

H2020-ICT-2018-2 /ICT-28-2018-CSA
SOMA: Social Observatory for Disinformation and Social Media Analysis



D3.5 Data Intelligence Toolkit v2

| | |
|-----------------------------|--|
| Project Reference No | SOMA [825469] |
| Deliverable | D3.5 Data Intelligence toolkit v2 |
| Workpackage | WP3, Observatory Set-up and Operation |
| Type | Other |
| Dissemination Level | Public |
| Date | 30/04/2020 |
| Status | V0.1 |
| Editor(s) | Luca Tacchetti, LUISS Datalab, LUISS University |
| Contributor(s) | Emanuele Camarda, LUISS Datalab, LUISS University |
| Reviewer(s) | ATC |
| Document description | This deliverable D3.5 introduces the final version of the software implementation of the AI Driven Observatory |

Document Revision History

| Version | Date | Modifications Introduced | |
|---------|------------|--------------------------|------------------|
| | | Modification Reason | Modified by |
| v0.1 | 01/03/2021 | Concept and Structure | LUISS University |
| v0.2 | 15/03/2021 | First Draft | LUISS University |
| v0.3 | 26/04/2021 | Final Draft | LUISS University |
| V0.1 | 30/4/2021 | Final | LUISS University |

Executive Summary

This deliverable D3.5 introduces the final version of the software implementation of the AI Driven Observatory, that puts into practice the outcome of the work performed for Deliverables 3.2 and 3.3. In particular, the report provides an overview of DisInfoNet Toolbox’s current functionalities and features, architecture and user manuals. The toolbox is designed to help the SOMA community to understand the dynamics of (fake) news dissemination in social networks. DisInfoNet combines text mining and classification with graph analysis and visualization to offer a comprehensive and user-friendly suite that allows users to (i) track relevant news stories and reconstruct their prevalence over time and space; (ii) detect central debating communities and capture their distinctive polarization/narrative; (iii) identify influencers both globally and in specific “disinformation networks”. Software development has been conducted according to the Agile methodology, based on iterative development of users requirements and epics. The subsequent platform Architecture incorporates the logical structure of the previous functional analysis.

Table of Contents

| | |
|---|----|
| 1. Introduction | 6 |
| 2. High Level Design | 7 |
| 2.1. Disinfonet: a Definition | 7 |
| 2.2. Functionalities | 8 |
| 3. User Requirements | 10 |
| 3.1. Product Backlog, Epics, User Stories | 10 |
| 3.2. Disinfonet: Epics | 10 |
| Table 2. DisInfoNet Epics | 12 |
| Table 3. DisInfoNet User Stories | 17 |
| 4. High Level Architecture..... | 17 |
| 4.1. Code Structure..... | 18 |
| 5. Data Intelligence Toolkit v2 | 21 |
| 6. Installation | 30 |
| 7. User Manual..... | 31 |
| 8. Conclusions | 37 |

List of Figures

| | |
|--|----|
| Figure 1:DisInfoNet's main pipeline..... | 8 |
| Figure 2:High Level Architecture..... | 18 |
| Figure 3:Dataset Management Homepage..... | 21 |
| Figure 4:. Import new dataset..... | 22 |
| Figure 5:. Dataset Overview..... | 23 |
| Figure 6:Filter | 24 |
| Figure 7:. Create new filter | 25 |
| Figure 8:Classifier | 26 |
| Figure 9:Create New Classifier | 27 |
| Figure 10:Cluster selection..... | 28 |
| Figure 11:About page..... | 29 |
| Figure 12:Privacy and Policy page..... | 30 |
| Figure 13:Create new filter interface..... | 32 |
| Figure 14:Histogram Prevalence | 33 |
| Figure 15:Histogram Polarization | 34 |
| Figure 16:Time Distribution | 35 |
| Figure 17:Map | 36 |

List of Terms and Abbreviations

| Abbreviation | Definition |
|--------------|-----------------------------------|
| API | Application Programming Interface |
| CSV | Comma-Separated Values |
| JSON | JavaScript Object Notation |
| ORM | Object-Relational Mapping |
| DB | Database |
| SVG | Scalable Vector Graphics |
| JPEG | Joint Photographic Experts Group |
| PNG | Portable Network Graphics |

1. Introduction

This document intends to collect the analyses carried out to design the DisInfoNet prototype. From the first phase of collecting user requirements, we moved on to writing the Product Backlog, accompanied by Epics and User Stories.

Starting from the functional analysis conducted, a high level architecture was designed that describes the technical components of the prototype and the main functions. Finally, we moved on to the back end and front end development of the Data Intelligence toolkit v2 (only some of the functionalities have been implemented).

1.1 Structure of the Deliverable

The current report is structured as follows.

- Section 2 provides a complete overview of DisinfoNet high level design and main functionalities. The section describes the general purposes of the toolbox and the tools already implemented in order to reconstruct relevant news stories on social media over time and space as well as to identify central debating communities and specific “disinformation networks” and actors.
- Section 3 is devoted to the analysis and collection of user requirements for the development of the DisinfoNet platform, conceived according to a “scrum” approach, that accounts for product backlog, epics’ definition, prioritized user-stories.
- Following the functional analysis, Section 4 presents the high-level architecture of the platform, based on a classic three levels architecture composed by a database, a backend and a frontend. The same section presents the code-structure.
- The final section of the report illustrates the final version, v2, of the Data Intelligence Toolkit.

2. High Level Design

2.1. DisinfoNet: a Definition

DisInfoNet Toolbox aims at supporting users of the SOMA verification platform in understanding the dynamics of (fake) news dissemination in social media and tracking down the origin and the broadcasters of false information. It combines text mining and classification with graph analysis and visualization to offer a comprehensive and user-friendly suite that allows users to:

- (i) track relevant news stories and reconstruct their prevalence over time and space;
- (ii) detect central debating communities and capture their distinctive polarization/narrative;
- (iii) identify influencers both globally and in specific “disinformation networks”.

