance Classification and Text Categorization

5.1.1.1 *Relevance Classification - Text categorization and concept mining*

HB had two anticipations, the first being cross-border fire events where reliably cross-lingual solutions are called in, and methodology selection with linkage to D5.4 Semantic information fusion framework (M30) as well. Both expectations were met by the experiments we present below.

Because there was a certain overlap between the tested approaches, we did not strictly distinguish between relevance classification, text categorization and concept mining. In our eyes, relevance classification pertains to the binary separation of relevant vs. non-relevant subsets of the incoming news in social media; text categorization precedes relevance assessment, excludes images but builds categories within the relevant subset based on their keyword content; and finally, concept mining identifies those hypernyms which collect a group of related keywords to build categories from.

HB carried out two sets of relevance classification experiments on altogether 5 social media datasets in English, Greek, Italian and Spanish, all provided by CERTH (Table 12). Results from the first round were reported at the first consortium meeting in Bari in July 2022, with those from the second one provided for the second meeting in Athens in December 2022. All the code and the corresponding output have been retained for integration into future workflows.

Project	Language	Start date	Number of tweets	Geotagged (yes/can-be/ no)	Manually annotated as relevant
beAWARE	Danish	17-jun-17	583	no	0
beAWARE	English	18-jun-17	81818	can-be	0

beAWARE	Greek	16-jun-17	425960	can-be	3871
beAWARE	Italian	16-jun-17	724899	can-be	1904
beAWARE	Spanish	16-jun-17	15351354	can-be	4073
INGENIOUS	Greek	10-sep-20	23032	yes	5742
INGENIOUS	English	17-mar-21	942295	yes	0

Table 12: Datasets from the beAWARE[17] and INGENIOUS[18] projects

The strategy we followed was testing the preferred methodology for feasibility first, leaving the evaluation of the results to the second period. The reason for this was that the initial set of test data did not contain relevance assessments, allowing for unsupervised machine learning of semantic content only. When we received datasets in Greek, Italian and Spanish, augmented by user feedback about positive and negative examples, this bottleneck was eliminated, and the standard ranking of tweet classification results became possible.

We note in passing that multilingual input created a new bottleneck as compared with machine learning on English texts only, but the technology applied, BERT (Devlin et al., 2018), allowed for a resolution. Also, a major development was the successful use of multilingual sentence embedding to train a relevance classification model in one language (for which relevance annotations exist) and apply to another language. This approach, which could be said to be an application of transfer learning, is based on the use of the Sentence-BERT framework (Reimers and Gurevych, 2019). A respective SILVANUS publication (Eklund and Daránvi, 2023) is in progress.

Results

In the starting setup, we had encouraging first results on a 2K sample as the random representation of the CERTH INGENIOUS dataset of fire-related tweets (950 K), in English, without relevance feedback. These results were therefore manually assessed. We applied a two-pronged approach, called Track A vs. Track B, as follows:

Track A:

(a) Semantic embedding of tweets, considered as short sentences, by Multilingual Sentence BERT[19]. After some parameter tweaking for optimization, encoded tweets were applied to train a Support Vector Machine (SVM) classifier. This experiment, using an 80/20 split of the data, resulted in a recall of 0.9661 and precision of 0.8680 (hence, the F1 score was 0.9144), with the approach being considered as promising.

(b) Labelling located semantic content about wildfires only by k-means clustering (MacQueen, 1967);

(c) Topology-preserving dimension reduction for visualisation by Uniform Manifold Approximation and Projection (UMAP)[20] (McInnes and Healy, 2018). UMAP is based on the assumption that data "resides" on a manifold in vector space and utilizes the existing data to get an impression of the structure (topology) of the manifold. It then tries to preserve the structure (topology) as well as possible in spaces of lower dimensionality (Figure 6,Figure 7).

Track B:

[17] https://beaware-project.eu/
[18] https://ingenious-first-responders.eu/
[19] https://www.sbert.net/
[20] https://umap-learn.readthedocs.io/en/latest/

Underlying this line of thought, the observation of topic outbursts goes back to (Kleinberg, 2002). After having also checked out two other methods, Top2vec[21] (Angelov, 2020) and TopSBM[22] (Gerlach et al., 2018), we decided for Topic Modelling (TM) by Latent Dirichlet Allocation (LDA) (Blei et al., 2003) (Figure 8). TM could reliably identify wildfires as the central concept in the dataset. In future work, by extracting changing compositions of index terms in document sets characterized by some progressive feature such as geographical location or timestamps, one can characterize event progress as a series of topic outbursts by Dynamic Topic Modelling (DTM) (Blei and Lafferty, 2006).

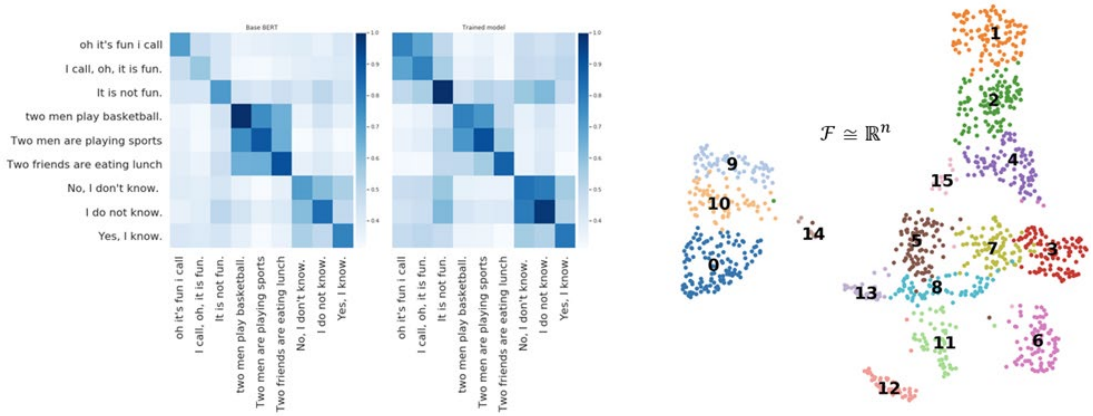


Figure 6: From SBERT to UMAP: Semantic embedding of tweets, dimension reduction

Figure 7: From SBERT to UMAP: Tweets clustering

[21] https://github.com/ddangelov/Top2Vec
[22] https://topsbm.github.io/

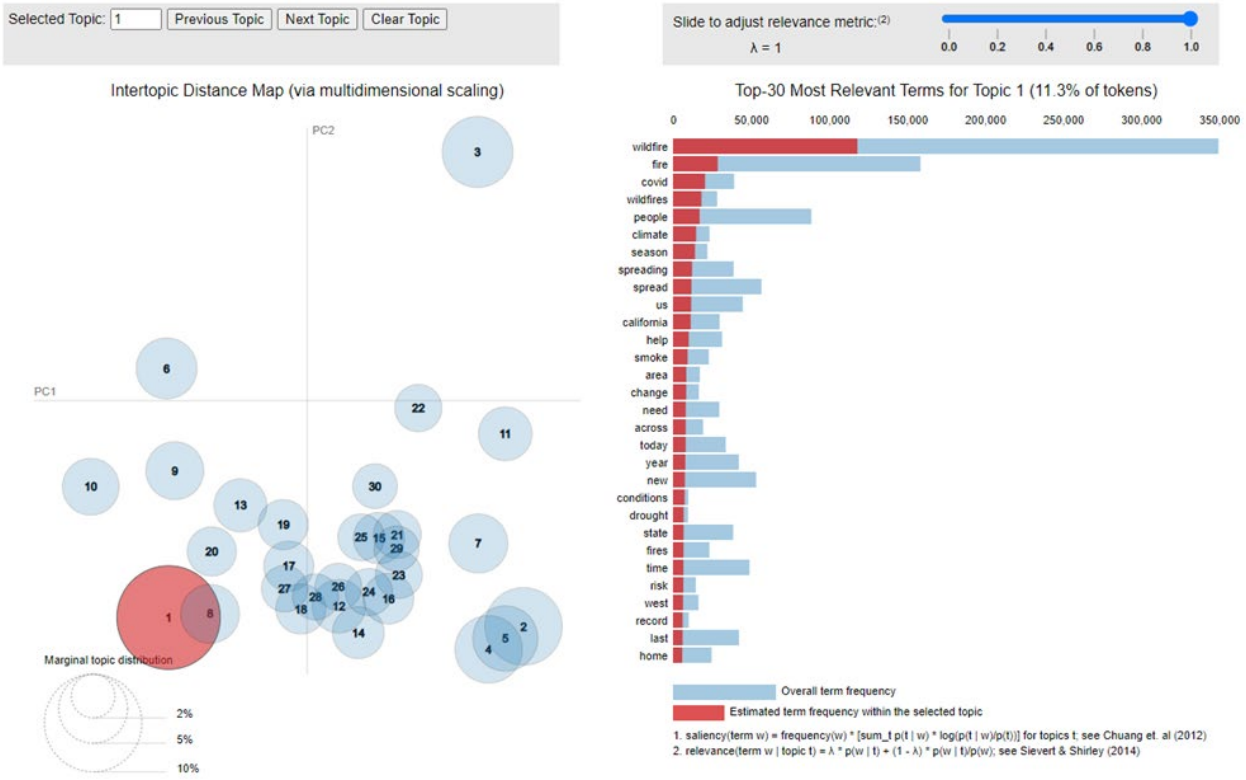


Figure 8: Topic maps result

In the second round of experiments, with four new user annotated datasets as input in Greek, Spanish, and Italian, we focused on the performance of different algorithmic combinations in Track A. The goal was to find out how classification by supervised learning vs. clustering by unsupervised learning perform on the test data. To this end four combinations of feature engineering and classification algorithms were employed with different parametrization, plus two combinations of feature engineering with an experimental flat clustering algorithm. The two ways of feature engineering were TFIDF vs. SBERT. All the classifications were evaluated by means of 5-fold cross-validation and the standard F1 measure, whereas for clustering the Rand index[23], a similarity measure between two clusterings was employed.

XGBoost[24] (Chen and Guestrin, 2016) is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. The applied clustering method, Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)[25] (Campello et al., 2016; McInnes and Healy, 2017), excels in situations where one can expect arbitrarily shaped clusters with different sizes and densities, with noise.

Dataset: INGENIOUS_Greek (N = 5779)

Classification

	TF-IDF + linear C-SVM (C = 5, k = 300)	**SBERT + linear C-SVM** **(C = 50, k = 202)**	SBERT + Gaussian Naive Bayes	SBERT + XGBoost[26]

[23] https://en.wikipedia.org/wiki/Rand_index
[24] https://github.com/dmlc/xgboost
[25] https://pberba.github.io/stats/2020/07/08/intro-hdbscan/
[26] https://github.com/dmlc/xgboost

Avg. recall (5-fold)	0.6765	**0.7470**	0.7342	0.6981
Avg. precision (5-fold)	0.6885	**0.7491**	0.6992	0.7270
Avg. F1 (5-fold)	0.6786	**0.7455**	0.6979	0.7080

Table 13: Classification results from the INGENIOUS_Greek dataset

Clustering (evaluation by Rand index)

TF-IDF + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.6126

SBERT + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.6136

Dataset: beAWARE_Italian (N = 1904)

Classification

	TF-IDF + linear C-SVM (C = 10, k = 300)	**SBERT + linear C-SVM (C = 100, k = 208)**	SBERT + Gaussian Naive Bayes	SBERT + XGBoost
Avg. recall (5-fold)	0.9173	**0.9413**	0.8748	0.8968
Avg. precision (5-fold)	0.9238	**0.9383**	0.8696	0.9083
Avg. F1 (5-fold)	0.9157	**0.9381**	0.8597	0.8956

Table 14: Classification results from the beAWARE_Italian dataset

Clustering (evaluation by Rand index)

TF-IDF + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.5484

SBERT + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.5591

Dataset: beAWARE_Spanish (N = 4073)

Classification

	TF-IDF + linear C-SVM (C = 10, k = 300)	**SBERT + linear C-SVM (C = 100, k = 209)**	SBERT + Gaussian Naive Bayes	SBERT + XGBoost
Avg. recall (5-fold)	0.7220	**0.7372**	0.6812	0.7022
Avg. precision (5-fold)	0.7442	**0.7530**	0.6802	0.7394
Avg. F1 (5-fold)	0.7239	**0.7398**	0.6696	0.7083

Table 15: Classification results from the beAWARE_Spanish dataset

Clustering (evaluation by Rand index)

TF-IDF + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.5533

SBERT + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.5533

Dataset: beAWARE_Greek (N = 3871)

Classification

	TF-IDF + linear C-SVM (C = 5, k = 300)	SBERT + linear C-SVM (C = 50, k = 212)	SBERT + Gaussian Naive Bayes	SBERT + XGBoost
Avg. recall (5-fold)	**0.8464**	0.8446	0.7990	0.8346
Avg. precision (5-fold)	**0.8584**	0.8522	0.8074	0.8476
Avg. F1 (5-fold)	**0.8447**	0.8437	0.7981	0.8327

Table 16: Classification results from the beAWARE_Greek dataset

Clustering (evaluation by Rand index)

TF-IDF + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.5011

SBERT + UMAP + HDBSCAN (experimental flat clustering, k = 2): 0.5011

Our findings were as follows:

- Regardless of parametrization, classification constantly outperformed clustering. Currently there is no supporting evidence that clustering can be used to identify relevant tweets, the practical implication of this being that relevance assessment of tweets needs manually annotated training data to work.
- A well-tuned Support Vector Machines (SVM) algorithm still beats the other algorithms tested. Further, out of the four classification experiments, SBERT outperformed TFIDF in three cases.
- The best result from the beAWARE_Italian dataset was F1 = 0.9381 at N = 1904. We can expect that the amount of data is not problematic per se but certain imbalances in the class distribution may become more pronounced in larger datasets. This, in turn, can be counteracted with resampling.

5.1.1.2 *Facebook Post Analyser*

The dataset (Figure 9) used for training and evaluating analysis models for Facebook posts in Slovak language contains 422 unique labelled public posts. The posts were collected manually due to several restrictions regarding the automated access to Facebook data. Automated access is not impossible; however, it requires an implemented business application to demonstrate the data processing and thus passing the app review process. There are also other conditions that apply, which make this process complicated and time consuming (e.g., registration and verification of a business account, video for review creation). In addition, there is Facebook Open Research and Transparency (FORT) programme, which provides academics and independent researchers with the tools and data, but application to the programme is postponed and available datasets are focused on topics out of the SILVANUS scope; i.e., ad targeting transparency and URL shares. Therefore, before fulfilling all the conditions and having the app ready and live, a simpler approach to accessing the data was taken: an unverified business app was implemented. This app had the same functionality as it would have if it successfully passed the review, but it lacked permissions to read public posts of other users and pages. However, it could read the posts published or reshared by the app developer. Therefore, a private Facebook Group Silvanus_SK was created

and the app was installed in it. This group was used to accumulate posts by resharing existing relevant public posts from Facebook's Social Graph. As a search query to the Social Graph, keyword “poziar” was used, which means “fire incident”. It is very general and yet the most relevant keyword regarding fire incidents in Slovak language. The results were restricted to post type only, and reshared to the SILVANUS_SK group from which they were downloaded by the app taking into account privacy protection rights as well as terms of service.

	id	text	type	location	IDN4	label
0	581713980054014_730229721869105	Nočný požiar strechy rodinného domu v obci Boj...	photo	Bojničky	506800	[urban_fire]
1	581713980054014_730229298535814	19.10.2022 Zásah požiar rodinného domu v obci...	photo	Mýtna	511641	[urban_fire]
2	581713980054014_730229218535822	POŽIAR VINOHRADU Držím palce hasičom bojujúcim...	link	Bratislava-Rača	529354	[wildfire]
3	581713980054014_730228998535844	Požiar sa rozhorel v byte na siedmom poschodí!...	link	Trenčín	505820	[urban_fire]
4	581713980054014_730228841869193	Zásah č.10/22 10.11.2022,9:45hod Miesto: Žehra...	photo	Žehra	526657	[urban_fire]
6	581713980054014_730228698535874	POŽIAR RODINNÉHO DOMU V STUPAVE Vo štvrtok, 15...	photo	Stupava	508233	[urban_fire]
7	581713980054014_730228575202553	!!Aktualizácia!! Požiar v byte na Prostějovske...	photo	Prešov	524140	[urban_fire]
8	581713980054014_730228475202563	Požiar v Krškanoch - areál IDEA, firma s odpad...	photo	Krškany	502430	[urban_fire]
9	581713980054014_730228008535943	Dňa 31.12.2022 nám bol nahlásený požiar chaty ...	photo	Úhorná	543683	[urban_fire]
10	581713980054014_730227715202639	Aktuálna situácia na Hnileckej ulici ⚠⚠ Ako ...	video	Dobšiná	525634	[urban_fire]
11	581713980054014_730227625202648	📌POZOR NA ZÁBAVNÚ PYROTECHNIKU! DHZO Zbyňov d...	photo	Zbyňov	518131	[urban_fire]
12	581713980054014_730227518535992	Včerajší požiar rodinného domu v obci Budimír ...	photo	Budimír	521221	[urban_fire]
13	581713980054014_729716111920466	18.10.2022 Požiar obytného karavanu v obci Mýtna.	photo	Mýtna	511641	[vehicle_fire]
14	581713980054014_729715995253811	🔥 AKTUÁLNE: POŽIAR V PETRŽALSKOM DOME SENIOROV...	photo	Bratislava-Petržalka	529460	[urban_fire]
16	581713980054014_729715595253851	NOČNÝ POŽIAR V BYTOVOM DOME SI VYŽIADAL DVE OB...	photo	Prešov	524140	[urban_fire]
17	581713980054014_729715358587208	TRANSPARENTNÝ ÚČET 🙏🔥🏠 Rodina Hovancová zriadi...	photo	Budimír	521221	[urban_fire]
18	581713980054014_729715165253894	Dnes v popoludňajších hodinách vypukol požia...	photo	Veľký Krtíš	515850	[urban_fire]
19	581713980054014_729715095253901	Veľký požiar nám zničil celú prevádzku v Ružin...	video	Bratislava-Ružinov	529320	[urban_fire]
20	581713980054014_729714958587248	POŽIAR RODINNÉHO DOMU: VŠETKO ZHORELO DO TLA D...	photo	Demjata	524336	[urban_fire]
22	581713980054014_729714488587295	AKTUALIZOVANÉ 23.55 Milí susedia, mnohí sa pýt...	video	Bratislava-Ružinov	529320	[urban_fire]
25	581713980054014_729713985254012	Dnes o 2.hodine ráno bol našej jednotke nahlás...	photo	Osuské	504602	[urban_fire]
26	581713980054014_729713891920688	SPOJME SA PRE DOBRÚ VEC A POMÔŽME ❁ Sú sprá...	photo	Ružomberok	510998	[urban_fire]
27	581713980054014_729713771920700	Požiar skládky odpadu medzi obcami Rumanová-Ri...	photo	Rumanová	500712	[urban_fire]
28	581713980054014_729713678587376	V BANIČNOM HORÍ RODINNÝ DOM 🚒🚒🔥 !! (aktualizác...	photo	Ružomberok	510998	[urban_fire]

Figure 9: Dataset preview

The dataset was manually annotated. The annotation was divided into two tasks: post categorization and named entity recognition. A collaborative annotation tool Doccano[27] was used for both tasks (Figure 10).