DisInfoNet is developed by Data Lab for web users, media experts, journalists and researchers to analyze Twitter discussions in a simple but rigorous way. More specifically, the toolbox help users to identify messages, news, comments and, in general, contents that are false, tendentious or propagandistic; it helps the members of the vast community around SOMA to understand the dynamics of dissemination of this content on social media, identifying users who have a particularly active role in this process and possibly reconstructing the news sources. These are the main features:

- a scraper for data collection from Twitter;
- a filter for the selection, within a set of documents, of texts that report or treat or comment on a news item of interest;
- a tool for extracting a graph - oriented to users or hashtags - from Twitter data;
- a toolbox for graph analysis, equipped with community detection tools, centrality and identification of hubs and influencers, classification of users based on their role in intra- and inter-community connections;
- a software to perform topic modeling on a corpus of documents;
- a classifier for understanding the polarization of a given message or user against a topic of interest.

2.2. Functionalities

DisInfoNet implements a pipeline designed to enable journalists and fact-checkers with no coding expertise assessing the prevalence of disinformation in social media data. This pipeline, depicted in Figure 1, consists of three main tools, described in the following sections. It can be fed with a CSV file or the (possibly GZ-compressed) JSON file returned by the Twitter search or streaming API. One of DisInfoNet's main features is the ability to extract and examine both keyword co-occurrence graphs and user interaction graphs induced by a specific set of themes of interest, thus providing valuable insights into the contents and the actors of the social debate around disinformation stories.

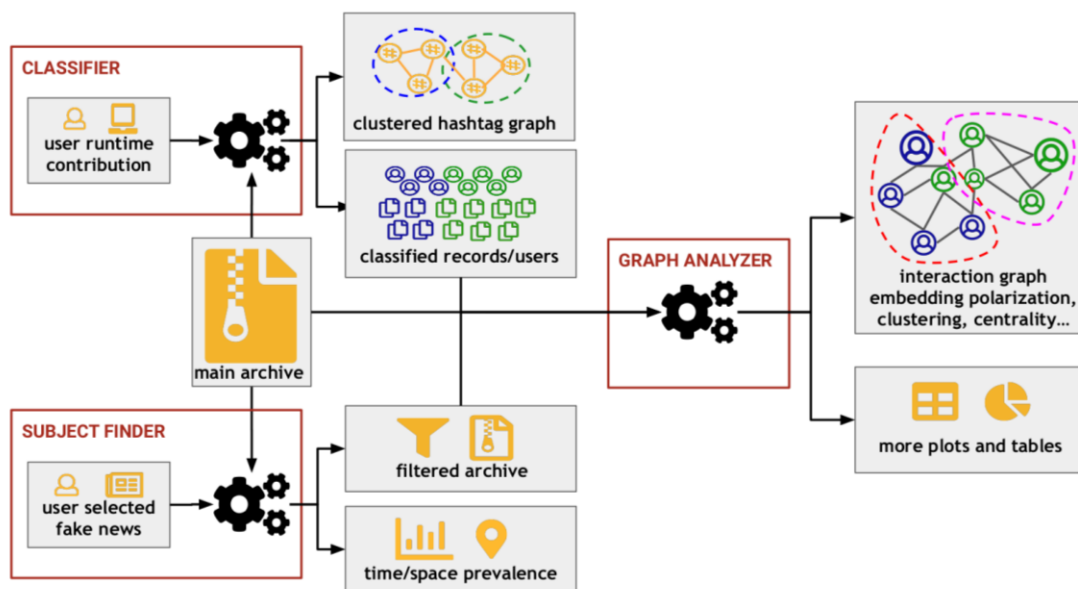


Figure 1: DisInfoNet's main pipeline

Data Collection: DisInfoNet provides a dashboard for collecting data from selected social media and news items. The current data collection tool is a tweepy-based Python scraper for Twitter data that accepts a list of keywords and returns a variable-length list of tweets containing at least one of such keywords.

Classifier: is based on a semi-automatic “self-training” process”, in which a list of hashtags associated with two (or more) classes of interest are used to automatically extract a training set. The classifier works according to a three-step process:

- It builds and cluster a **keyword co-occurrence graph**¹, that presents the user with an excerpt of the keywords associated with the obtained classes.
- It provides the **user** with the possibility to **prune** of central (high pagerank) yet generic and/or out-of-context keywords, detrimental to clustering.
- After performing **community detection** on the graph, the user is presented with an excerpt of these clusters for manual inspection and she/he can pick and label any of these communities as representative of a specific class.

As a result, the classifier can use as many as 10/100/100× more hashtags for classification than with any realistic fully manual approach, without sacrificing accuracy and possibly bringing to light previously unknown highly discriminative hashtags. When only two classes are used (eg. pro/anti, yes/no), the obtained classification can be extended to users by averaging over the classification of all tweets produced by a specific user.

The **Graph Analyzer** provides users with graph mining and visualization. It first extracts a directed user interaction graph, wherein two users (e.g., authors) are connected based on how often they interact (e.g., cite each other). It then computes a set of global and local metrics, including: distances, eccentricity, radius and diameter; clustering coefficient; degree and assortativity; PageRank, closeness and betweenness centrality. It also partitions the graph into communities, relying on the Louvain or Leading Eigenvector algorithms, and applies the Guimera`-Amaral cartography.

Temporal distribution by class: The figure provides the one-day rolling mean of each obtained class in the considered period of time, compared with the overall trend.

Spatial distribution by class: The figure provides the geographic distribution of tweets, based on the percentage of geotagged tweets.

Hashtag graph: The graph plots the hashtags, highlighting clusters and vertex sizes (using pagerank).

¹ The hashtag graph is extracted from the available dataset by connecting any two hashtags that co-occur at least once in the same tweet and weighing the edge by the total count of such co-occurrences.

3. User Requirements

The phase of analysis and collection of user requirements for the development of the **Disinfonet** platform has followed an approach known as “**scrum**”. Scrum is the most popular agile framework for software development, and it is characterized by iterative development, self-organized and empowered team, time boxed meeting. In detail, following the scrum methodology we draft the product backlog, define the epics, finalize the user stories.

3.1. Product Backlog, Epics, User Stories

In its simplest definition, the **Product Backlog** is a list of all the features that must be implemented in the project. In Scrum it plays the role of specification of the requirements. The items making up the Backlog are technical or user-centric in nature, written in the form of a user story, the analogue of use cases in a context of UML based analysis of requirements.

- A User Story defines a user's need with respect to an object or context of use. A hypothetical user story model can be the following: **As a [subject], I want|have to [action], so that [goal]**
 - *Subject*: the user story owner. It can be the user but it is advisable to specify it (eg "administrator", "logged in user", "unauthenticated visitor"). In this way the story is easier to understand and frame in the appropriate context.
 - *Action*: what the actor wants to do. If it is a mandatory action, the prefix “have to” have to be used, otherwise the prefix "I want".
 - *Goal*: what the actor wants to achieve by carrying out the action described by the story.

3.2. Disinfonet: Epics

Below is the detail of the epics identified for the design of the **Disinfonet prototype**. Epics are divided by area: **overview component**, **data collecting component**, **analysis component**, **visualization component**. The functional analysis shown below is the complete analysis for the design of the DisInfoNet prototype. However, in the first release of January 2020 (Data Intelligence toolkit v1), only some of the features designed are developed and available.

| 1 | Overview Component |
|-----|--|
| 1.1 | Upon first access, the user receives information relating to the platform and its main partners: Luiss Data Lab, Aletheia, Soma |
| 1.2 | The user can view the graphs of the latest searches |
| 1.3 | Upon login, the user can decide to perform analysis on existing datasets |
| 1.4 | The user upon login can create a new analysis |

| 2 | Data Collecting Component |
|-----|--|
| 2.1 | The user must have a twitter developer account with the relative access keys which must be shared on the platform on first access. The access keys linked to the user will be recorded in the db for future access |
| 2.2 | The user can create a new search by filling in the filter fields |
| 2.3 | The user can specify synonyms for each word entered in the query |
| 2.4 | The user can decide not to collect retweets, replies |
| 2.5 | The launched searches go back as far as possible with the possibility of inserting the start date and proceed forward automatically if not stopped |

| 3 | Analysis Component |
|-----|--|
| 3.1 | The search results can be downloaded in csv |
| 3.2 | General search results can be further filtered by one or more TAGs (exact words, hashtags or phrases) and each exact word, hashtag or phrase can be synonyms. The result can be downloaded in csv. |
| 3.3 | The results of the general search can be elaborated through the classifier, which proposes some hashtag clusters for the communities. The user can decide to keep the clusters and assign him a LABEL or skip the clusters that do not interest him. The clusters will instruct other data, training sets. |

| 4 | Visualization Component |
|------|--|
| 4.1 | The user can create geolocated maps of all the overall tweets collected |
| 4.2 | The user can decide to apply filters to the visualization - select words to give different color to the map |
| 4.3 | The user can create geolocated maps of the tweets already analyzed with the TAG filter if available |
| 4.4 | The user can create geolocated maps of the tweets already analyzed with the LABEL filter if available |
| 4.5 | The graphs can be downloaded in png and svg |
| 4.6 | The user can create a network chart for all the overall tweets collected |
| 4.7 | The user can decide to apply filters to the visualization - select words to give different color to the network graph |
| 4.8 | The user can create a network chart of the tweets already analyzed with the TAG filter if available |
| 4.9 | The user can create a network chart of the tweets already analyzed with the LABEL filter if available |
| 4.10 | Charts can be downloaded in svg and png |
| 4.11 | The user can create a time trend graph for all the overall tweets collected |
| 4.12 | The user can decide to apply filters to the visualization - select words to assign different color to the time trend graph |
| 4.13 | The user can create a time trend graph of the tweets already analyzed with the TAG filter if available |
| 4.14 | The user can create a time trend graph of the tweets already analyzed with the LABEL filter if available |

Table 1: DisInfoNet Epics

3.3. Disinfonet: User Stories

The Epics were broke out into prioritized user stories (high/low). The “status” column identify which user stories have already been implemented at the time of this deliverable (Data Intelligence Toolkit v1).

| ID | Area | User Stories | Priority | Status |
|----|-------------------------------|--|----------|--------|
| 1 | Overview Component | By accessing the Homepage of the DisInfoNet platform, the user finds a description with a brief summary of the prototype | HIGH | DONE |
| 2 | | By accessing the homepage of the DisInfoNet platform, the user finds the results of the last search of the Data Lab. The results can be browsed in interactive mode. | HIGH | DONE |
| 3 | | Through the DisInfoNet logo it is always possible to return to the homepage | HIGH | DONE |
| 4 | | By accessing the Homepage, the user can decide to start a new search by clicking on the "start a search" button, also present as an item in the "new search" menu. | LOW | |
| 5 | | By clicking on the button and / or on the menu item "start a search" the authentication window opens. | LOW | |
| 6 | | The user finds 3 menu items: "start a search", "About us" and "Login" (characterized by an icon) | LOW | |
| 7 | | By clicking on "about us" you will land on an explanatory text | HIGH | DONE |
| 8 | | By clicking on "Login" you will land on the authentication page | LOW | |
| 9 | | The SOMA logo is present on the platform (https://www.disinfoobservatory.org/) | HIGH | DONE |

| | | | | |
|----|----------------------------------|--|-----|--|
| 10 | Data Collecting Component | The user logs in with username (email) and password | LOW | |
| 11 | | If you do not have the credentials, you can request the credentials through a specific call to action | LOW | |
| 12 | | Password recovery must be possible | LOW | |
| 13 | | Entering the private area, the user can choose whether to analyze existing datasets or activate a new data collection | LOW | |
| 14 | | If the user chooses to start a new data collection, they are asked for the access keys linked to the Twitter Developer user they owned (https://developer.twitter.com/en/application/use-case) | LOW | |
| 15 | | When creating a new collection, the user must necessarily assign a title to it (with the addition of a possible description) | LOW | |
| 16 | | The user must fill in the fields relating to the search rules: these fields are the filters of the search that will be launched | LOW | |
| 17 | | The filters that the user can fill in are: All of these words, this exact phrase, any of these words, none of this words, this hashtag, written in (language), near this place, advanced rules (filter retweets - filter reply). People: to this account, mentioning accounts, from these accounts. | LOW | |
| 18 | | The user can also decide to specify synonyms for each word entered in the search query | LOW | |
| 19 | | The user can select the dates delimiting the search (minimum search start date-maximum end date). If the user does not select the minimum date, the search goes back in time as much as possible. | LOW | |
| 20 | | If the user does not stop the search, it proceeds forward in time | LOW | |