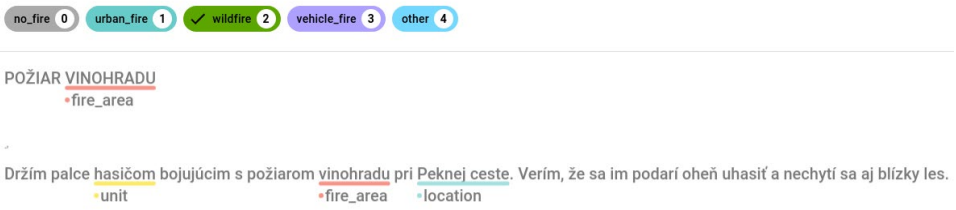


Figure 10: An example of an annotated post

Attribute Name	Description	Type	Example
extracted_locations	The locations extracted from the post's text, represented as JSON objects (see below)	JSON Object Array	`[ { "mention": "Pekna cesta" "placename": "Bratislava-Raca", "crs": "EPSG:4326", "geometry": { "type": "Polygon",`
mention	How the extracted location is mentioned in the original text	String	

[27] https://github.com/doccano/doccano

	placename	The name of a geocoded location; could be an area containing the original location.	String	<pre>"coordinates": [
				[
				[
				-570668.1600000001,
				-1270095.6799999997
				],
				[
				-570000.5,
				-1270060.3900000006
				],...,
				[
				-570668.1600000001,
				-1270095.6799999997
				]
				]
				]
				}
				}
				]</pre>

	crs	The type of the coordinates reference system (CRS) that is used. This attributed is needed to correctly interpret geographical coordinates of geometries (see below).	String	
	geometry	A geometry object representing extracted location in a GeoJSON[28] format; e.g., POINT or POLYGON	JSON Object	
textual_concepts		A list of textual concepts inferred by topic modelling models.	JSON Object Array	<pre>["vineyard fire", "firefighters fighting", "nearby forest"]</pre>

textual_categories		A list of categories with probabilities assigned to the input text. Categories are from a predefined set:
		no_fire, urban_fire, wildfire, vehicle_fire, other
			JSON Object Array	<pre>[
				{
				"name": "wildfire",
				"score": "0.98"
				},
				{
				"name": "urban_fire",
				"score": "0.02"
				}
				]</pre>

Table 17: JSON attributes of the Facebook Post Analyser output

5.1.1.3 *Relevance estimation module*

With the increasing prevalence of social media usage, it has become a valuable source of information for real-time situational awareness, especially during natural disasters such as wildfires. However, manually sifting through an enormous amount of social media data can be time-consuming and inefficient. This is where CERTH's relevance estimation module comes in handy.

CERTH has developed a relevance estimation module that can quickly and accurately classify social media posts related to fire incidents. The module uses natural language processing and machine learning techniques to analyze the text of a social media post and determine its relevance to fire incidents. It can classify a post as either related to fire or not in a matter of seconds and provide a confidence score that indicates how certain the classification is.

Relevance estimation module can be used as a standalone web service, making it easy to integrate into existing systems. All that's required is a text of a social media post and the service will return a JSON string that shows if the post text is related with fire incidents and a score. An example of the returned JSON string is shown below:

```
ouput: {
        fire_related: true,
        score: 0.92
```

[28] https://geojson.org/

```
}
```


Table 18: An example of relevance estimation service output

5.1.2 Textual Concepts Extraction

5.1.2.1 *Multilanguage Dataset Processing for Concept Extraction from Social Media Posts*

Part of the data used for this task was made available by our partner CERTH. Other sources of data were found on dataset repositories such as Kaggle.

Italian

The Italian dataset provided by CERTH consists of a JSON file where a collection of tweets from recent years are stored. The tweets are available in a variety of language, such as Italian, English, Greek or Spanish.


```
{
    "id": "856179106916044801",
    "text": "fiamme a cembra https://t.co/Z5adQfBQnK via @YouTube",
    "timestamp": "2017-04-23T16:13:00Z",
    "relevant": true,
    "is_a_retweet_of": null,
    "quotes": null,
    "estimated_locations": []
},
{
    "id": "856181623674605568",
    "text": "Ancora fiamme sul passo San Jorio https://t.co/1FELFQlACY",
    "timestamp": "2017-04-23T16:23:00Z",
    "relevant": true,
    "is_a_retweet_of": null,
    "quotes": null,
    "estimated_locations": []
},
{
    "id": "856185323193348097",
    "text": "E' morto il bimbo lanciato da casa in fiamme https://t.co/o4CcBz4bYU",
    "timestamp": "2017-04-23T16:37:42Z",
    "relevant": true,
    "is_a_retweet_of": null,
    "quotes": null,
    "estimated_locations": []
},
```


Table 19: Sample of the Italian dataset from CERTH

The Italian part consists of about 600 hundred tweets. They have already been pre-filtered by CERTH for relevance in relation to fire events/accidents through manual annotation, although a method to automatically apply this filter might be needed for the production phase.

English

For the English language, no dataset was available so a search for public textual social media datasets regarding fire events was conducted. So far, a dataset for a Kaggle challenge about classifying whether tweets are about actual disasters or not was found[29]. The idea would be to filter out the tweets marked as non-disaster and from this subset further filter out non-fire disaster tweets.

The disaster tweets are 3200 out of 7500. Probably less than half of these will be about fire so further datasets might be needed.

Another idea for enriching the data sources would be to automatically translate the tweets between languages.

[29] Natural language processing with disaster tweets. Kaggle. (n.d.). Retrieved February 7, 2023, from https://www.kaggle.com/competitions/nlp-getting-started/data?select=train.csv

Some preliminary exploration has been done in this field. While most of the automatic translation APIs are paid, there are some FOSS alternatives. Of course, their quality is not on par with the paid alternatives but it might be sufficient for the task, so it could be worth to look into them.

5.1.2.1.1 Process

The goal for this sub-task is to further refine information by extracting concepts of interest from the text of social media posts.

The categories (or concepts) to be extracted have been identified based on the information found in the available tweets. The distribution of the categories on the training set is as follows:

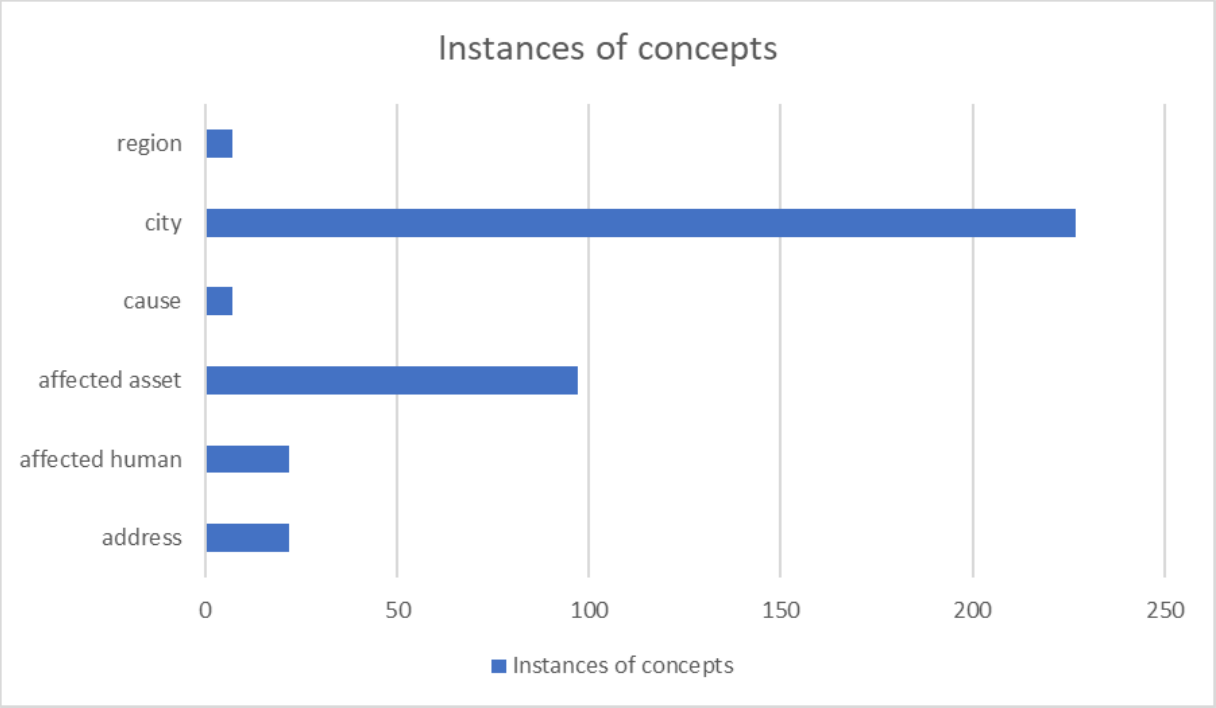


Figure 11: Distribution of concepts in the Italian dataset provided by CERTH

Italian

Regarding the Italian language, two different language approach were tested for extracting concepts related to fire incidents from tweets.

The first approach, Pure ML, uses a machine learning approach and was trained using EAI's proprietary platform that allows for a technique called active learning.

The second approach, Symbolic, takes a two-step method and uses EAI's internal knowledge graph to determine the relevance of concepts related to fire incidents.

Both models have their own advantages and disadvantages, with the Pure ML model being faster but having less control, and the Symbolic model providing more control but being slower.

- **Pure ML**

A machine learning language model was created for the extraction of *address, affected human, affected asset, cause, city,* and *region* concepts using 508 tweets in Italian as the training set and 123 tweets as the test set. The received tweets were already pre-categorized for relevance in relation to fire incidents.

The model was trained to extract the information contained in the tweets and put them into the specified fields, using a proprietary platform built by EAI. This platform provides a feature called "Active learning" which allows the user to train a machine learning algorithm by going through the following workflow:

1. Start tagging a small number of documents from scratch with the desired categories.
2. Train a model based on the documents tagged so far.
3. The model will try to tag the next small set of untagged documents.
4. Validate or correct the model tags on the next set of documents.
5. Retrain the model with the next set of tagged documents.
6. Repeat until satisfied with the performance.

The goal of the model is to accurately identify the different types of categories contained in the tweets, allowing for more efficient analysis and categorization of the data.

Pros

- Faster development

Cons

- Less control
- Needs significant data
- Filtering needed to discriminate if text is about fire event or not.
- **Symbolic**

A symbolic language model is being created for the extractions of the same concepts identified in the previous paragraph.

The mode is being built with two steps in mind:

1. Classification for fire-event relevance
2. Extraction of relevant concepts

The first step leverages EAI's internal knowledge graph to determine whether the concepts related to fire that appear in the text are actually about an accident or used in a metaphorical way ("This song is fire").

Once determined that the tweet is related to an accident, the model proceed to extract all the relevant concepts it can find.

Pros

- More control
- Needs less data
- Ability to filter for relevance in relation to fire events in the same model as the concept extraction.

Cons

- Slower development

English

Considering the experience with the Italian model, the English language model will probably be built following the symbolic approach. Even though it might be a slower approach, we don't know at this point how many English tweets will be available and it's also preferable to include the relevance filtering inside the model.

As described in the Data Collection paragraph, the gathering/searching for data is still ongoing. However, it is reasonable to expect that the structure of the symbolic rules developed for the Italian model could be reused in English.

5.1.2.1.2 Output

The output of the concept extraction will be a JSON file. Some of the fields it will contain are:

- Original text,
- Fields
- Field name (name of the concept)
- Positions in the text (indexes)
- Score
- Value (actual text)

The JSON may be further refined before being sent out in accordance with the partners requirements. The following is a sample of this JSON.


```
{
    "document": {
        "content": "Bari, bus Amtab va in fiamme in via Bruno Buozzi: ferito l'autista https://t.co/Z2OIgCo42b via @Borderline_24\n",
        "extractions": [
            {
                "fields": [
                    {
                        "name": "city",
                        "positions": [
                        "score": 1,
                        "value": "Bari"
                    }
                ],
                "template": "city"
            },
            {
                "fields": [
                    {
                        "name": "afftected_asset",
                        "positions": [
                        "score": 0.5149,
                        "value": "bus"
                    }
                ],
                "template": "afftected_asset"
            },
            {
                "fields": [
                    {
                        "name": "address",
                        "positions": [
                        "score": 0.9999,
                        "value": "via Bruno Buozzi"
                    }
                ],
                "template": "address"
            },
            {
                "fields": [
                    {
                        "name": "affected_human",
                        "positions": [
                            {
                                "end": 66,
                                "start": 59
                            }
                        ],
                        "score": 0.9999,
                        "value": "autista"
                    }
                ],
                "template": "affected_human"
            }
        ]
    }
}
```


Table 20: JSON output sample

5.1.2.2 *Concept extraction in NLP using BERTopic and KeyBERT*

For concept extraction or concept mining, we chose keyword mining as an approach more at home in Natural Language Processing (NLP). Using a sample of 1 K tweets from the INGENIOUS dataset in English (950 K, from CERTH), two methods, BERTopic[30] (Grootendorst, 2022) vs. keyBERT[31] (Grootendorst, 2020) were tested. BERTopic is a topic modelling technique that leverages transformers and c-TF-IDF to create dense clusters allowing for easily interpretable topics whilst keeping important words in the topic descriptions. It supports guided, supervised, semi-supervised, manual, long-document, hierarchical, class-

[30] https://maartengr.github.io/BERTopic/index.html
[31] https://maartengr.github.io/KeyBERT/

based, dynamic, and online topic modelling, including visualisations. On the other hand, KeyBERT is a minimal and easy-to-use keyword extraction technique that leverages BERT embeddings to create keywords and keyphrases that are most similar to a document.

BERTopic needed a larger sample of tweets, whereas the idea behind KeyBERT is to mine concepts aka topics from single texts. As Figure 12 shows, focusing on wildfires, in the case of BERTopic the labels assigned to document groups can be considered as concepts, whereas KeyBERT's results are closer to hypernym identification by extracting related concepts from running text.

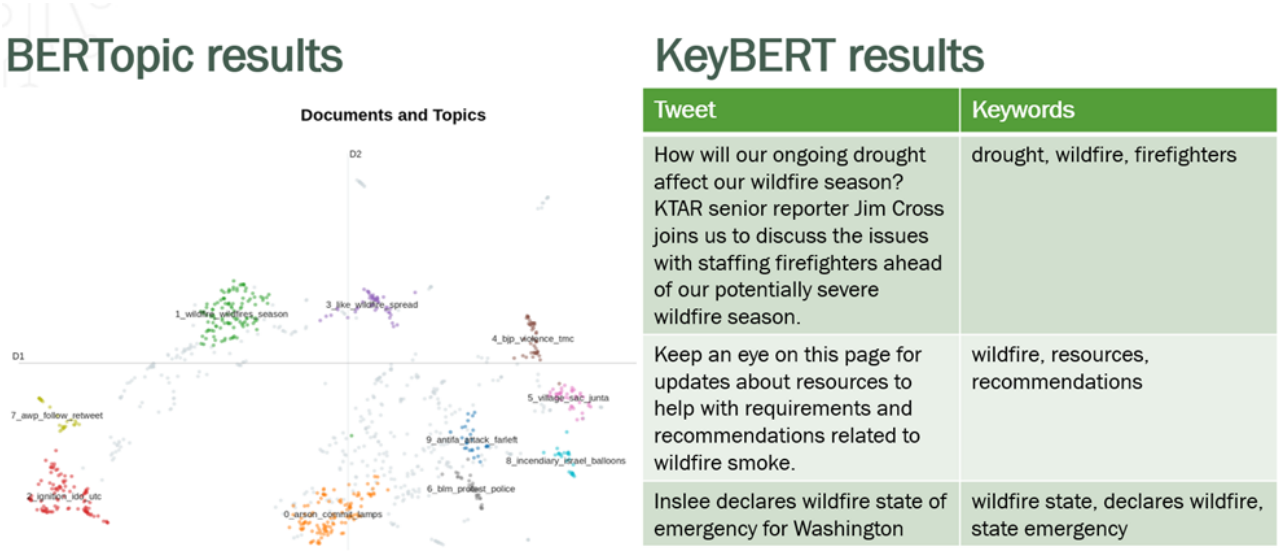


Figure 12: Extracted textual concepts on document collection vs. individual document level

5.1.3 Event Recognition

5.1.3.1 *Data collection*

The data for this task will be the same used for concept extraction.

5.1.3.2 *Process*

The goal for this sub-task is to further refine information by identifying events from the text of social media posts.

The work is being setup as a mix of machine learning and symbolic rules approach, which should allow for a speed increase in the development.

The machine learning algorithms offered by the EAI Platform allow for the creation of basic symbolic rules to bootstrap their development. This is done by semi-automatically annotating the texts with the identified events until a sufficient number of annotations is available for the inference of symbolic rules.

The annotation process is described as semi-automatic because as the number of annotations grow, so does the number of suggested annotations, which are inferred by the engine and speed up the process.

After this first annotation phase, a manual refining of the inferred rules will follow. This phase represents the bulk of the task and will involve development of symbolic rules for each event in the taxonomy.

5.1.3.3 *Event Taxonomy*

A small taxonomy was outlined for the categorization of events mentioned in tweets. This taxonomy represents a hierarchical structure of domains related to fire incidents.

At the time of writing most of the domains reflect the classes included in the SILVANUS ontology developed in T3.1. However, it is expected that additional domains will emerge from the exploration of the dataset as well (for example the presence of smoke or the topic of air quality).

The taxonomy includes domains related to causes of fires, such as "cause" which further branches out into subdomains such as "accident," "deliberate," "natural," "negligence," and "unknown." It also includes domains related to fire incident characteristics such as "area_burned," "climate_parameter," and "vegetated_area."

Lastly, it includes domains related to vulnerable objects that may be affected by a fire, such as "infrastructure," "property," "structure," and "living_being," which further branches out into subdomains such as "human" and "animal."

Overall, this taxonomy serves as a useful organizational framework for categorizing and understanding different aspects of fires and their impact.

The taxonomy at the time of writing is the following:

- smoke
- cause
    - accident
    - deliberate
    - natural
    - negligence
    - unknown
- area_burned
- climate_parameter
    - precipitation
    - season
    - sky_cover
    - temperature
    - wind_condition
- incident
- vegetated_area
- vulnerable_object
    - asset
        - infrastructure
        - property
        - structure
    - living_being
        - human
        - animal

As mentioned, it is highly likely to be subject to modifications.

5.1.3.4 *Output*

The output of the concept extraction will be a JSON file. Some of the fields it will contain are:

- document (Original text),
- categorization (list of events found)
    - name (internal name of the event)
    - label (user friendly name of the event)
    - Score

The JSON may be further refined before being sent out in accordance with the partners requirements. The following is a sample of this JSON.

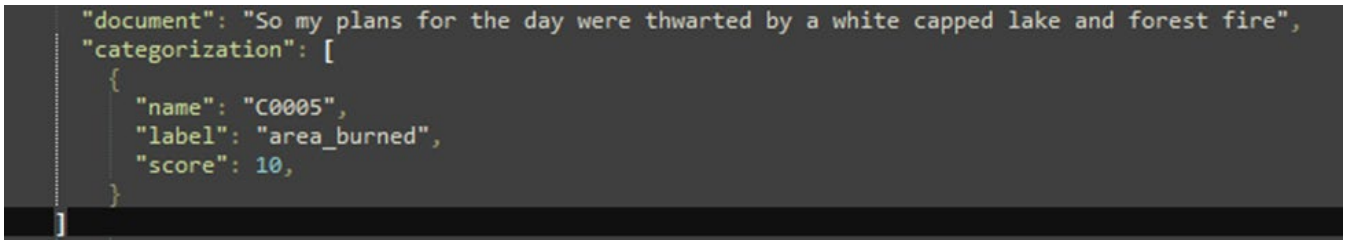


Figure 13: JSON output sample

5.1.4 Locations Extraction

5.1.4.1 *Localization module*

Social media platforms often fall short when it comes to providing essential geolocation information for posts. This information has various of uses such as link satellite data with social media posts or pin point fire event in a map. Unfortunately, the data collected from the Twitter API is lacking in terms of geolocation, with only a small percentage of tweets providing this information.

To tackle this issue, CERTH has implemented a localization module. This module is designed to search for locations mentioned within the text of social media posts and link them to precise coordinates in the World Geodetic System (WGS 84). The module was initially developed as part of the European project EOPEN and has been further refined and improved by CERTH.

The localization module operates by taking in Twitter text, pre-processing it, and forwarding it to a Long Short-Term Memory (LSTM) network. For each word or phrase that is identified as a location, a Named Entity Recognition label is assigned. These labels are then used to form queries to the OpenStreetMap API, which provides the geolocation data and fetches the exact coordinates for each location.

This module is available as a standalone web service, which can be easily integrated into the systems. It accepts as input the text of a social media post and returns a JSON Object Array with the complete name of the place, the type of the coordinates reference system that is used, and its precise coordinates. This information is then added as an attribute to the JSON of the social media posts collected by the crawler, providing as many as possible retrieved social media posts with geolocation data. An example of the JSON structure is provided for reference in Table 21.

```
"extracted_locations": [
    {
     "placename": "Jurien Bay, Shire Of Dandaragan, Western Australia, 6516, Australia",
      "crs": "WGS84"
      "geometry": {
        "type": "Point",
        "coordinates": [{
          "lat":115.0406027,
          "lon":-30.3040478
        }]
      }
    }
]
```

Table 21: CERTH's location extraction JSON output

5.1.4.2 Text classification and location extraction

To get the locations of forest fires, an API is created using Name Entity Recognition of social media data that has been identified as text related to forest fires.

For location data in Indonesian, Twitter data is used. There are 2 main steps in this process, namely text data classification and identifying the location of the classified text.

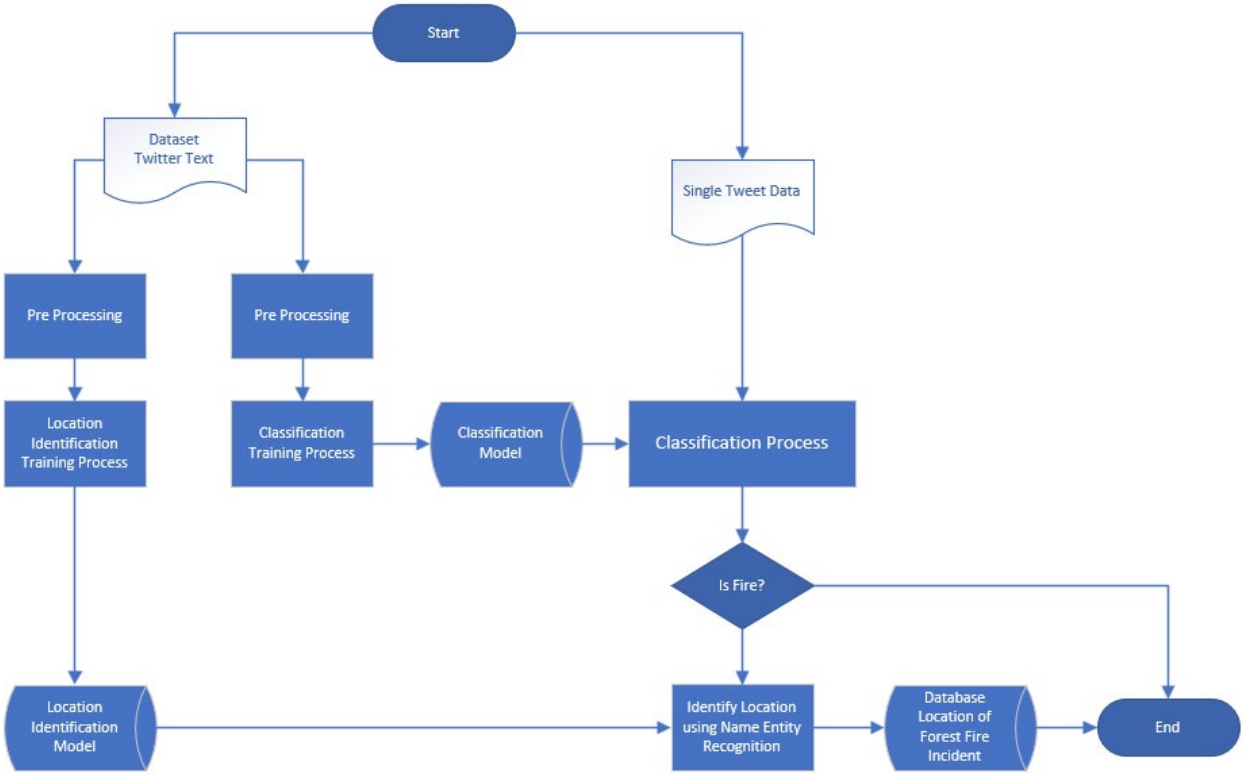


Figure 14: Proposed framework

5.1.4.2.1 Text data classification

In this project, we collected data from twitter related to forest-fire in Indonesia from 22 June 2022 to 28 July 2022. The number of data is 1,047. Then, manual labelling is conducted to classify 6 classes as shown in the Table 22.

No	Bahasa Indonesia	English
1	Kebakaran	Forest-fire
2	Pencegahan	Prevention
3	Rehabilitasi	Rehabilitation
4	Mitigasi	Mitigation
5	Penanggulangan	Countermeasures
6	Tidak diketahui	Unknow

Table 22: Label class twitter dataset

Commonly, developing text classification in Machine learning or Deep learning needs some pre-processing techniques, such as case folding, tokenizing, filtering, and stemming. We do pre-process techniques using

regular expression and Sastrawi library for stemming and remove stop word in Bahasa Indonesia. The dataset preview can be seen in the Figure 15.

	id	screen_name	text	clean_text	dt	label_id	is_predict	label_name	label	Desc
0	23	kumparan	Untuk mencegah terjadinya kebakaran hutan dan ...	cegah jadi bakar hutan lahan sumatera selatan ...	2022-06-22 7:03:12	1	1	pencegahan	kebakaran	NaN
1	42	updatebalii	Kebakaran hutan dan lahan (karhutla) hingga ki...	bakar hutan lahan karhutla hingga kini intai m...	2022-06-22 10:57:45	1	1	pencegahan	kebakaran	NaN
2	26	beritaindo	Sumsel siapkan ribuan personel mitigasi karhut...	sumsel siap ribu personel mitigasi karhutla	2022-06-22 7:26:23	2	1	kebakaran	mitigasi	NaN
3	1	kompascom	Petugas berpatroli ke kawasan hutan dan lahan ...	tugas patroli kawasan hutan lahan cegah jadi k...	2022-06-22 0:55:44	1	1	pencegahan	pencegahan	NaN
4	5	GATRA_com	Masalah Karhutla di Muba Makin Urgen, Pemkab K...	masalah karhutla muba makin urgen pemkab kebut...	2022-06-22 1:32:02	1	1	pencegahan	pencegahan	NaN

Figure 15: Dataset preview

According to the data crawled now, there are 136 data labelled for forest fire, 458 data for prevention, 54 data for rehabilitation, 127 data for mitigation, 164 data for countermeasures, and 90 remainder of the data are labelled as unknown.

It is seen that class distribution is not balanced. To deal with an imbalanced dataset situation, we need a technique to balance the data. We implement Random Oversampling (ROS) to make the dataset balanced. Random Oversampling includes selecting random examples from the minority class with replacement and supplementing the training data. Imbalanced-learn library is used in this experiment and shows in Figure 16.

```
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(random_state=42)
X, y = ros.fit_resample(X, y)

✓ 0.4s
```

Figure 16: Random Oversampling code

Finally, after Random Oversampling phase, the number in each class become balance. All of the class labels consist of 458 data. Table 23 describes the composition of class label data before and after balancing step.

No	Class label	Before balancing	After balancing
1	Forest-fire	136	458
2	Prevention	458	458
3	Rehabilitation	54	458
4	Mitigation	127	458
5	Countermeasures	164	458
6	Unknow	90	458

Table 23: Number of label class before and after balancing

The next step is split dataset into subgroups (training and testing). The training dataset is the initial subset, which is used to fit the model. On the other hand, testing dataset is used for evaluating the trained model. We split 80% for training data and 20% for testing data.

We used a Deep Learning technique called Long Short-Term Memory to train the model (LSTM). LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. We also employ fastText word embedding to convert text to vector numbers. It is

believed that fastText could improve text classification widely used for NLP. Finally, a word vector with a size of 784,200 300 was employed based on the preceding embedding matrix procedure.

The result of the experiment using Bidirectional LSTM (2 layers) and fastText word embedding are shown in Figure 17. Training was carried out in 50 epochs to build the model.

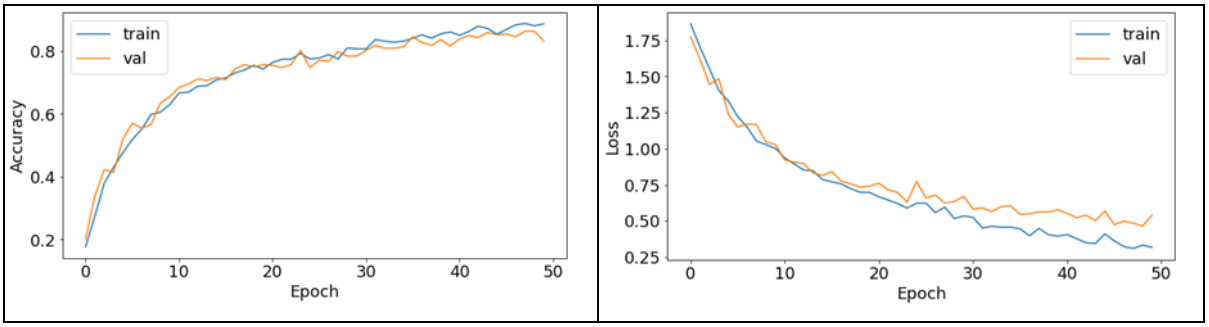


Figure 17: Training accuracy and training loss of the LSTM model

As can be seen, the training and validation results show a smaller distance. The validation accuracy cross the training accuracy at earlier epoch of training. It is indicated that the model reached convergency earlier. Training accuracy achieved 93.68% and the testing accuracy is 84%. The report detail shows in the Table 24.

No	Class label	Precision	Recall	F1-Score
1	Forest-fire	0.83	0.95	0.89
2	Prevention	0.90	0.60	0.72
3	Rehabilitation	0.92	1.00	0.96
4	Mitigation	0.73	0.92	0.81
5	Countermeasures	0.73	0.68	0.71
6	Unknow	0.98	0.88	0.93

Table 24: Classification report for testing the model

5.1.4.2.2 API for text classification

Training produces a model, which is compiled in the H5 file format. The API for text class prediction is developed based on the H5 file format. We developed the API using Flask API Framework. The API platform has been implemented at AWS server with url: http://34.229.139.48:5000/isfire

The API has been changed to categorize into two classes (true and false). The forest-fire label was set to "true" in the first class, while other labels were set to "false." The API for text classification could be accessed using code below:

```
$ curl --location --request POST 'http://34.229.139.48:5000/isfire'
--header 'Content-Type: application/json'
--data-raw '{
"captions":
[
"Telah terjadi kebakaran di daerah gambut Sampit",
"Sebaiknya warga tidak bermain api unggun"
]
}'
```

The output from this request:

```
{
"res": [{
```

```
        "en_sentence": "There has been a fire in the Sampit peat area",
        "fire_status": true,
        "sentence": "Telah terjadi kebakaran di daerah gambut Sampit"
        },
        {
        "en_sentence": "people should not play bonfire",
        "fire_status": false,
        "sentence": "sebaiknya warga tidak bermain api unggun"
        }],
        "status": "success"
}
```


5.1.4.2.3 Identifying location

Named-entity recognition (NER) is one of the methods to get location data from articles. In this project, we focused on identify location and time from dataset. We used Bidirectional LSTM-CRF method to find the location and time in the dataset. The dataset is collected from google search using keyword “kebakaran hutan di Indonesia” (Forest-fire in Indonesia). Before labelling the data with a BIO-Tag, we must perform a pre-processing stage. The dataset contains 1,521 sentences with the total number of words is 27,736. The preview of dataset with BIO-Tag as seen in the Figure 18.

	Sentence	Word	Tag
4697	267	Wilayah\n\nAceh	O
4698	267	Sumatera	B-geo
4699	267	Utara	I-geo
4700	267	Sumatera	B-geo
4701	267	Barat	I-geo
...	...	...	...
4756	267	Papua	O
4757	267	ASEAN	O
4758	267	Indonesia	B-geo
4759	267	Sumatera	B-geo
4760	267	Kalimantan\n\nIndex	O

64 rows × 3 columns

Figure 18: Preview dataset with BIO-Tag

As can be noted, there are more tags with a value of 0 than any other tags. The number of words with 0 tag is 26,051, the location tag with B-geo is 703, and the I-geo tag is 372. There are 341 and 270 words with time information utilising B-tim and I-tim tags, respectively.

The next step is converting string text to the vector number with maximum length of sentences is 30. Then, we split data to training set (80%) and testing set (20%). Bidirectional LSTM and CRF perform to train the classification model.