| | | | | |
|----|---|---|------|------|
| 21 | Analysis Component | Once the user believes he has completed a search, he can download the results via a csv file. The file contains all the results without distinction whatsoever. | LOW | |
| 22 | | The user may decide to filter the general search. In this case, before downloading the csv file can fill in specific fields such as: words, hashtags, or exact sentences. In this case the search can be filtered as if it were a pivot table. Once filtered, the specific file can be downloaded by the user. This filter is called TAG. | HIGH | DONE |
| 23 | | The user can decide to deepen the research carried out to identify one or more communities. | HIGH | DONE |
| 24 | | If the user wants to deepen the identification of the communities, he can click on the "Classify the dataset" button. | HIGH | DONE |
| 25 | | Once activated, the classifier proceeds to classify the dataset according to the precompiled algorithm | HIGH | DONE |
| 26 | | Once the processing is complete, the user can decide to: attach a label to the clusters obtained from the classification, skip the clusters that he deems not useful or unnecessary. This filter is called LABEL | HIGH | DONE |
| 27 | | The user can decide to use clusters as a training set basis for further data analysis | HIGH | DONE |
| 28 | Visualization Component | Once the analysis is completed, the user can decide to browse the results according to the different components and according to the research carried out | HIGH | DONE |
| 29 | | The user can view: maps, network graphs, time trends | HIGH | DONE |
| 30 | Visualization Component - maps | When accessing the homepage, the user finds some geo-localized maps with the datasets of the latest searches | HIGH | DONE |

| | | | | |
|----|--|---|------|------|
| 31 | | The user can create geolocated maps of all the overall tweets collected | HIGH | DONE |
| 32 | | The user can decide to apply filters to the visualization - select words to assign different colors to the map | LOW | |
| 33 | | The user can create geolocated maps of the tweets already analyzed with the TAG filter if available | HIGH | DONE |
| 34 | | The user can create geolocated maps of the tweets already analyzed with the LABEL filter if available | LOW | |
| 35 | Visualization component - network graph | The user, when accessing the homepage, finds some network graphs with the datasets of the latest searches | HIGH | DONE |
| 36 | | The user can create a network chart for all the overall tweets collected | HIGH | DONE |
| 37 | | The user can decide to apply filters to the visualization - select words to give different color to the network graph | LOW | |
| 38 | | The user can create a network chart of the tweets already analyzed with the TAG filter if available | HIGH | DONE |
| 39 | | The user can create a network chart of the tweets already analyzed with the LABEL filter if available | HIGH | DONE |
| 40 | Visualization component - highchart trend | The user, when accessing the homepage, finds some temporal trends with the datasets of the latest searches | HIGH | DONE |
| 41 | | The user can create a time trend graph for all the overall tweets collected | HIGH | |

| | | | | |
|----|--|--|------|------|
| | | | | DONE |
| 42 | | The user can decide to apply filters to the visualization - select words to assign different color to the time trend graph | LOW | |
| 43 | | The user can create a time trend graph of the tweets already analyzed with the TAG filter if available | HIGH | DONE |
| 44 | | The user can create a time trend graph of the tweets already analyzed with the LABEL filter if available | LOW | |

Table 2: DisInfoNet User Stories

4. High Level Architecture

The high-level architecture for the platform features different software components that incorporate the same logical subdivision proposed in the functional analysis phase.

The infrastructure of the tool is managed through a docker cluster where are defined the following components:

- **Nginx:** this container has the role of reverse proxy, manages all the communications between the systems and is the external access point to the system.
- **Flask:** this is the container where the main application is launched. The backend is written in python and uses flask as a framework for web application management. Flask works over a **uWSGI** web server that manages all the REST requests.

SqlAlchemy is used under flask to manage the ORM problem and interaction with mysql databases. The frontend's logic is written in javascript through the framework angularJs. Backend and frontend communicate through RestApi and exchange data in json format. **Marshmallow** is used for the serialization of data. A worker is responsible for loading datasets on clickhouse.

- **Classification:** it's a worker for classification task. Classificazione communicates with mariadb to check the status of extraction, and with clickhouse for getting collection data and saving classification results.

- **Clickhouse:** In this database are stored the collection data and some elaborated data for optimize analysis and visualization. It is a column oriented database, so is more efficient for queries that involve a big number of rows and few columns.
- **Mariadb:** in this database are stored information about user defined filters, dataset and classifications.

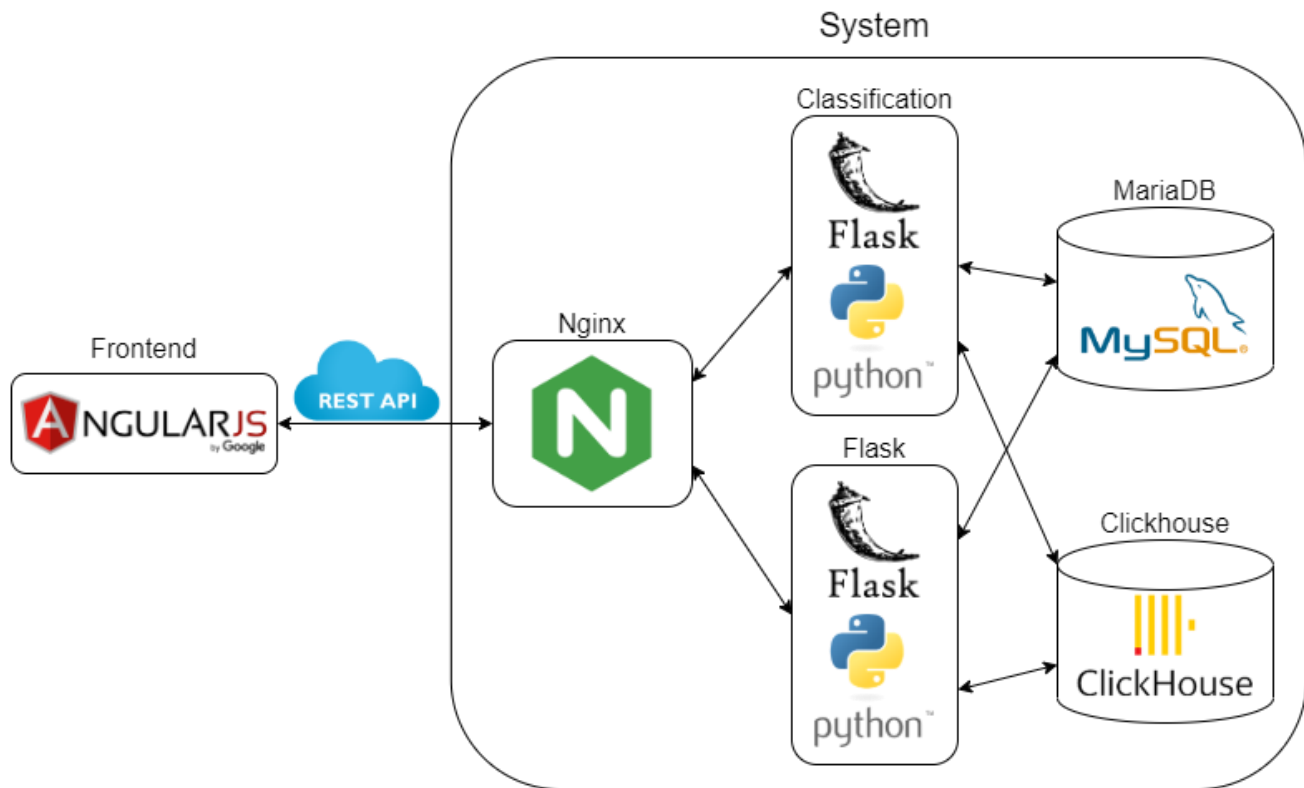


Figure 2: High Level Architecture

4.1. Code Structure

About the file structure of the application, we have the following main directories and files:

- **classification/:** contains all codes for the classification task;
- **clickhouse-config/:** contains all the configuration files for customize clickhouse;
- **flask/:** contains all the codes for the core of the application;
- **logs/:** automatically generated folder that contains all logs extracted by the container;
- **mariadb/:** automatically generated folder that contains a copy of all management, log and settings files of MySQL database;
- **nginx/:** contains all the configuration files for customize the web server nginx;
- **docker-compose.yml:** the configuration file for the docker cluster containers.

The core codes of the application are inside the directory flask, that is organize in the seguent way:

- application/: contains the logic applications;
- clickhouse/: contains all codes to initialize the clickhouse database and connects to it;
- graph_analysis/: contains all codes and algorithms for graph analyzation;
- text_analysis/: contains all codes and algorithms for text codification, manipulation and management;
- utils/: this folder contains all utility codes;
- app.ini: contains all the settings for the initialization of uwsgi application server;
- datasetManagement.py: the runnable code for the dataset upload management worker;
- db_json.json: data to initialize the mysql database with some example;
- Dockerfile: contains the image of the flask containers;
- manage.py: this script manages the mysql database basic operations (like create, delete and seed);
- requirements.txt: the list of python library necessary for the correct functioning of the application;
- run.py: the runnable starting file for the application;

The backend logic it's implemented inside the directory flask/application. In particular there are the seguent files/folders:

- classModels/: contains the definition of all class models for the clickhouse database;
- utils/: this folder contains all utility codes;
- __init__.py: initialize the application loading settings from setting.py and starts the application;
- views.py: contains the routing of backend's method;
- models.py: define the entity model of the application's classes;
- schemas.py: define the serialization of entity in the database;
- setting_template.py: template for the settings of the application, to fill with all the necessary credentials.

The front end it's implemented inside the directories flask/application/templates and flask/application/static:

- templates/ contains all main page html files; fronted follows the Single Page Application model;

- `static/partials` contains all the static html page injected inside the `index.html` through the routing's logic define with AngularJs;
- `static/js`: contains all javascript code with the logic of frontend application;
- `static/lib`: all support libraries for frontend;
- `static/img`: all static image;
- `static/css`: all css styles files.

About the logic of fronted, the application it's so divided:

- `app.js`: initialize the application and define the routing through the AngularJs' module `'ngRoute'`;
- `controllers.js`: implements the logic of all pages through its corresponding controllers;
- `directive.js`: implements some custom html directives;
- `filters.js`: implements some custom filters for displaying texts;
- `services.js`: implements some utilities function for javascript logic, such as custom ajax methods.

Backend exported the following methods:

- `.../dataLake/query`: Post method to obtain Twitter data from clickhouse with generic query;
- `.../getAllDataset`: Get method to obtain all available dataset;
- `.../saveFilter`: Post method to save a new filter in the database;
- `.../deleteFilter/<filter_id>`: Delete method to eliminate the filter with id *filter_id* from database;
- `.../checkClassifier/<extraction>`: Get method to check if there is at least one classifier for the specific extraction;
- `.../saveClassifier`: Post method to save a new classifier in the database;
- `.../getAllFilters/<extraction>`: Get method to obtain all filters for a specific extraction;
- `.../getClassifiers/<extraction>`: Get method to obtain the list of all classifiers linked to a specific extraction;
- `.../classifier/takeTopHashtag/<extraction>`: Get method to obtain the list of all hashtags for a specific extraction;
- `.../classifier/takeCommunities/<extraction>`: Post method to send the list of hashtags to delete and obtain the list of main communities for a specific extraction;
- `.../uploadFile`: Post method to upload a new dataset on the system;
- `.../greetings`: Testing method.

The software code of the application is available on Bitbucket, with authorization, at the following link:
<https://bitbucket.org/emanuelecamarda/disinfonet-docker>.