5.1.4.3 *Location extraction in Slovak Facebook posts*

Location extraction from Slovak Facebook posts is based on gazetteers, external geolocation services and deep learning. Gazetteers are built from the Database of Standardized Geographical Names from the territory of the Slovak Republic, updated on 07th July 2020. Geographical names feature classes: Historical name, Variant name, Cadastral territory, Part of the municipality, Municipality, District and

Region.[32] Geographical names are linked to a territorial and administrative boundaries[33] in coordinate reference system S-JTSK[JTSK] (code EPSG:5514). There is a StringGazetteer[34] implementation used from the python-gatenlp library. The deep learning models are based on SlovakBERT[35] model and the NER (Named Entity Recognition) task is learned on top of the wikiann dataset[36]. This dataset contains 50907 Slovak sentences with NERs as: location, organization and names. We split it as follows: training set: 30907, validation: 10000 and test: 10000 sentences. The dataset structure is the same as described in the Figure 19. The model achieved results are described in following Table 25 (please note, that these results are achieved over test dataset, which was not seen by the model during training). Please note that additionally to this model, we had trained also the sentence tokenizer model (it achieves almost 100%, so we do not share the results here).

	Overall	Location	Person	Organization
Precision	0.94174	0.94702	0.95880	0.91352
Recall	0.95125	0.95900	0.97694	0.90989
F1	0.94647	0.95297	0.96778	0.91170
Accuracy	0.98122	-	-	-

Table 25: Classification report for testing the model

The model was trained with following hyperparameters:

- learning_rate: 5e-05
- train_batch_size: 64
- eval_batch_size: 8
- seed: 42
- optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- lr_scheduler_type: polynomialLR (1st degree)
- num_epochs: 20

Training results are described in the following Table 26.

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.21312	1	483	0.12892	0.88894	0.90919	0.89895	0.96475
0.09079	2	966	0.10156	0.90137	0.93715	0.91891	0.97278
0.06219	3	1449	0.08746	0.93113	0.93813	0.93462	0.97802
0.04543	4	1932	0.09262	0.92974	0.94204	0.93585	0.97773
0.03119	5	2415	0.09506	0.92210	0.94151	0.93171	0.97697
0.02652	6	2898	0.10014	0.93141	0.94121	0.93628	0.97821
0.02037	7	3381	0.10650	0.93150	0.94159	0.93652	0.97844
0.01527	8	3864	0.10609	0.93828	0.94399	0.94113	0.97967

[32] https://www.geoportal.sk/files/zbgis/na_stiahnutie/shp/gn_shp.zip
[33] https://www.geoportal.sk/files/zbgis/na_stiahnutie/shp/ah_shp_3.zip
[34] https://gatenlp.github.io/python-gatenlp/gazetteers.html
[35] https://arxiv.org/abs/2109.15254
[36] https://huggingface.co/datasets/wikiann

0.01446	9	4347	0.10237	0.93517	0.94775	0.94142	0.97934
0.01073	10	4830	0.10733	0.94106	0.94948	0.94525	0.98024
0.00775	11	5313	0.10419	0.94105	0.95159	0.94629	0.98132
0.00682	12	5796	0.11002	0.93615	0.94903	0.94255	0.98035
0.00489	13	6279	0.11416	0.94419	0.95008	0.94713	0.98125
0.00510	14	6762	0.11141	0.94195	0.94903	0.94548	0.98086
0.00449	15	7245	0.11206	0.94128	0.95196	0.94659	0.98139
0.00302	16	7629	0.11609	0.94229	0.95249	0.94736	0.98136
0.00229	17	8013	0.12310	0.94299	0.95369	0.94831	0.98146
0.00194	18	8397	0.12385	0.94292	0.95256	0.94772	0.98164
0.00171	19	8781	0.12518	0.94220	0.95339	0.94776	0.98172
0.00139	20	9165	0.12430	0.94325	0.95347	0.94833	0.98182

Table 26: Model training report

Access to additional external geocoding services is brought by GeoPy[37] library, which provides implementations for many different services; e.g., Goggle Maps Platform, OpenStreetMap, Nominatim, Bing Maps.

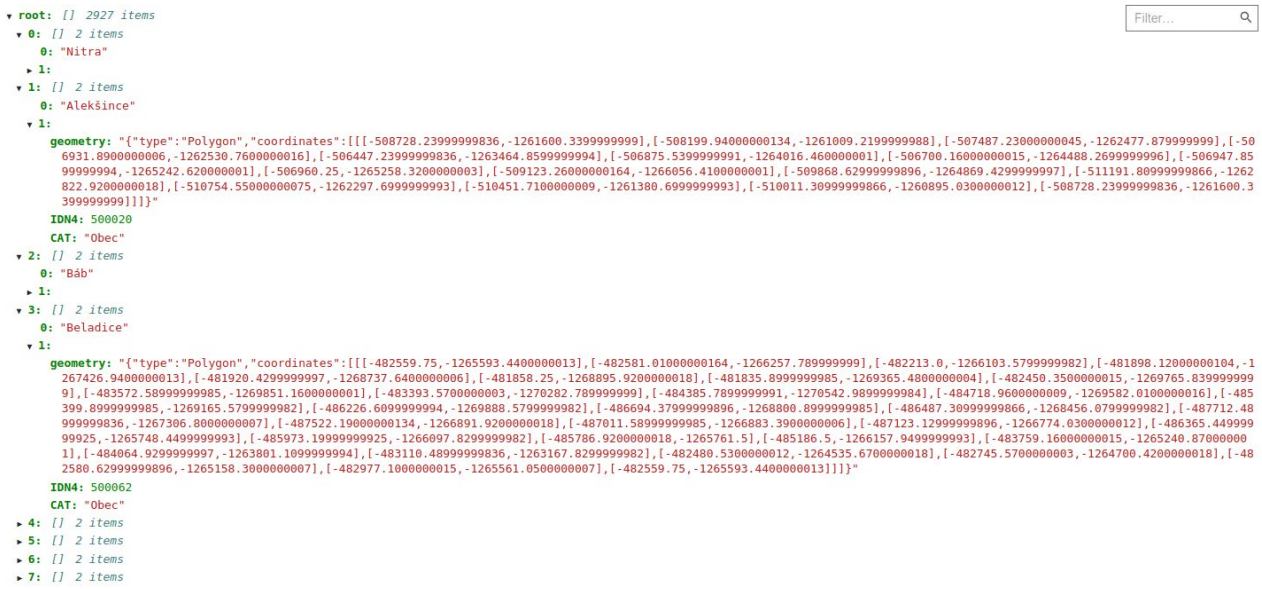


Figure 19: An example of a gazetteer definition containing Slovak geographical names with geometry attributes and category (obec - municipality)

5.2 Visual Analysis

This subsection provides a summary of the visual analysis provided by the Social Media Analysis Toolkit that processes visual information gathered by the Social Media Crawlers. The methods employed by these components are:

I. Fire and Smoke Detection in Images: This method utilizes computer vision techniques to identify flames or smoke in images (see section 5.2.1).

[37] https://geopy.readthedocs.io/en/stable/#module-geopy.geocoders

II. Visual Concept Extraction: Uses object recognition algorithms to extract meaningful visual concepts and objects from images, such as buildings, vehicles, and people. The extracted information is stored in the metadata of the post (see section 5.2.2).

III. Location Extraction: Extracts location information from images of a social media post (see section 5.2.3).

5.2.1 Fire and Smoke Detection in Images

The image fire (and smoke) detection task, consists of series of interconnected subtasks, such as the data collection and pre-processing (filtering), and finally, training of the neural network. Each of these subtasks will be explained in detail in the following subsections.

5.2.1.1 *Dataset Creation*

For the training of Machine Learning (ML) algorithms, and especially deep Neural Networks (NNs), a large number of data is needed to achieve the desired performance. Therefore, images from various sources were collected to build a sufficient in size and diversity dataset. By diversity, we are referring to the content of the images, i.e., the depicted scenes contained fires at various stages (e.g., smoke, flames, etc.), from different angles/viewpoints (e.g., aerial and ground images), in different environments (e.g., fields, houses, forests, etc.) and different resolutions to ensure a robust algorithm.

The image sources consisted of fire/smoke benchmark datasets (FLAME (Alireza et al., 2020), Corsican Fire Database (Toulouse et al, 2017), FiSmo (Cazzolato et al., 2017), EFDNet dataset (Li et al., 2020) and BoWFire (Chino et al., 2015)), online image repositories such us Kaggle[38], websites (https://www.flickr.com/, https://unsplash.com/ and https://www.forestryimages.org/index.cfm), images uploaded in social media (e.g., Twitter) and finally, images captured from SILVANUS pilot sites (e.g., Slovakia). Images gathered from social media were collected using the Social Media Crawlers described in section 4.1. Note, that the image sets provided from each source were considered only as candidates for the dataset, as they needed to pass through the pre-processing pipeline described in the next subsection. Therefore, only a subset of the collected data made it to the final version of the dataset.

Once the filtering of the images was completed, the remaining images were manually annotated to the following categories:

- **Fire:** images containing only fire.
- **Smoke:** images containing only smoke.
- **Both:** images containing both fire and smoke and finally.
- **None:** images that do not contain either fire or smoke.

These annotations, were used in the final step of this task – the neural network training for the fire and smoke detection in images (explained in section 5.2.1.4 AI model training). Note that the category 'Both' is needed to test the fusion of the fire and smoke model results in these more specific scenarios and test their behaviour in more challenging images (e.g., fire covered by smoke or more red/yellow-ish smoke because of the fire).

[38] Datasets collected from Kaggle are the following:

- https://www.kaggle.com/datasets/brsdincer/wildfire-detection-image-data
- https://www.kaggle.com/datasets/ritupande/fire-detection-from-cctv
- https://www.kaggle.com/datasets/atulyakumar98/test-dataset
- https://www.kaggle.com/datasets/phylake1337/fire-dataset/code
- https://www.kaggle.com/datasets/dataclusterlabs/fire-and-smoke-dataset

A summary of the total number of images collected per category (e.g., fire), and after the filtering stages, can be seen in Table 27. The refined dataset, from now on will be referred to as Silvanus Image Dataset (SID) for brevity.

Category	#images
Fire	1410
Smoke	2759
Both	5206
None	11361
Total	20736

Table 27: SID dataset images' breakdown per classification category (i.e., fire, smoke, both, none)

5.2.1.2 *Data pre-processing*

The collected images were examined to determine their suitability to the task in hand with the aim of creating the SID dataset, that would later be used for the detection of fire/smoke. Through this study, some unrelated image categories were identified and for this reason, a set of filtering steps were defined to rid the image set (and any new incoming data) of them.

After studying the images, we noticed that even in the cases (e.g., social media, websites) where a keyword/tag (e.g., fire) was used to retrieve images the keywords/tags were very broad terms with several meanings, leading to the assembly of various images unrelated to the task in hand. Also, because of the social media platforms' nature and the website tagging, there were multiple cases of duplicate images, for example through reposts or an image with multiple tags (e.g., wildfire and fire). Additionally, an image could appear in multiple sources (e.g., in an online repository and social media), making the need for a deduplication step more prominent. For these reasons, the raw data could not be directly used for the training of the NNs, but rather be passed through several filtering stages to ignore as many as possible of these unwanted images.

It is worth mentioning, that the proposed image refinement pipeline will be used throughout the project's lifespan to ensure its better performance and avoid wasting resources. The only step that will not be included in the pipeline deployed in the SILVANUS cloud is the image deduplication. The main reason is, that it is not feasible to compare incoming images to all previously gathered ones (because of their exponential volume growth) and the system will spend precious time doing so, losing some of its prominent features (e.g., early fire event detection). Though, the exclusion of the deduplication step is not harmful to the system's effectiveness (during inference they are insignificant), but it is crucial for the training of the NNs as duplicates make them more prone to overfitting.

Namely, the following image categories needed to be filtered out:

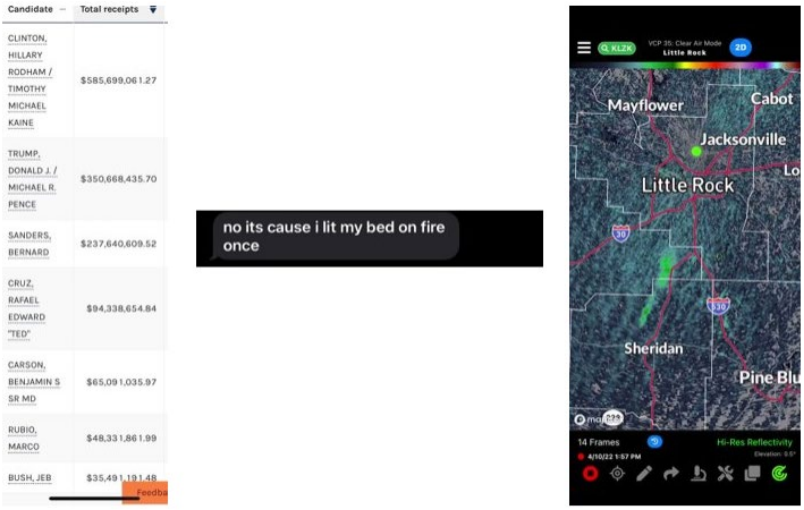
1. Image **duplicates**
2. **Greyscale** images
3. **Inappropriate** content (e.g., nudity/pornographic content)
4. **Unrealistic** images (e.g., drawings/posters/animation)
5. **Altered** images (e.g., cropped images, screenshots)
6. **Fuzzy** images (e.g., blurry, with little spatial info)
7. **Small** images

Example images from these categories, in the same order, can be seen in Figure 20(a-f), except from category 3 (inappropriate content) for apparent reasons.

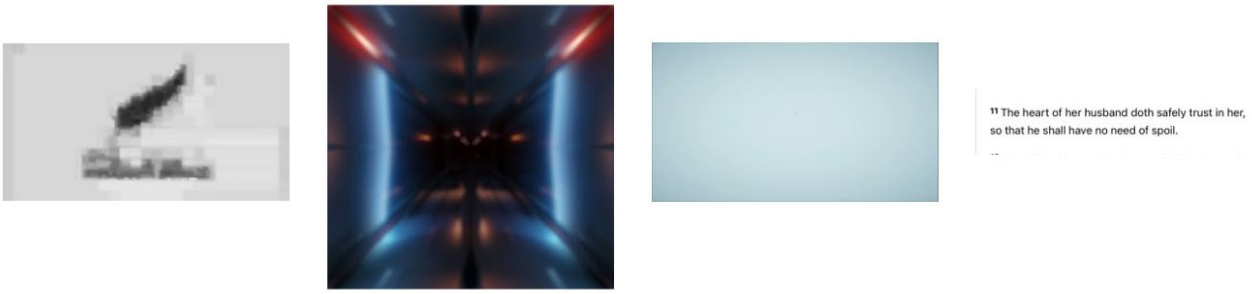
a) Duplicates: original image (left), exact duplicate (middle) and near duplicate (right).

b) Greyscale image examples.

c) Unreal images: animation (left), posters (middle) and drawing (right).



d) Altered images: cropped (left and middle) and screenshot (right).



e) Fuzzy images: blurry (left) and little spatial info (right).

f) Small sized images (shown in real size).

Figure 20: Image examples of unwanted image categories identified in the collected data.

Images belonging to the aforementioned categories needed to be removed from collected images, as they are not fit for the training of a NN or do not depict a real situation. In other words, exact and near duplicates (e.g., slight horizontal/vertical shiftings/translocations) will interfere with the NN training (making it more prone to overfitting), very small images (i.e., less than 10K pixels) will be very distorted when scaled to match the NNs input size (average NN input is 224x224≈50K pixels). The same applies to very blurry images, that are hard to be labelled by humans, nevertheless be an algorithm, especially after the noise addition during the image pre-processing stage (which will make them even more noisy). Lastly, greyscale images can be considered as an outlier because more often than not, the collected images will be coloured. Furthermore, usually, greyscale images correspond to old fire events (before the breakthrough of RGB cameras) or modified images (transformed from RGB to greyscale), which again can be considered as unwanted categories.

The remaining image categories (inappropriate content, greyscale, unrealistic/altered images and images with little spatial information) can be included in the dataset but we know beforehand they are not related to the task. Therefore, we thought it would be best to filter out as many as possible of these types of images beforehand and allow the network to spend its resources in learning features from more useful images. Furthermore, in some cases, a fire might be present, but it does not depict a real case scenario (Figure 21), if such an image *"escapes"* the filters it will be kept in the dataset and annotated as a fire image (the same logic applies to the remaining classification categories). That is, because firstly we aim to recognise if a fire is present in the image and in a later stage (in combination with the concept extraction) distinguish fake from real events (to avoid any false alarms). As a side note, we are well aware that altered images can portray real events of fire, but in an emergency situation someone would post the raw image, so the same tactic will be applied here as in the case of unreal images.

Figure 21: Image examples that contain fire but do not depict real situations (emergencies)

5.2.1.3 *Image filtering pipeline*

For the identification of images belonging to the aforementioned categories, several methods were applied, including image statistics and Machine Learning (ML) methodologies, an overview of the pipeline can be seen in Figure 22.