5. Data Intelligence Toolkit v2

The following section provides an overview of the main features already developed for the Data Intelligence Toolkit.

Dataset Management Homepage

A brief introduction to the DisInfoNet prototype is available on the dataset management homepage and in the main page of all single dataset overviews (Figure 3).

Under the description section, there is the *Dataset List* section where all the datasets currently loaded in the system can be viewed. For each dataset it is possible to view its name and current status. If the status is "*complete*", it is possible, by clicking on it, to access the analysis page for that specific dataset. In particular, there are three starting datasets already available for consultation: **Referendum_november**, **Amazonia** and **Qanon Eng**.

LUISS Data Lab [About](#) [Privacy Policy](#)

Disinfonet

Operated by the H2020 SOMA Project, the recently established Social Observatory for Disinformation and Social Media Analysis supports researchers, journalists and fact-checkers in their quest for quality information. At the core of the Observatory lies the DisInfoNet Toolbox, designed to help a wide spectrum of users understand the dynamics of (fake) news dissemination in social networks. DisInfoNet combines text mining and classification with graph analysis and visualization to offer a comprehensive and user-friendly suite that allows users to:

1. Track relevant news stories and reconstruct their prevalence over time and space;
2. Detect central debating communities and capture their distinctive polarization/narrative;
3. Identify influencers both globally and in specific "disinformation networks".

Dataset list

| Dataset Name | Status |
|--------------------------------------|----------|
| Referendum November | complete |
| amazonia_twitter | complete |
| 7244_Cambiamento climatico - hashtag | complete |

Disinfonet is part of the SOMA (Social Observatory for Disinformation and Social Media Analysis) project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825469.

Figure 3: Dataset Management Homepage

- At the bottom of the page, there is the *Import new dataset* section (Figure 4), through which you can insert a new dataset from your file system into DisInfoNet. The dataset must be a JSONEachRow in the [JSON Tweet Format](#), compressed through gz.

Dataset list

| Dataset Name | Status |
|---|-----------------|
| Referendum November | complete |
| amazonia_twitter | complete |
| 7244_Cambiamento climatico - hashtag | complete |
| Qanon Eng | complete |

Import new dataset

Dataset must be a JSONEachRow in [JSON Tweet Format](#), compressed through gz.

No File Selected



Disinonet is part of the SOMA (Social Observatory for Disinformation and Social Media Analysis) project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825469.

Figure 4.: *Import new dataset*

Dataset Overview

Each dataset has its own overview page (Figure 5) for the management of the analyzes, accessible by clicking on the name of the dataset, in the Dataset List of the Dataset Management Homepage. On this page you can view all the graphs relating to the analysis and manage filters and classifiers.

The top of the page contains a description of the SOMA project and the DisInfoNet platform. The first column on the left of the page contains all the views available for the dataset with their description. Below, in Chapter 8, there will be a detailed description of each view.

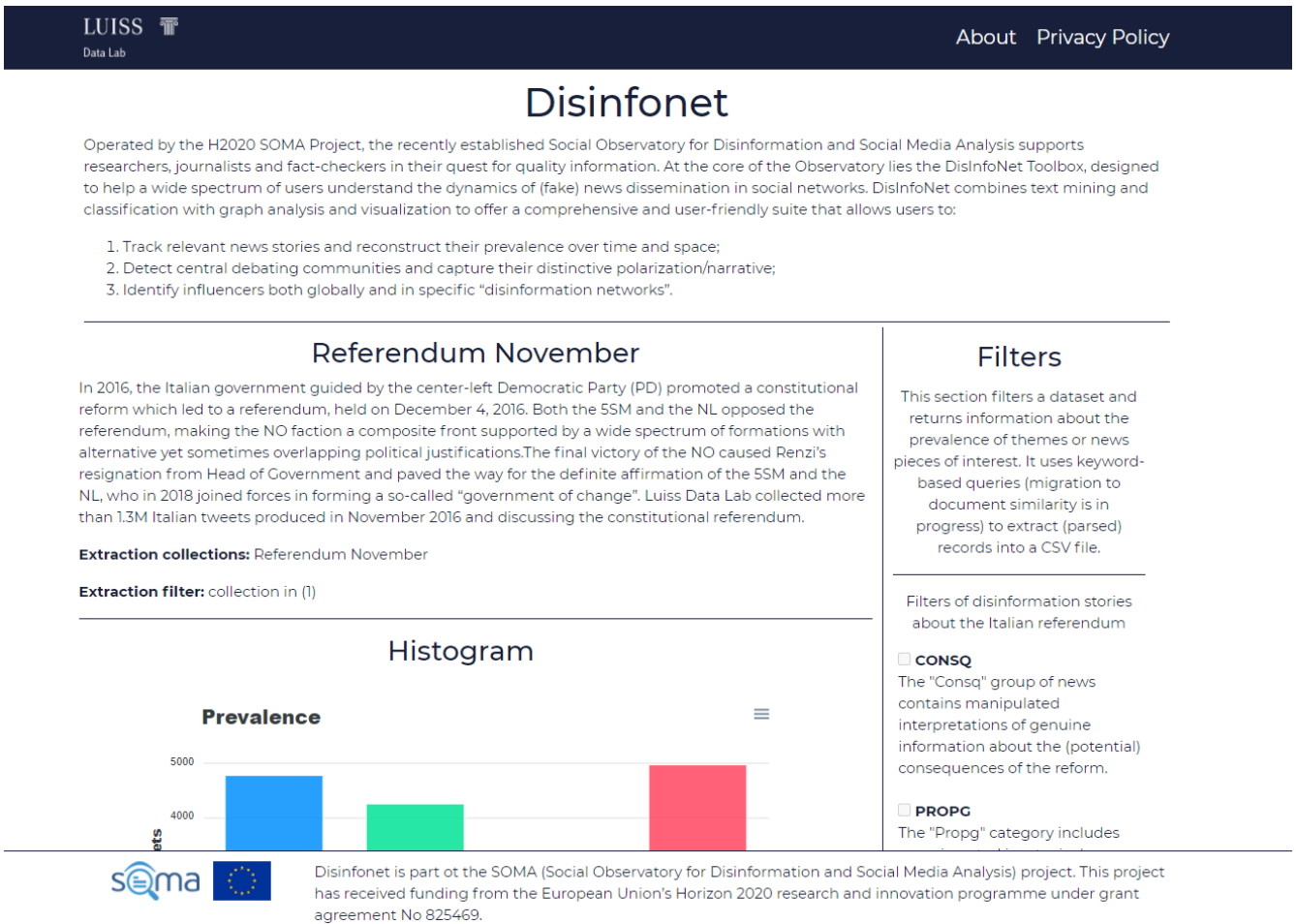


Figure 5.: Dataset Overview

Filter

An important feature of the DisInfoNet tool is the ability to create and manage filters, which group tweets linked to the same disinformation topic. The Filters section (Figure 6) is located to the right of the views in the Dataset Overview, and contains the list of existing filters, a button to redirect to the page for creating a new filter and a button to delete the selected filter.