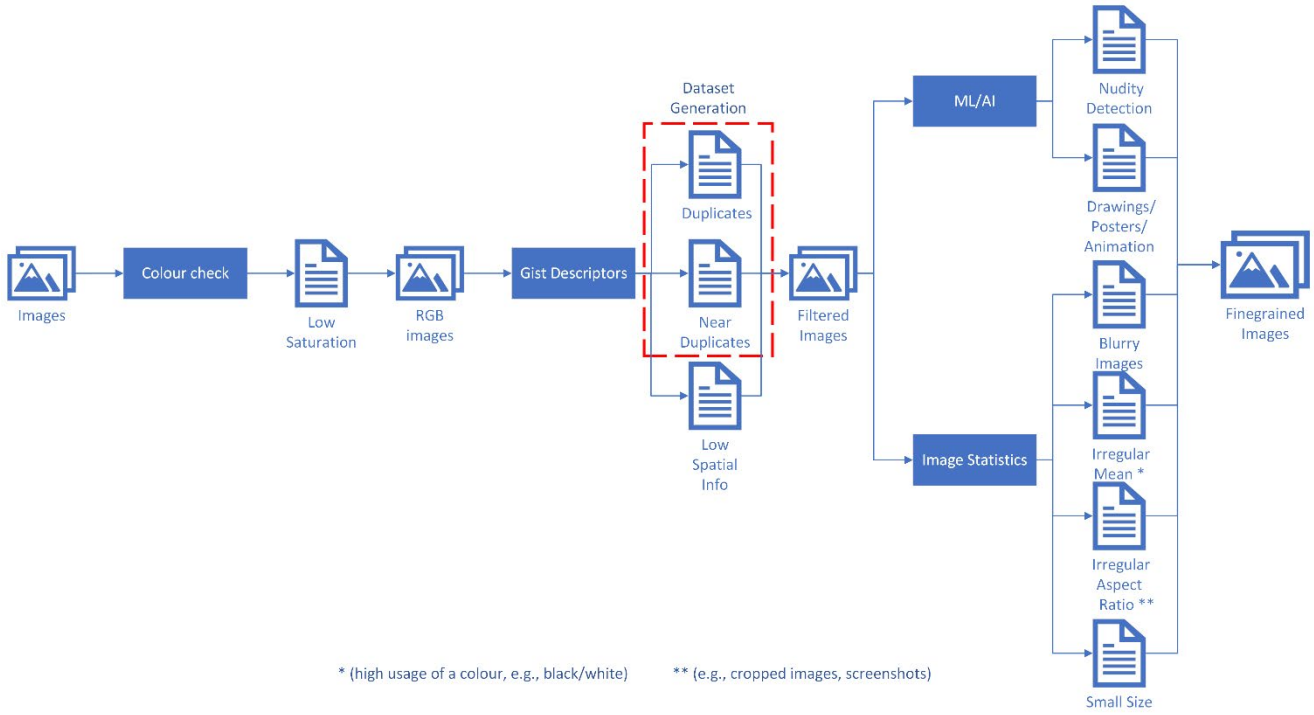


Figure 22: Visualisation of the proposed image filtering pipeline. The steps enclosed in the red-dashed rectangle will be used only during the dataset generation

The first category of images to be removed, are the **greyscale images**. Their identification was simply done by detect which images have low saturation (*"A grayscale or black-and-white photo has no colour saturation"*[39]), when converting RGB channels to HSV.

Next, was the identification of **duplicates** and ***"empty"*** images (i.e., images that depict few to none objects). To do that we employed gist descriptors (Oliva et al., 2001), an image representation that captures the dominant spatial structure of the depicted scene –gist of the image- to determine duplicates or empty scenes.

Specifically, duplicate images will have similar gist descriptors (Figure 23), therefore, they should be close in the vector space when comparing them. Specifically, we measure the closeness of the descriptors with the chi-squared (X^2) distance[40], a metric frequently used in similar tasks. Note, that X^2 distance is close to 1 when identical images are compared and near zero otherwise.

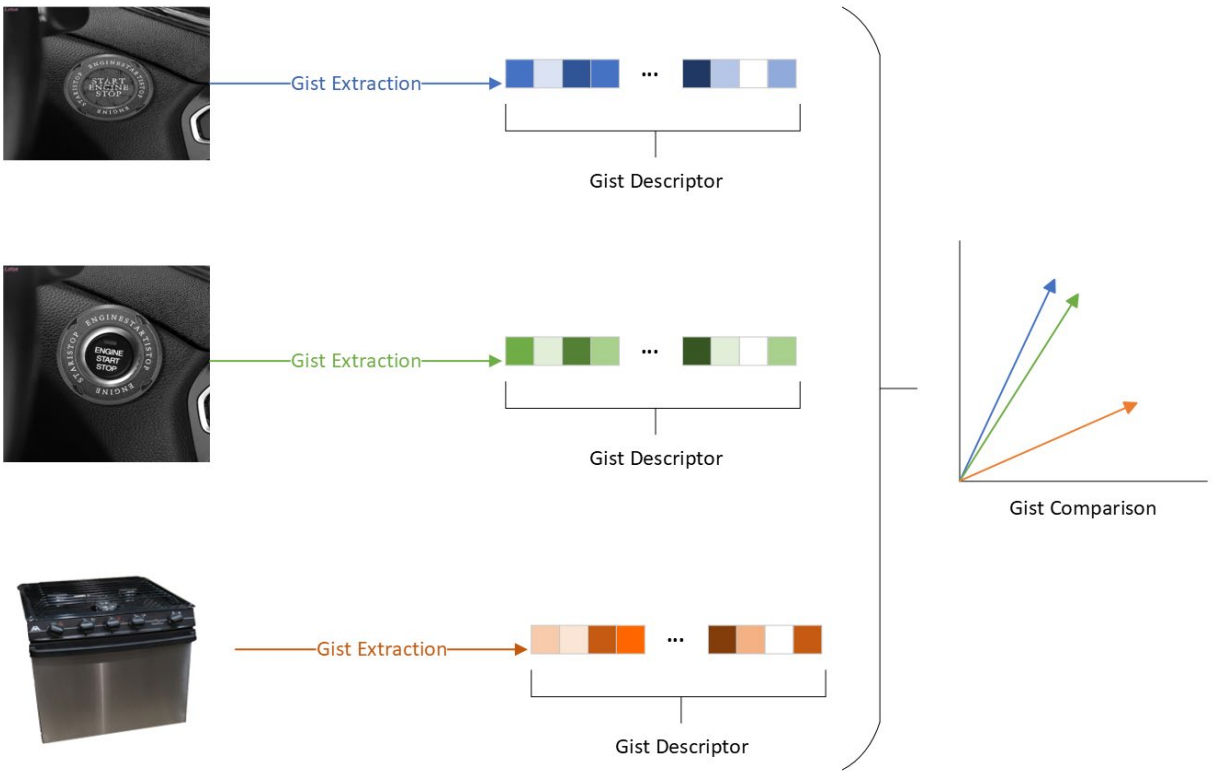


Figure 23: Image duplicate identification with Gist descriptors

The difference between **near and exact duplicates** is the percentage of the compared images' similarity, because of the slight perturbations in the depicted scene a smaller amount of gist features will be "closer". In other words, images whose X^2 distance (similarity) is ≥ 0.6 are considered duplicates and for distances between 0.4 and 0.6 near duplicates. Initially, these thresholds might seem low, but we wanted to ensure that all types of duplicates were filtered out, as they have a significant impact on the NNs' training. Considering that, during the training of NNs, images undergo an augmentation process (produce alterations

[39] Adobe, 2023," What is saturation? "Section. https://www.adobe.com/cy_en/creativecloud/photography/discover/photo-saturation.html#:~:text=Saturation%20describes%20the%20intensity%20of,wildflowers%20might%20be%20extremely%20saturated.

[40] $X^2 = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2 / (x_i + y_i)$

of the input for robustness), it is crucial to remove near duplicates because they can be produced during that phase (e.g., horizontal/vertical shifts, cropping, etc.).

After the completion of the **deduplication** phase, the remaining gist descriptors are compared with a zero vector, i.e., the vector corresponding to an image with no spatial information. Images that had more than 50% similarity with the zero vector were marked as empty. Through this step, we identified *"blank"* images, not images whose files were corrupted, but rather images that did not depict any particular object (i.e., the majority of the image is covered by a single colour).

The spatial information check completes the image filtering stage using the gist descriptors, leading to the next steps of the image filtering, those based on ML and those on image statistics (which can run in parallel to speed up the process).

As previously mentioned, in the collected data many of the images are unrelated to the task. Such image subcategories were **drawings** and **animation**, among others, that might illustrate fires but **not real** ones. Additionally, a number of images with inappropriate content were noticed that we wanted to remove from the dataset as well. Since these subcategories belong to a specific type of images and follow some kind of pattern, the fastest way to identify them is with NNs that were specifically trained for this task. For the recognition of images with inappropriate content we used the OpenNSFW2 (Yung, 2021) and the NSFW-Detector (Gant, 2020) models, with the latter being also used for the identification of unrealistic images (e.g., animation)

For the **ML filtering**, all images that passed through the first filtering phase were passed to the pre-trained networks to get the probability of them belonging to one of the undesired categories. Specifically, OpenNSFW2 outputs a probability for the image to be of pornographic content, if that probability was ≥ 0.7 (the network was confident enough for its decision), the image was added to the list of images to be discarded (because of its content). On the other hand, the NSFW-Detector is a multi-classification model, trained to recognise drawings and pornographic images (among others), therefore it was used to filter different type of images. For the drawings category, a threshold of ≥ 0.7 was used in order to collect a large amount of human made images (e.g., posters, animation, etc.), but for the case of pornographic images a higher threshold was required. Concretely, a threshold of ≥ 0.85 was used compared to the 0.7 for OpenNSFW2, because we noticed that the model was a bit biased towards any kind of exposed skin (non-pornographic), groups of people and with clothing in a similar colour with the skin tone. A cause of this could be the quantity and variety of images the network has seen during training. For these reasons, we increased the threshold (for an image to be considered inappropriate) to reduce the number of false positives.

It must be noted, that NSFW-Detector could only be used for the identification of unreal images, since OpenNSFW2 was utilised for the recognition of inappropriate images. Considering though, that the first simultaneously provides the probabilities for the different classes, it was decided to keep that information and filter out any inappropriate images the latter might have missed. Respectively, NSFW-Detector could not be solely used for nudity detection, as it would have a higher false negative rate than OpenNSFW2, because the second is trained solely on that task, thus it is more accurate. Finally, it is important to mention that AI algorithms do not guaranty that all instances of images of the unwanted categories will be identified, as they do not perform perfectly, but the majority of these images will be recognised and removed.

To sum up, with the use of pre-trained ML models we managed to remove images from unwanted categories and create a dataset that is more focused on real photographs and fire events. Next, is the image filtering stage with the use of image statistics.

In this last part of the image filtering stages, we will go over the image statistics criteria we used for the final sieving of the collected images. The image statistics were derived with the help of the FastDup library[41] . When referring to image statistics we mean values that can be derived from an image's features (e.g., mean colour). For the purposes of our filters, we used the features of mean, width, height and blurriness.

Starting from the image size, images with less than 10K pixels were considered too small and discarded as on average a NN has an image input is approximately 50K pixels (224x224). That is, such an image will need to be stretched to fit the desired input size, causing a lot of distortions. Furthermore, small images usually do not contain real scenes, but rather emojis, button images, etc., making them unfit for the dataset. Next were images with irregular aspect ratio, that is, images whose width-height analogy (aspect ratio) is very large or too small. This indicates that the image was altered (e.g., cropped) or is a screenshot, which again indicate that these images are not from occurring events. Thus, images with aspect ratio less than 0.5 or larger than 3 were discarded. Most common aspect ratios[42] have a value between 1 (1:1) and 1.78 (16:9), or 3 in the case of panoramic images (3:1), thus we discarded the more extreme cases.

Regarding the mean and blurriness of the image they were used to filter out images with high frequency of white/black, fake images or images that were already too blurry to begin with. If an images mean was too low (≤ 2) or too high (≥ 200) it was an indication that the majority of the image was covered by black or white, respectively. This step is similar to the spatiality check, which can capture some of the images that *"escaped"* that check or do contain multiple objects but in a monotone background (e.g., advertisements), which again are not related to our task. Also, we noticed that fake/cartoonish images tend to have a very high mean (because of the colour palette they use), which captures with the high threshold (≥200). Next, we noticed that if the blurriness factor was too low (≤ 40) it meant that the image was already very blurry, thus hard to analyse even by humans, therefore it would be even harder for an AI algorithm to distinguish.

This was the final step of the image filtering phase, were images with high black/white frequency, small/irregular size or too blurry were removed from the dataset to make it more relative to our task and reduce outliers.

To conclude, the aforementioned filtering steps serve only as a sieve that collects the majority of the unrelated images to the task, but not all. Reducing the number of images that are processed by the NNs, thus the SILVANUS framework, significantly improves its response time and assists with the early detection of a fire event.

5.2.1.4 *AI model training*

Given the objectives of the project, it was desired to identify fire and smoke particles as soon as possible to reduce their consequences. Therefore, we focused not only on fire detection from images, but also on smoke detection, as it is an early tale-tell of fire. For this purpose, two different AI models needed to be developed, namely a fire and a smoke detection network. Furthermore, two supplementary models were developed for the identification of the fire's/smoke's location, respectively, within the image to offer additional information to the end users (e.g., first responders, fire fighters, etc.). Having the additional information of the fire's/smoke's location within the image will later help with the information fusion (e.g., sec. 5.2.1.2 Visual Concept Extraction) for the risk assessment of an event.

It is important to mention that we emphasised our research on the development of the fire detection model, thus the results presented here will be referring only to fire. Once the fire model is finalised, along with its supplementary fire localisation network, it will be trivial to implement its smoke detection counterpart. The main idea of the AI model pipeline –the one deployed on the SILVANUS framework, is that the trained network will be presented with a series of images to classify into 'fire' or 'no_fire', followed by

[41] Danny Bickson and Amir Alush. 2022. FastDup | A tool for gaining insights from a large image collection. https://github.com/visualdatabase/fastdup (2022)

[42] Photography Life, "The Most Common Aspect Ratios" section. https://photographylife.com/aspect-ratio

the localisation model (only for images classified as 'fire') and finally the presenting of the identified fire events to the UI platform (see sec. 7 Visualization of fire events). For the case of smoke detection, the same pipeline will be applied with the only difference being the use of the smoke detection and localisation networks.

Regarding the selection of the appropriate models, we experimented with different CNN architectures and assessed them based on their prediction accuracy and inference latency. Besides these factors, the chosen model should also be lightweight, as it would also be deployed on hardware-constrained devices (e.g., IoT devices - Task 4.4). Based on the aforementioned criteria, it was decided to use a variation of ShuffleNetV2 (Ma et al., 2018), namely, the ShuffleNetV2-OnFire (Thomson et al, 2020) model, a lightweight network that demonstrated high performance on both fire detection and localisation (using superpixels) tasks. The superpixel segmentation task (see example in Figure 24 (right)) refers to the splitting of an image into superpixel regions, and then their classification into 'fire' or 'no_fire' areas. Adopting a superpixel segmentation algorithm, instead of the conventional with bounding boxes, has the advantage of focusing the network's attention on smaller and more irregular areas of the image (instead of predefined rectangle/bounding box sizes) – making the detection of small fire sources even within dense forestry more accurate.

Figure 24: (left) Example output of the fire binary classification model (with fire probability = 1) and (right) an output example of the fire superpixel localisation algorithm

The chosen model could be used as is, but tuning the network on the SID dataset will familiarise it with our type of data, thus increase its performance. On the other hand, training from scratch the network will erase any learned context from their larger-in-size dataset (over 360K images), that we would not be able to achieve with the data in hand (~20.7K). Consequently, we used the pre-trained binary classification model, with the provided weights released by the developers of ShuffleNetV2-OnFire, as our starting point. To finetune the network, we re-trained its last fully connected layers with the SID dataset, in order for it to accordingly adjust its weights and achieve the best possible performance.

To obtain the best performing variation of the model, we conducted a series of experiments. Specifically, various architecture re-designs were tested, along with different combinations of optimization algorithms and hyper-parameters (e.g., learning rate). The model that achieved the highest performance so far, was trained with Adam optimizer for 200 epochs and a learning rate of 1e-4. Example results of the fire classification and localisation can be seen in Figure 24.

For the training of the fire detection model, the SID dataset was separated into training and validation sets, with a ratio of 85:15 (17625:3111 images). A significant problem that we faced during the training was the

class imbalance[43], a common problem to many datasets that can cause a network to be biased (i.e., have a higher tendency to predict one class over the other). By class imbalance, we are referring to the disanalogous number of images assigned to each class (non-fire images were significantly more frequent than those of fire), which interferes with the network's training. In order to mitigate this issue, and improve the overall performance, we used a weighted loss function. The loss weights for each class were inversely proportional to their frequency (i.e., $1/(\#class\ images)$).

The following graphs (Figure 25) present the learning curves acquired during the binary classification model's training on our dataset. The fact that the validation accuracy curve tracks the training accuracy so good, indicates that our model capacity is not high enough to capture all the information, and so we might need to increase the number of parameters in future trials. Though, increasing the number of parameters has the trade-off of also increasing the network's size, which is not desired as the model will be deployed on hardware-constrained devices, meaning further experimentation is needed to find the perfect balance between network capacity and size.

The prediction accuracy of the model on the validation data is 91% and from the confusion matrix (Figure 26) we notice that the prediction accuracy of the positive class ('fire') is 79%, while on the negative class is 94%, giving us a good ground to build upon.

In the following months, we will continue experimenting with different versions of the ShuffelNetV2-OnFire, by also introducing to it more challenging data (fire-like objects, e.g., sunset), to increase its robustness. Following the network finetuning, we will also try to experiment with the superpixel segmentation algorithm by tweaking some of its parameters (e.g., size of patches) to better localise fires in difficult environments/backgrounds (e.g., yellowish fields). Once we are satisfied with the fire models, we will start with their adjustment for the smoke detection task to complete the given task.

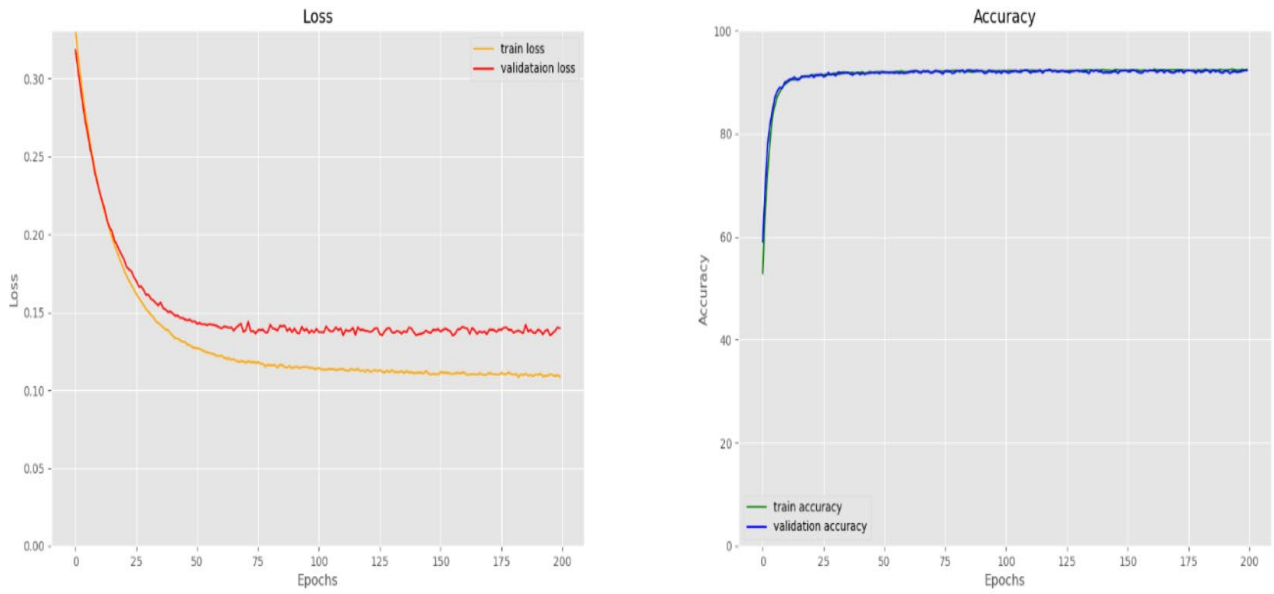


Figure 25: (left) Loss curve and (right) accuracy of ShuffleNetV2 OnFire while being re0trained on SID dataset

[43]Medium, https://medium.com/gumgum-tech/handling-class-imbalance-by-introducing-sample-weighting-in-the-loss-function-3bdebd8203b4

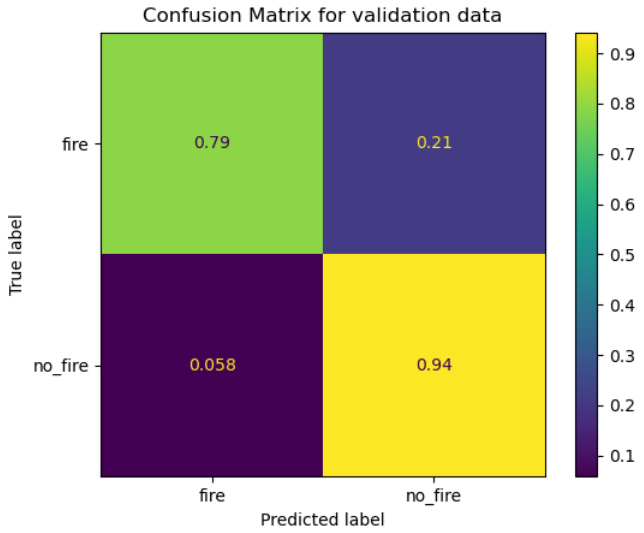


Figure 26: Confusion matrix of ShuffleNetV2 OnFire tested on validation set of SID dataset

5.2.2 Visual Concepts Extraction

5.2.2.1 *Concept detection module*

CERTH's Concept detection is the process of categorizing new observations into known classes by constructing a model using a set of training data, and then using the model to classify new data. The classifier, which maps observations to predefined classes, uses a feature vector to describe the measurable properties of each instance. Feature values may be binary, categorical, integer-valued, or real-valued, and can correspond to pixel values in images or word occurrence frequencies in text.

The implementation is based on a framework that uses Caffe (Jia et al., 2014) and involves the use of a 22-layer GoogleNet network which was trained on a subset of 5055 ImageNet concepts, following rules to merge similar concepts and remove scientific terms and concepts with few positive images. The resulting classification layer had dimensionality of 5055, which was reduced to 345 to target the TRECVID Semantic concepts[44] (Smeaton et al., 2009). The concept extraction module receives an image as input, tests it using the fine-tuned DCNN, and produces a list of concepts and their probabilities. The framework uses a double threshold to limit the number of concepts and only consider the top 10 concepts with higher probabilities above 0.1, based on the observation that 10 tags are sufficient for image description.

The visual concept extraction can quickly and easily analyse the images associated with posts about your use case, giving insights and a better understanding of what people are saying. A module is employed for the analysis and extraction of visual concepts from social media, which retrieves high-level information (i.e., concepts) from the images associated with the posts. These concepts can be utilized as a means to assess if the images are pertinent to the topics that are of interest (e.g., image related to fire incident) or to retrieve similar content (e.g., query tweets that have fire related images). This module, developed within the framework of the European project EOPEN.

Visual concept extraction module can be used as a standalone web service. All that's required is a URL of an image from social media, and the service will return a JSON string with a list of the top ten concepts extracted from the image. An example showcasing the concepts extracted from post images using this service is provided below:

[44] http://www-nlpir.nist.gov/projects/tv2012/tv11.sin.500.concepts_ann_v2.xls

```
"concepts": [
            "smoke",
            "fire_truck",
            "Explosion_fire",
            "Scene_Text",
            "factory",
            "person",
            "outdoor",
            "Running",
            "Building",
            "male_person",
        ]
```


Table 28: An example of concepts extracted from an image

5.2.2.2 *Object recognition*

To complement the classification of images for the presence of fire and/or smoke (and their localisation within the image), HB focused on the recognition of other depicted objects in order to measure the severity of an event. Namely, our target was to locate people and vehicles in images by means of object detection that would serve as another parameter to the fire event's severity assessment. For example, if people are spotted near the fire, the event's severity is higher because lives are at risk and imminent help is needed, or in a case of a vehicle close to a fire, further explosions can be caused, which again increases the scene's severity.

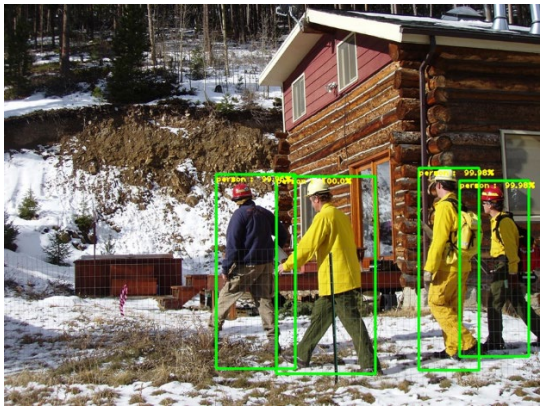
Object detection is the process of segmenting an image into objects that are in turn assigned class labels with probabilities. There exist excellent Python libraries and pre-trained models to that end, that can be employed with just a few lines of code.[45] These models are examples of vision transformers, i.e., the Transformer architecture applied to image classification, trained on ImageNet[46], which is the largest publicly available dataset of annotated images.

We received two image datasets, the first collection harvested on Flickr (CTL, 6414 images categorized into groups with fire, smoke, both, vs. none), the second one on Twitter (CERTH, annotated by CTL, cca 20K images). In the first experiment, a deep learning (DL) architecture called BEIT[47] (Bao et al., 2021), a theoretical relative of BERT.

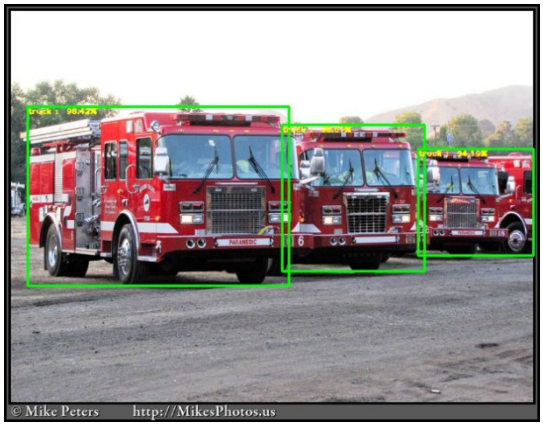
In the second round, on the same dataset, we ignored the above four categories and refocused on humans and vehicles instead. This time the Python library called ImageAI[48] (Moses, 2018), together with the YOLOv3[49] object detection algorithm ((Redmon and Farhadi, 2018) Figure 27 *a-c).* Future work will test these algorithms on the Twitter image dataset as well.

[45] https://huggingface.co/google/vit-base-patch16-224
[46] https://www.image-net.org/
[47] https://huggingface.co/microsoft/beit-base-patch16-224-pt22k-ft22k
[48] https://github.com/OlafenwaMoses/ImageAI
[49] https://pjreddie.com/darknet/yolo/

a) Identified vehicles in image



b) Identified persons in image



c) Identified trucks in image

Figure 27: Detection of humans and vehicles with high certainty by a combination of ImageAI and YOLOv3

5.2.2.3 *Concept extraction from images and text combined*

For this task, ATOS created a tool able to combine the text and the images in search of fire related information. This tool is primarily based on the use of BLIP (Bidirectional Pretrained Transformer for Multimodal Learning) (Li et al, 2022). BLIP is part of the text and image cross training revolution that has his most well-known architecture in CLIP ((*Contrastive Language–Image Pre-training*) released in 2021.

BLIP is a deep learning algorithm that combines text and image information in a bidirectional transformer architecture to perform multiple tasks such as visual question-answering and image captioning (Figure 28) for a real example). The model is trained on large amounts of text and image data and fine-tuned on specific tasks by adjusting the result. The bidirectional structure allows the model to use information from both text and image modalities to make predictions, and the transformer architecture helps capture complex relationships between the two modalities.

Question:
['It this a photo of a wildfire?','Is there any s
moke?','What time is this during the day?',

'How long has the fire been burning?`]

Answer: [['yes'], ['yes'], ['afternoon'], ['long
time']]

Figure 28: Example of Question-Answer using BLIP

The evolution of these algorithms started with the Transformer architecture, introduced in (Vaswani et al., 2017). The Transformer revolutionized the field of natural language processing (NLP) by allowing models to efficiently handle long-term dependencies in sequential data, such as text. This Transformer architecture has been successfully extended to image, sound and other media and also combinations like text and image.

Based on CLIP, in SILVANUS we have created a tool to extract fire-related information from information taken from social media. For this, we need text and images where we can extract, primarily the information of the existence of a fire and, secondarily, all kind of meta-information that can be used to enrich the information like geographical positions, possible elements affected, etc. (Figure 29)

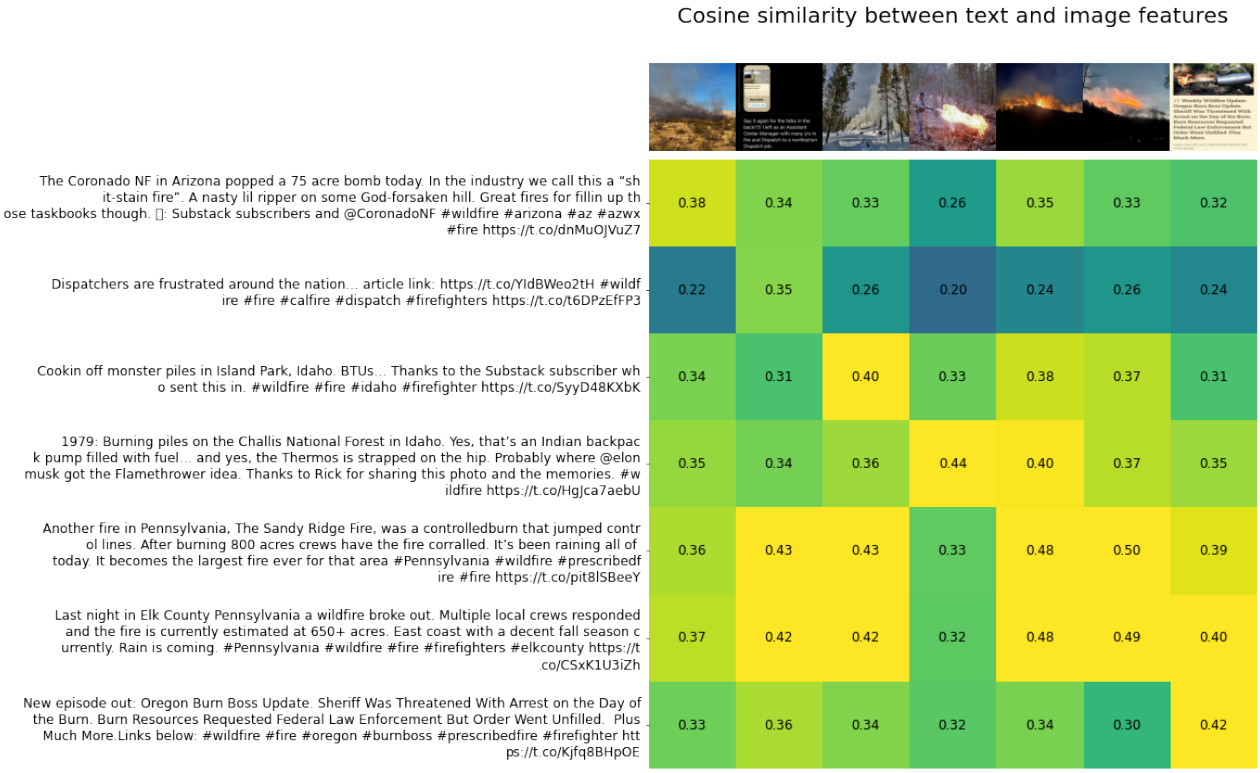


Figure 29: Extraction of information cross-checking image and text

5.2.3 Localization estimation from images

ATOS have developed a tool for estimate the geographical position of the images presented in social media. The idea is to use this information as a support in case of a media post containing an alert about a fire. The images attached to the post, can provide us an extra information about the approximate position of the fire, thus enriching the information available. Note that the outcomes of this tool are just indicative, since there is a great imprecision in the results available. This imprecision varies depending on the image (depending on amplitude of the image, the presence of landmarks, etc). The images have to be “outdoors” images.

This tool has been created based on the paper by Müller-Budack et al. (2018). This paper presents a deep learning approach for estimating the geolocation of photos. The algorithm created uses a hierarchical model that first classifies the scene in the image and then estimates the geolocation based on the scene classification.

The model consists of two parts: a scene classification network and a geolocation estimation network. The scene classification network is trained to recognize the scene in the image and outputs a scene feature vector. The geolocation estimation network takes the scene feature vector as input and outputs the estimated geolocation. The model was trained using nearly 5 million geo-tagged images classified in 26263 areas covering the whole planet.

In the Figure 30 we can see a real result of the use of the tool (in this case with a very precise result)

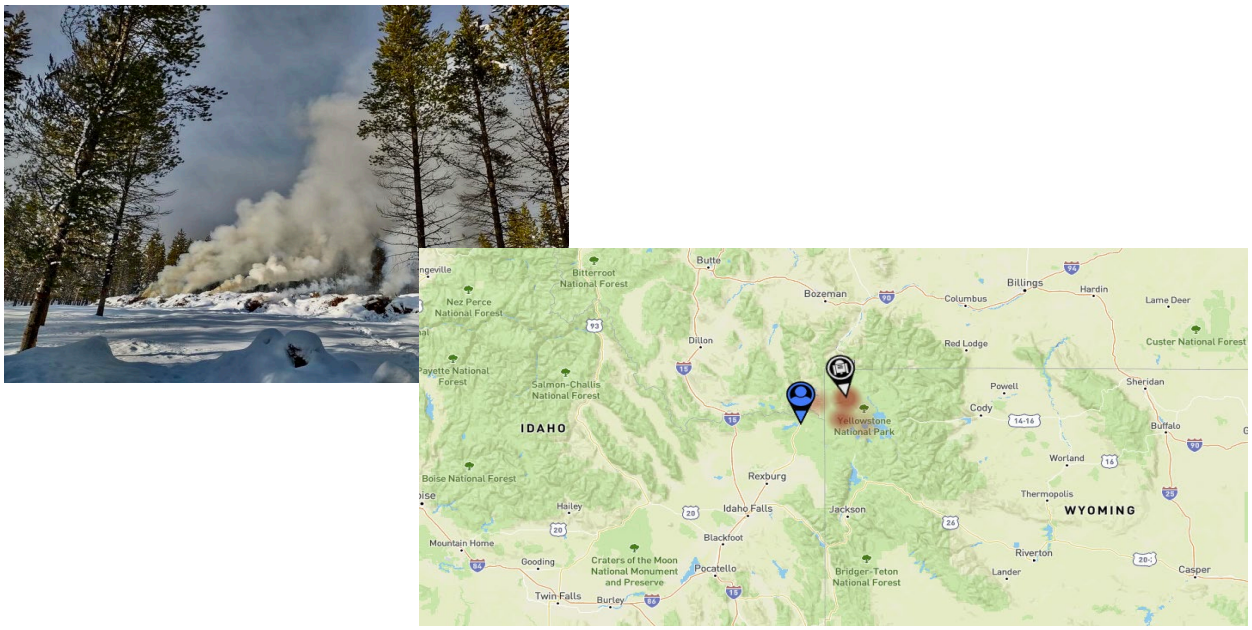


Figure 30: Geo-location of fire using image

6 Fire Events stored to knowledge database.

This section provides a comprehensive overview of the JSON structure of single social media posts that are acquired by the Fire Detection module from the Social Media Sensing API. Additionally, it describes the JSON structure used to represent fire events and how these structures are transformed into RDF semantic representations for storage in the knowledge database.