Filters

This section filters a dataset and returns information about the prevalence of themes or news pieces of interest. It uses keyword-based queries (migration to document similarity is in progress) to extract (parsed) records into a CSV file.

Filters of disinformation stories about the Italian referendum

☐ **CONSQ**
The "Consq" group of news contains manipulated interpretations of genuine information about the (potential) consequences of the reform.

☐ **PROPG**
The "Propg" category includes news inserted in a typical populist frame, opposing people vs the elite.

☐ **QUOTE**
The "Quote" category includes entirely fabricated quotes of public figures endorsing one or the other faction or defaming voters of the other side.

☐ **FRAUD**
The "Fraud" category involves the integrity of the electoral process, gaining unauthorized access to voting machines and altering voting results.

Delete Filters

Create New Filters

Figure 6:Filter

The analyst/researcher can decide to create a new filter in the dataset through the interface below (Figure 7), specifying all necessary fields. For each word entered in the query, a special section will be automatically generated in which to insert any synonyms.

LUISS
Data Lab

AboutPrivacy Policy

Create new Filter

Tag (the main topic of the analysis, composed by one or more filters):

Tag is required

Description (a brief description of the main topic of the analysis):

Title (the filter of the analysis):

Title is required

Query (the exact word or group of words to filter the dataset):

Query is required
[Queries use the Google's logic research](#)

Add Query

Save Filter

Back Results

Figure 7.: Create new filter

Classifier

In the lower right column of the homepage there is the Classifier section (Figure 8). A Classifier is an object that allows you to assign a label to each tweet, based on a semi-automatic binary polarization process. The Classifier section will show all the available classifiers, with their current status and the associated labels and descriptions. The analyst/researcher can decide to create a new classification in the dataset through the appropriate button.

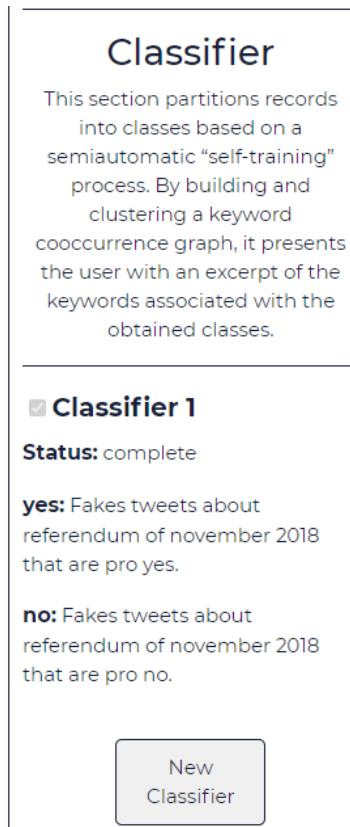


Figure 8:Classifier

In the Create new classifier section (Figure 9), the researcher/analyst can train a new classifier for the current dataset. A brief description of the entire classification process can be found at the top of the page.

In the first phase of the process the 30 top hashtags of the dataset will be displayed, and among these the user must select those to be deleted, not related to the search, to carry out a first cleaning of the training set.

Create new Classifier

Classifier is based on a semi-automatic "self-training" process", in which a list of hashtags associated with two classes of interest are used to automatically extract a training set. The classifier works according to a three-step process:

1. It builds and cluster a keyword co-occurrence graph, that presents the user with an excerpt of the keywords associated with the obtained classes.
2. It provides the user with the possibility to prune of central (high pagerank) yet generic and/or out-of-context keywords, detrimental to clustering.
3. After performing community detection on the graph, the user is presented with an excerpt of these clusters for manual inspection and she/he can pick and label any of these communities as representative of a specific class.

As a result, the classifier can use as many as 10/100/100× more hashtags for classification than with any realistic fully manual approach, without sacrificing accuracy and possibly bringing to light previously unknown highly discriminative hashtags.

[Back Results](#)

Top Hashtag

These are the top 30 hashtag. Now you can clean the dataset deleting any hashtag that not interest you. Click on "Finish" when you're done.

- | | | | |
|--|---|---------------------------------------|--|
| <input type="checkbox"/> #iovotono | <input type="checkbox"/> #referendum | <input type="checkbox"/> #renzi | <input type="checkbox"/> #referendumcostituzionale |
| <input type="checkbox"/> #bastaunsi | <input type="checkbox"/> #iodicono | <input type="checkbox"/> #4dicembre | <input type="checkbox"/> #iovotosi |
| <input type="checkbox"/> #no | <input type="checkbox"/> #riformacostituzionale | <input type="checkbox"/> #ottoemezzo | <input type="checkbox"/> #costituzione |
| <input type="checkbox"/> #pd | <input type="checkbox"/> #m5s | <input type="checkbox"/> #portaaporta | <input type="checkbox"/> #salvini |
| <input type="checkbox"/> #iostocoanmarino | <input type="checkbox"/> #si | <input type="checkbox"/> #noino | <input type="checkbox"/> #deluca |
| <input type="checkbox"/> #boschi | <input type="checkbox"/> #movimentonesti | <input type="checkbox"/> #bastaunsi | <input type="checkbox"/> #renziacasa |
| <input type="checkbox"/> #piazzapulita | <input type="checkbox"/> #lariachetira | <input type="checkbox"/> #si | <input type="checkbox"/> #nonrubo |
| <input checked="" type="checkbox"/> #trump | <input type="checkbox"/> #italia | | |

Finish



Disinfonet is part of the SOMA (Social Observatory for Disinformation and Social Media Analysis) project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825469.