6.1 JSON structure of the social media posts and the fire events.

This sub-section presents a brief description of the JSON structure of the social media posts have after they are enriched with additional knowledge from the analysis modules of the Social Media Analysis Toolkit (see section 2).

Firstly, social media crawlers (see section 4.1) collect posts based on defined search criteria (see section 3). Subsequently, each post undergoes analysis by the modules of the Social Media Analysis Toolkit, which are accessed via simple HTTP requests. During this analysis, posts are enhanced with additional fire-related knowledge and stored in a mongoDB database.

To detect fire events, the Fire Event Detection module frequently queries the collection of individual tweets through the Social Media Sensing API. The social media posts are forwarded to the Fire Event Detection module in common JSON representation identical for all the social media post collected from all the crawlers. The Table 29 below shows the common JSON structure for single posts.

Single post				
Attribute Name	**Source**	**Attribute type**	**Description**	**Example Value(s)**
id	Crawlers	String	A unique identifier for each post	"1474011868100448259"
text	Crawlers	String	The text of the post	"The Bolt Creek wildfire in Washington state has grown to an estimated 7,600 acres"
timestamp	Crawlers	String	Date and time of the publication of the post, in the following format: YYYY-mm-ddTHH:MM:ssZ	"2022-09-07T15:00:00Z"
media_url	Crawlers	String Array	The URL(s) of the media that are attached to the post	["https://pbs.twimg.com/tweet_video_thumb/E-mfl0aXIAE72Jm.jpg", ...]
media_type	Crawlers	String	The type of the media that are attached to the post	"image"
is_retweet	Crawlers (Twitter)	Boolean	Whether the post is a retweet	true/false
is_quote	Crawlers (Twitter)	Boolean	Whether the post is a quote	true/false

language	Crawlers	String	The language of the post's text	"en"
platform	Crawlers	String	The platform that the post originates from	"twitter", "facebook", etc.
analysis_concepts	(Textual-Visual) Concepts Extraction	JSON Object Array	The concept extraction analysis from the post's text or media or both as JSON obects with the attributes "analysis_type", "concepts" and "metadata"	[{ "analysis_type":"Textual", "concepts":["wildfire state","affected_human","smoke","..."], "metadata":[{ "concept_name":"wildfire state","prob":"0.98"},{"concept_name":"..."},{"concept_name":"smoke","prob":"0.72"}]}]
analysis_type		String Array	The information is extracted from visual or textual analysis or both	
concepts		Array	The concepts that can be extracted from the post's text or media or both	
metadata		JSON Object Array	Additional information that is included in each textual analysis module (e.g. ml probabilities)	
text_is_relevant	Relevance Classification	JSON Object Array	Whether the post is relevant or not to our use cases and containt the attribute "fire_related"	[{"fire":true}]
fire		Boolean	Shows if the text of post is fire related	
text_categories	Text Categorization	JSON Array	The categories that the post's text refer to.	[{ "category_name": "wildfire", "score": "0.98"},
event_type	Event Recognition	String Array	The type of the events the post refers to.	{"events": [{“event_name”: “fire”,“score”: 30,“positions”: […]}]} “Positions” refers to the indexes of the whole text and of the tokens that influenced the resulting event

media_contains	Fire & smoke Detection	String	Whether the post's media display a fire and/or smoke	"Fire" or "Fire-Smoke" or "Smoke"
extracted_locations	Locations Extractions (Textual-Visual)	JSON Object Array	The locations extracted from the post's text, represented as JSON objects with the attribute's "placename" and "geometry"	[{"analysis_type":"Textual", "placename" : "Triest, Trieste, Friuli-Venezia Giulia, 34121-34151, Italy", "geometry" : {"type" : "Point", "crs": "WGS84", "coordinates" : [{"lat": 13.7772781,"lon": 45.6496485}]}}, ...]
analysis_type		String Array	The information extracted from visual or textual analysis or both	"Textual" or "Visual" or "Textual-Visual"
placename		String	The name of a location	"Triest, Trieste, Friuli-Venezia Giulia, 34121-34151, Italy"
geometry		JSON Object	A JSON object with the attribute's "type" and "coordinates"	{"type" : "Point", "crs": "WGS84", "coordinates" : [...]}
type		String	This always has value "Point" to support GeoJSON	"Point"
crs		String	The type of the coordinates reference system that is used.	"crs": "WGS84" or "crs": "EPSG:5514"
coordinates		JSON Object Array	The pair of the coordinates of a location contained in the attributes "lat" and "lon"	"coordinates" : [{"lat": 13.7772781,"lon": 45.6496485}
lat		Double	The longitude coordinates	13.7772781
lon		Double	The latitude coordinates	45.6496485

Table 29: JSON structure for the enhanced single posts

In addition, once the Fire Event Detection module identifies a fire event, it is not only stored in a separate collection in mongoDB, but it is also sent to the knowledge graph for further utilization by other SILVANUS modules. A fire event is characterized by the identification of a group of social media posts that pertain to a fire incident. These events are transmitted in JSON format to CTL's CASPAR tool(as described in section 6.2) to be integrated into a semantic model and stored in the knowledge base. The JSON structure of fire events is provided in the Table 30 below.

Fire event (group of posts)				
Attribute name	**Source**	**Attribute type**	**Description**	**Example value(s)**

id	Fire Event Detection	String	A unique identifier for each event	"T20220419160000"
posts		JSON Object Array	All the posts that constitute the event, as "Single post" JSON Objects (above)	
timestamp		String	Date and time of the creation (detection) of the event, in the following format: YYYY-mm-ddTHH:MM:ssZ	"2022-04-19T16:00:00Z"
gallery		String Array	The URL(s) of the media that are attached to all the posts of the event	["https://pbs.twimg.com/tweet_video_thumb/E-mfl0aXIAE72Jm.jpg", ...]
event_description		String	A short (automatic) description of the event	"Possible forest fire in Triest"
tags		String Array	A combination of all the textual and visual concepts extracted from the posts of the event	["building on fire", "animal in danger", "smoke", "car", "person", "daylight", ...]
event_type		String	The type of the event, based on the recognized event types of the posts of the event	"emergency"
location		JSON Object	The location of the event, based on the extracted locations of the posts of the event	{ "placename" : "Triest, Trieste, Friuli-Venezia Giulia, 34121-34151, Italy", "geometry" : {"type" : "Point", "crs": "WGS84", "coordinates" : [{"lat": 13.7772781,"lon": 45.6496485}]}}

Table 30: JSON structure for the fire events

It should be noted that the JSON structure mentioned above and the semantic representation described in subsequent sections are currently in the development phase and are subject to change in the future. The JSON structures provide a solid foundation for establishing a common information representation for the communication between the various T4.4 SILVANUS modules.

6.2 JSON-to-RDF Mapping

This sub-section briefly presents the adopted semantic data integration process of CERTH social-sensing (T4.4) within SILVANUS; a more detailed account of this work will be given in the upcoming project deliverable D5.4.

Semantic Data Integration is the process of bringing together data from various and potentially disparate sources into a cohesive representation. This is achieved through the use of a data-driven architecture that is based on a semantic model, which is commonly built using the Resource Description Framework (RDF), a W3C standard for knowledge representation on the web. In this framework, resources are represented as subject-predicate-object triples and are stored in a special database called a triplestore. RDF-based semantic models are known as ontologies and play a crucial role in the integration of heterogeneous data from different sources into a unified representation. The ability to easily import, standardize and interlink data into an RDF triplestore is a key aspect of knowledge management solutions that rely on semantic models.

For this task, the process of populating the initially empty ontology (D3.1) within the SILVANUS project is managed by the CTL's CASPAR (Structured Data Semantic Exploitation Framework), a powerful tool for integrating structured data into a semantic model, which is being extended for the specific needs of SILVANUS. The CASPAR architecture is illustrated in Figure 31 .

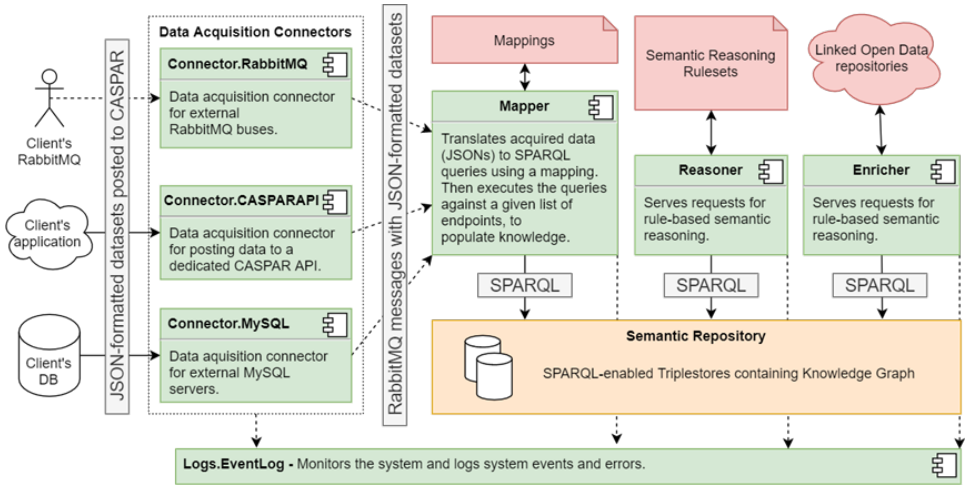


Figure 31: Block diagram portraying the CASPAR architecture

CASPAR has a set of mechanisms for acquiring structured data from various sources, mapping input data to semantic entities, integrating knowledge into a semantic repository, enriching existing knowledge from linked open data sources, and performing rule-based reasoning to generate new knowledge. CASPAR uses a domain-specific language based on JSON syntax to define mappings between input data and ontology concepts, and templates, individuals, and properties serve as the building blocks of this mapping. Templates focus on specific parts of the input data, individuals declare nodes to be created or updated in the semantic knowledge graph, and properties indicate desired associations between nodes or with literal values.

CASPAR transforms inputs into representations that are in line with the RDF through user-defined mappings. These mappings match input data fields to semantic entities such as concepts and relationships.

The mapping used to change the JSON excerpt in section 6.1 into SPARQL queries, which are then employed to fill the semantic model with the appropriate instance data, is shown in the following listing.

CERTH social-sensing mapping:

```
{
    "templates": [
        {
            "prefixes": [
                {
                    "prefix": "rdf",
                    "namespace": "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                },
                {
                    "prefix": "rdfs",
                    "namespace": "http://www.w3.org/2000/01/rdf-schema#"
                },
                {
                    "prefix": "xsd",
                    "namespace": "http://www.w3.org/2001/XMLSchema#"
                },
                {
                    "prefix": "xml",
                    "namespace": "http://www.w3.org/XML/1998/namespace"
                },
                {
                    "prefix": "owl",
                    "namespace": "http://www.w3.org/2002/07/owl#"
                },
                {
                    "prefix": "silvanus_demo",
                    "namespace": "http://www.semanticweb.org/SILVANUS_onto_demo#"
                }
            ],
            "individuals": [
                {
                    "path": "posts.[*]",
                    "namespace": "http://www.semanticweb.org/SILVANUS_onto_demo#",
                    "classes": ["silvanus_demo:Post"],
                    "properties": [
                        {
                            "predicates": [
                                "silvanus_demo:hasID",
                                "rdfs:label"
                            ],
```

```
            "object": {
                "path": "posts.[*].id",
                "datatype": "xsd:string"
            }
        },
        {
            "predicates": ["silvanus_demo:hasText"],
            "object": {
                "path": "posts.[*].text",
                "datatype": "xsd:string"
            }
        },
        {
            "predicates": ["silvanus_demo:hasTimestamp"],
            "object": {
                "path": "posts.[*].timestamp",
                "datatype": "xsd:dateTime"
            }
        },
        {
            "predicates": ["silvanus_demo:hasAbsoluteMediaURL"],
            "object": {
                "path": "posts.[*].media_url.[*]"
            }
        },
        {
            "predicates": ["silvanus_demo:hasMediaFormat"],
            "object": {
                "path": "posts.[*].media_type"
            }
        },
        {
            "predicates": ["silvanus_demo:isRetweet"],
            "object": {
                "path": "posts.[*].is_retweet",
                "datatype": "xsd:boolean"
            }
        },
        {
            "predicates": ["silvanus_demo:isQuote"],
            "object": {
                "path": "posts.[*].is_quote",
                "datatype": "xsd:boolean"
```

```
            }
        },
        {
            "predicates": ["silvanus_demo:hasLanguage"],
            "object": {
                "path": "posts.[*].language"
            }
        },
        {
            "predicates": ["silvanus_demo:hasPlatform"],
            "object": {
                "path": "posts.[*].platform"
            }
        },
        {
            "predicates": ["silvanus_demo:isRelevant"],
            "object": {
                "path": "posts.[*].is_relevant",
                "datatype": "xsd:boolean"
            }
        },
        {
            "predicates": ["silvanus_demo:isFake"],
            "object": {
                "path": "posts.[*].is_fake",
                "datatype": "xsd:boolean"
            }
        },
        {
            "predicates": ["silvanus_demo:hasTextCategory"],
            "object": {
                "path": "posts.[*].text_categories.[*]"
            }
        },
        {
            "predicates": ["silvanus_demo:hasTextualConcept"],
            "object": {
                "path": "posts.[*].textual_concepts.[*]"
            }
        },
        {
            "predicates": ["silvanus_demo:hasVisualConcept"],
            "object": {
```

```
                "path": "posts.[*].visual_concepts.[*]"
            }
        },
        {
            "predicates": ["silvanus_demo:hasEventCategory"],
            "object": {
                "path": "posts.[*].event_type.[*]"
```

Table 31: An example of RDF mapping of CERTH social media JSON

The task of ontology population involves creating new nodes and edges in the knowledge graph. Mappings are written in JSON format and consist of templates, individuals, and properties:

- Templates allow for focusing on specific parts of the input data.
- Individuals declare the nodes to be created or updated in the knowledge graph, where they represent instances of one or more classes in the ontology.
- Properties specify the edges to be added in the knowledge graph, connecting nodes or linking a node with a literal value. Properties can correspond to object properties (relating two instances) or datatype properties (relating an instance to a literal value). They are defined by a set of predicates (relationship types) and objects (values to be assigned to the property). Objects, which are specified as JSON paths, can either point to literal values in the input or to other individuals.
- The update_on field is an optional feature that allows for linking new knowledge with existing nodes in the knowledge graph.

6.3 Knowledge Bases

The CASPAR framework utilizes Ontotext's GraphDB as its primary triplestore for populating the ontology. GraphDB is a conformant implementation of the W3C's RDF and OWL standards, the de-facto standards for representing and managing linked data. With its advanced performance, scalability and enterprise-readiness, GraphDB is equipped to handle the large volumes of data generated by the CASPAR framework's semantic repository's knowledge graphs.

The triplestore's powerful inference engine allows for real-time inferencing and reasoning over the ontology, ensuring that the framework can discover new knowledge and relationships in an efficient and scalable manner. GraphDB's conformance to W3C standards further enhances the interoperability of the CASPAR framework utilizing state-of-the-art technology and adhering to industry standards. The use of GraphDB reinforces the framework's position as a robust, scalable and standards-compliant solution for semantic data integration and management.

An ontology was created for the CASPAR framework's social media sensing component, merging with the upper-level ontology in D3.1 to provide a consistent representation of data. This unified and standardized representation provides a robust foundation for data analysis and inferencing, supporting social media sensing and analysis tasks effectively. The integration of the ontology demonstrates the framework's commitment to providing a comprehensive and standards-compliant solution for semantic data integration and management.

Figure 32 demonstrates how the sample observations in section 6.1 are integrated into the knowledge graph using the Graffoo[50] format after being processed by CASPAR.

[50] https://essepuntato.it/graffoo/

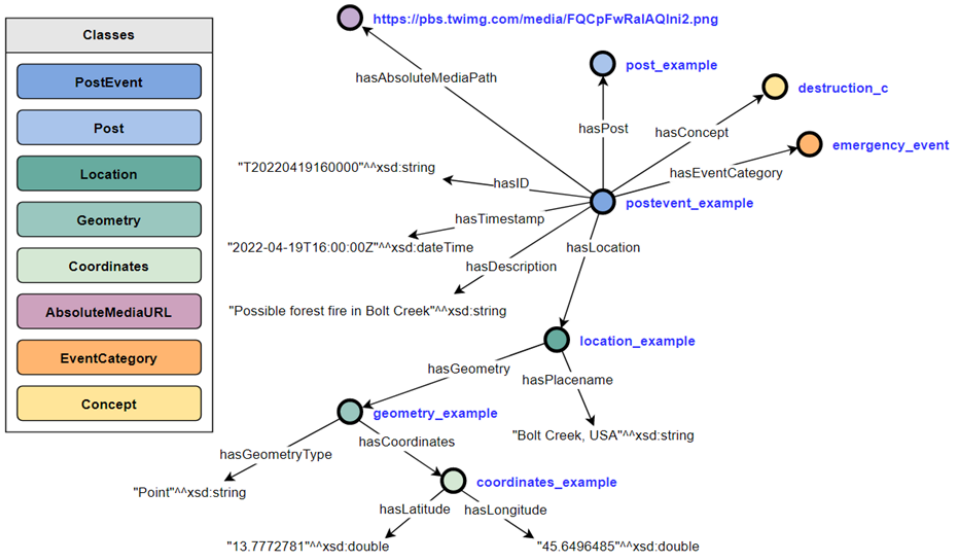


Figure 32: Representation of the sample observations in the KG

7 Visualization of fire events

After the fire incidents are detected, they need to be properly stored and organized in order to be easily retrieved and analyzed. In this case, the fire incidents are mapped from JSON format to RDF representation, which allows for a more structured and semantic way of representing the data. This makes it easier to search and query the data using SPARQL, a query language specifically designed for querying RDF data.

Once the fire incidents are stored in graphDB (see section 6.3), they are ready to be requested by the SILVANUS visualization Dashboard. The Dashboard (Figure 33) serves as a user interface that allows users to interact with the fire-related information collected by the project. The platform is designed with multiple layers, each exhibiting different types of fire-related information.

One of these layers is the Social Media Sensing layer, which contains the fire incidents detected by the Fire Event Detection module. These fire incidents are represented as pins on a map, allowing users to see the location of each incident. When a pin is clicked, a popup view appears, displaying additional information about the fire incident.

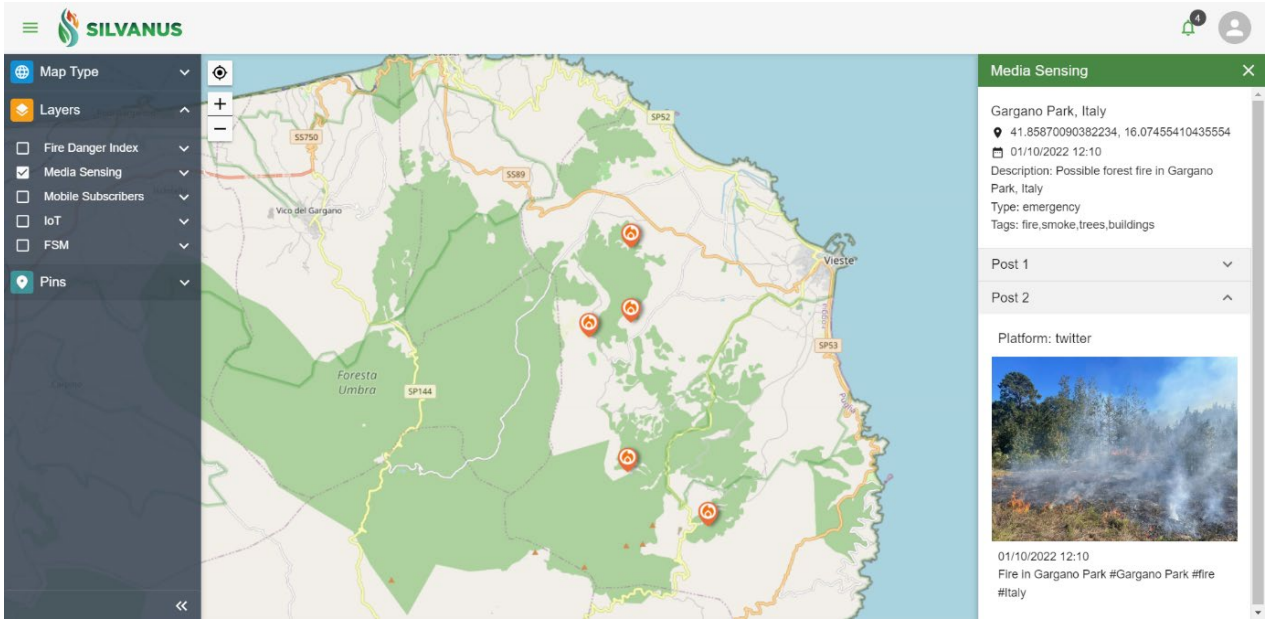


Figure 33: The Dashboard – Social Media Sensing layer

The popup view (Figure 34) is an essential feature of the SILVANUS visualization platform, as it provides users with a more detailed view of the fire incidents detected by the Fire Event Detection module. The popup view is designed to contain a wealth of information, including:

- *Social media posts*: The social media posts from which the event was detected are displayed in the popup view. This information is important, as it allows users to understand the context in which the event was detected.
- *Images*: The popup view may also include an image of the possible event, giving users a visual representation of the fire incident.
- *Social media platform*: The social media platform where the posts are collected is also displayed in the popup view. This information can be helpful for users who want to understand the demographics of the social media users who are reporting the fire incidents.
- *Location*: The location of the fire incident is displayed on the popup view, allowing users to see where the fire is located and how close it is to other important locations or landmarks.
- *Date and time*: The date and time the incident was detected is also included in the popup view, giving users a sense of the timeline of the event.

- *Description*: The popup view may also contain a description of the event, which can provide additional context and details about the fire incident.
- *Predicted type*: The predicted type of the fire incident is also displayed in the popup view, which can be helpful for users who want to understand the potential impact of the fire.
- *Visual and textual analysis*: Finally, the popup view may include visual and textual analysis in the form of tags, which can help users quickly identify key characteristics of the fire incident, such as its severity or the type of terrain it is located in.

All of this additional information helps users to better understand the scope and impact of the fire incident and can ultimately aid in the prevention and mitigation of future fires.

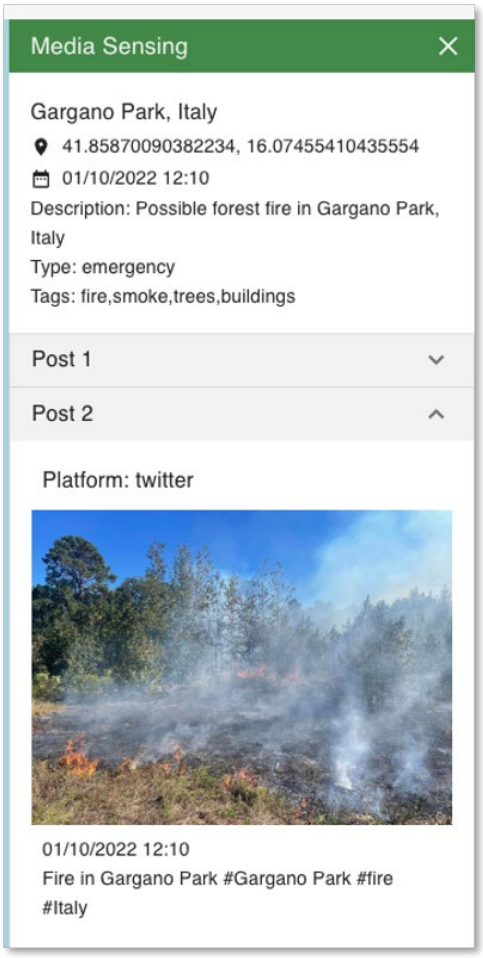


Figure 34: The information for a Twitter fire event (mock up)

Overall, the SILVANUS visualization Dashboard serves as a valuable tool for understanding and analysing fire-related data from social media. By organizing and displaying the data in an intuitive and user-friendly way, the platform helps users to better understand the scope and impact of the fire incident and can ultimately aid in the prevention and mitigation of future fires.

8 Crawling Results

The Twitter data that was collected and stored from the beginning of the crawl until February 28, 2022. The crawling process started on December 23, 2021, and over this short period of time has already amassed approximately 1 million fire related tweets for pilots P01, P03, P07, P09 and P10.

As demonstrated in Table 32, English is the most widely used language, garnering more tweets in all Pilots as expected. The second most popular language is Italian, with about 30,000 tweets, followed by Greek with roughly 12,000. Unfortunately, it seems Indonesian Twitter users do not often engage in discussions or reporting of fire related issues on the platform. However, Italian and Greek Twitter users appear to be quite active in the topic of fire incidents.

Table 32 showcases the initial findings of the Twitter crawl. It's evident that there is a considerable difference in the number of retrieved tweets between Pilot P07 and the rest. This disparity can be attributed to the broader search criteria employed in Pilot P04, which includes a wider range of keywords and accounts. However, as the Twitter crawler continues to be optimized with the addition of new keywords to other Pilots and the removal of certain keywords from P07, it is expected that this gap will be reduced over time.

Furthermore, the location information directly obtained from Twitter's JSON is limited, with only 0.1% to 0.8% of tweets being geolocated. However, by utilizing CERTH Localization web service, the number jumps to a substantial 24.2% to 49.5% of tweets with geolocation. The presence of geotagged social data can prove to be extremely valuable, as it can be combined with other sources, such as satellite data. This highlights the fact that Twitter does not possess a significant number of geotagged tweets, making the utilization of the localization web service crucial for geotagging tweets. Furthermore, it's worth noting that 5.7% to 10.8% of the tweets contain at least one image. This seemingly small amount of data actually holds a wealth of visual information that can be leveraged for comprehensive visual analysis through the use of the Social Media Analysis Toolkit.

Pilot	language	Collected Tweets	Location from Twitter	Location from CERTH's Localization	Contain Image
P01	English	40,220	120 (0.3%)	9,733 (24.2%)	914 (8.9%)
	Italian	30,656	122 (0.4%)	15,196 (49.5%)	1749 (5.7%)
P03	English	63,047	378 (0.6%)	16,898 (26.8%)	4933 (7.8%)
P07	English	329,587	2,636 (0.8%)	88,322 (26.7%)	19154 (5.8%)
	Greek	11,802	11 (0.1%)	4,371 (37.0%)	1283 (10.8%)
P09	English	7,487	44 (0.6%)	2,629 (35.1%)	637 (8.5%)
P10	English	10,236	51 (0.5%)	2,480 (24.2%)	916 (8.9%)
	Indonesian	0	0 (0.0%)	0 (0.0%)	0 (0.0%)

Table 32: Statistics for collected tweets

9 Concluding remarks

Wildfires have significant impacts on global ecosystems, economies, and communities. They can destroy homes, businesses, and infrastructure, resulting in billions of dollars in economic losses. They also threaten public health by releasing harmful smoke and particulate matter into the air, displace people, cause casualties, and destroy natural habitats.

Social media can play an important role in supporting wildfire detection and response efforts. By enabling real-time communication and information sharing, social media can help to improve public safety and protect communities from the devastating impacts of wildfires. Additionally, it can help to raise awareness about wildfires and encourage individuals to take steps to reduce their risk of starting or spreading fires.

The objective of the SILVANUS social media sensing is to explore the social media to obtain citizen observations related to fire, extract higher knowledge from the visual and textual content of the social media posts in order to contribute to early-stage detection of wildfires by detecting fire events and visualize them in Dashboard to better understand the scope and impact of the fire incidents. Ultimately, this process can help with the prevention and mitigation of future fires.

The components, functionality, and technical details of the process are presented in detail in this text, including the Social Media Crawlers (Twitter, Facebook, Web), the Social Media Analysis Toolkit components, the Fire Events Detection module, JSON-to-RDF MAPPING, Knowledge Base, and the User Interface (Silvanus Dashboard). Relevant co-design and co-creation activities are carried out, and the interaction and communication results are provided in the text.

The future technological developments and the further refinements actions of the already defined data structures and developed social media sensing modules that will take place in the future are:

- Implementation of all visual and textual analysis components from the Social Media Analysis Toolkit.
- Integration of all the Social Media Analysis Toolkit components to the collected data from social media posts.
- Implementation of the API that forwards social media posts and stores fire events to the mongoDB to and from the Fire Event Detection module.
- Implementation of the Fire Event Detection module in order to detect fire event from social media posts.
- Further, refinement of the search criteria in order to reduce the irrelevant social media posts (reduce noise).
- Updates in JSON format and RDF mapping of the social media posts and the fire events.

10 References

Alireza Shamsoshoara, Fatemeh Afghah, Abolfazl Razi, Liming Zheng, Peter Fulé, Erik Blasch, November 19, 2020, "The FLAME dataset: Aerial Imagery Pile burn detection using drones (UAVs)", IEEE Dataport, doi: https://dx.doi.org/10.21227/qad6-r683.

Angelov, D. (2020). Top2vec: Distributed representations of topics. arXiv preprint arXiv:2008.09470.

Blei, D. M., & Lafferty, J. D. (2006). Dynamic topic models. Proceedings of the ICML. ICML'06. pp. 113–120. doi:10.1145/1143844.1143859

Blei, D.M., Ng, A.Y., Jordan, M.I., Lafferty, J. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research. 3: 993–1022. doi:10.1162/jmlr.2003.3.4-5.993.

Campello, R.J.G.B., Moulavi, D., & Sander, J. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science, vol 7819. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_14.

CAZZOLATO, M.T.; AVALHAIS, L.P.S.; CHINO, D.Y.T.; RAMOS, J.S.; SOUZA, J.A.; RODRIGUES-Jr, J.F.; TRAINA, A.J.M.. FiSmo: A Compilation of Datasets from Emergency Situations for Fire and Smoke Analysis. In: SBBD2017 - SBBD Proceedings of Satellite Events of the 32nd Brazilian Symposium on Databases - DSW (Dataset Showcase Workshop). Available at: sbbd.org.br/2017/wp-content/uploads/sites/3/2017/10/proceedings-satellite-events-sbbd-2017.pdf. SBC. 2017.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785-794)

D. Y. T. Chino, L. P. S. Avalhais, J. F. Rodrigues and A. J. M. Traina, "BoWFire: Detection of Fire in Still Images by Integrating Pixel Color and Texture Analysis," 2015 28th SIBGRAPI Conference on Graphics, Patterns and Images, Salvador, Brazil, 2015, pp. 95-102, doi: 10.1109/SIBGRAPI.2015.19.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1810.04805

Eklund, J., & Darányi, S. (2023). Cross-lingual tweet categorization using multilingual sentence embedding models. Short paper to be submitted to ASIS&T 2023, London, 27-31 October.

Gerlach, M., Peixoto, T. P., & Altmann, E. G. (2018). A network approach to topic models. Science advances, 4(7), eaaq1360.

Grootendorst, M. (2020). KeyBERT: Minimal keyword extraction with BERT. https://doi.org/10.5281/zenodo.4461265

Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv:2203.05794.

Hangbo Bao, Li Dong, Furu Wei (2021). BEiT: BERT Pre-Training of Image Transformers. https://arxiv.org/abs/2106.08254

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S. and Darrell, T., 2014. "Caffe: Convolutional architecture for fast feature embedding", In Proceedings of the 22nd ACM international conference on Multimedia (pp. 675-678). ACM.

Kleinberg, J. (2002). Bursty and Hierarchical Structure in Streams. Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining.

Laborde, Gant. 2020. Keras model of NSFW detector. https://github.com/GantMan/nsfw_model (2022)

Li, J., Li, D., Xiong, C., & Hoi, S. (2022, June). Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In International Conference on Machine Learning (pp. 12888-12900). PMLR.

Li, Songbin, Qiandong Yan, and Peng Liu. "An efficient fire detection method based on multiscale feature extraction, implicit deep supervision and channel attention mechanism." IEEE Transactions on Image Processing 29 (2020): 8467-8475. Available at: http://www.nnmtl.cn/EFDNet/

Ma N., Zhang X., Zheng H. T., Sun J., "ShuffleNetV2: Practical Guidelines for Efficient CNN Architecture Design", Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 116-131

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Vol. 1. University of California Press. pp. 281–297.

McInnes, L., & Healy, J. (2017). Accelerated hierarchical density based clustering. In IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 33-42)

McInnes, L., & Healy, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, ArXiv e-prints 1802.03426

Moses, O. (2018). ImageAI, an open source Python library built to empower developers to build applications and systems with self-contained Computer Vision capabilities. https://github.com/OlafenwaMoses/ImageAI

Muller-Budack, E., Pustu-Iren, K., & Ewerth, R. (2018). Geolocation estimation of photos using a hierarchical model and scene classification. In Proceedings of the European conference on computer vision (ECCV) (pp. 563-579)

Oliva, A., Torralba, A. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. International Journal of Computer Vision 42, 145–175 (2001).

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv:1804.02767.

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1908.10084

Saroj, A., & Pal, S. (2020). Use of social media in crisis management: A survey. International Journal of Disaster Risk Reduction, 48, 101584.

Shan, S., Zhao, F., Wei, Y., & Liu, M. (2019). Disaster management 2.0: A real-time disaster damage assessment model based on mobile social media data—A case study of Weibo (Chinese Twitter). Safety science, 115, 393-413.

Smeaton, A.F., Over, P. and Kraaij, W., 2009. "High-level feature detection from video in TRECVid: a 5-year retrospective of achievements", In Multimedia content analysis (pp. 1-24). Springer, Boston, MA.

Thomson W., Bhowmik N. and Breckon T. P., "Efficient and Compact Convolutional Neural Network Architectures for Non-temporal Real-time Fire Detection", 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), 2020, pp. 136-141

Toulouse, Tom, Lucile Rossi, Antoine Campana, Turgay Celik, and Moulay A. Akhloufi. "Computer vision for wildfire research: An evolving image dataset for processing and analysis." Fire Safety Journal 92 (2017): 188-194, doi: https://doi.org/10.1016/j.firesaf.2017.06.012. Available at: https://cfdb.univ-corse.fr/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30

Yung, Bosco. 2021. TensorFlow 2 implementation of the Yahoo Open-NSFW model. https://github.com/bhky/opennsfw2 (2022) [Original source: https://studycrumb.com/alphabetizer] [Original source: https://studycrumb.com/alphabetizer]