Figure 9: Create New Classifier

In the second phase, the algorithm will calculate the largest clusters present (Figure 10); the researcher/analyst must select the 2 clusters of interest, adding a description. The 2 selected clusters will become the training set for the new classifier.

The classifier will now go into the "computing" state while DisInfoNet will perform all the necessary calculations; once finished it will go into the "complete" state and it will be possible to view the results.

Cluster selection

These are the top cluster of dataset. Choose 2 cluster for classification and specify labels for them. Click on "Finish" when you're done.

☐ ["referendum","no","costituzione","brexit","si"]

☒ ["referendumcostituzionale","bastaunsi","iovotesi","riformacostituzionale","italia"]

Yes

All tweets pro yes.

☒ ["iovotono","iodicono","4dicembre","m5s","renziacasa"]

No

All tweet pro no.

☐ ["renzi","bastaunsi","pd","riforma","si"]

☐ ["ottoemezzo","lariachetira","portaaporta","piazzapulita","tagadala7"]

☐ ["lavoro","jobsact","scuola","labuonascuola","buonascuola"]

☐ ["leopolda7","leopolda2016","firenze","leopolda","facciaafaccia"]

☐ ["beatricedimaio","renzie","brunetta","laprocuraindaga","iacoboni"]

☐ ["trendingtopic","idoneifantasma","abilitatitfa","odproroga2018","8000esiliatifaseb"]

☐ ["mentana","basta","stato","rai3","calderoli"]

Finish

Figure 10: Cluster selection

About

The About page (Figure 11), accessible from the header, provides useful information on the Luiss Data Lab and SOMA project.



About us

Disfonet is the main algorithmic engine of the SOMA verification platform, designed to support its users in understanding the dynamics of (fake) news dissemination in social media and tracking down the origin and the broadcasters of false information.

Disfonet Toolbox combines several features: data collection, data analysis and visualization for exploring conversations from social networks.

DisInfoNet Toolbox is developed by [LUISS Data Lab](#), a Research Center based at the Department of Political Science of LUISS Guido Carli. The center carries out basic and applied research in the field of Big Data and digital transformation, relying on a multidisciplinary perspective.

Disfonet is powered by [SOMA](#) "Social Observatory for Disinformation and Social Media Analysis", a H2020 Project aimed at supporting, coordinating and guiding the efforts of researchers, fact-checkers and journalists contrasting online and social disinformation.



Disfonet is part of the SOMA (Social Observatory for Disinformation and Social Media Analysis) project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825469.

Figure 11: About page

Policy and Privacy

The Policy and Privacy page (Figure 12), accessible from the header, provides useful information on the use and treatment of personal data and information of the user that uses the application.

Privacy Policy and Treatment of Personal Data

Premise

The information describes the characteristics of the treatments carried out by Luiss in relation to the navigation data of the website soma-disinforet.luiss.it.

What personal data do we collect?

Luiss Datalab, as Data Controller, collects the following data:

- user identification data, such as, for example, the IP addresses and domain names of the computer;
- data relating to the use of the website by the user, through the use of technical, analytical third-party and profiling cookies;
- data relating to user preferences, expressed while browsing the website;
- data provided voluntarily by the user - identification information - for access to certain services offered through the website, for which reference is made to the specific information present in the relevant reference sections.

As regards the use of cookies, these are made up of text files that are automatically generated in the user's computer after visiting some pages of the site. Some of these files (session cookies) are automatically removed when the browser is closed. Another type of cookies, on the other hand, is recorded and stored on the User's computer (for example, to automate the procedure for accessing the reserved area, the User can choose that his User-ID and password identification data are stored in one of these files). A further typology is made up of analytical cookies, used to collect information, in aggregate form, on the number of users and on the methods by which they visit the website. The aforementioned cookies can be installed directly by the site operator or from a different site that installs them through the first (so-called "third party" analytical cookies). Finally, the last type of cookies is made up of profiling cookies, used for the purpose of sending advertising messages, for example through personalized banners, in line with the preferences expressed by the User while surfing the net. If the User prefers not to receive cookies, he can prevent their transmission by the website by properly configuring his Internet browsing browser. In some cases, however, the use of some parts of the site may be conditioned by the storage of cookies on the User's computer.

For what purposes do we use your personal data?

Luiss processes personal data for the following purposes:

- manage the correct use of the website;
- evaluate, in statistical form, the use of the website by users;



Disinforet is part of the SOMA (Social Observatory for Disinformation and Social Media Analysis) project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825469.

Figure 12: Privacy and Policy page

6. Installation

To install the tool, follow the steps below:

- install docker and docker-compose
- up docker-compose ("docker-compose up -d --build" from project directory)
- (only first time) create and seed mysql from flask ("python manage.py create_db && python manage.py seed_db --seedfile db_items.json" from flask container)
- (only first time) create, initialize and seed clickhouse tables from flask ("create database dataintelligence" inside clickhouse-client cli from clickhouseclient container + "python clickhouse/clickhouse_connection.py && python clickhouse/gz_to_clickhouse.py ./data/ref-twitter_november_coll.json.gz 1" from flask container)
- start worker for classification ("nohup python classification.py &" from classification container)
- start worker for dataset management (nohup python datasetManagement.py & from flask container)

7. User Manual

To follow a description of the main features of the sites:

- **Import a new dataset:** it is possible to import a new dataset inside the system. To do it, in the main page click on the button “browse...” at the end of the page to select a dataset from your own filesystem; the dataset must be a JSONEachRow in the [JSON Tweet Format](#), compressed through gzip.

After choosing a file, click on the download button and wait until the dataset is transferred to the server. The new dataset will appear in the list of dataset with the status *loading*.

After a few minutes (depends on the dataset dimensions), the status of the dataset will go to *complete* and it will be ready to perform a new analysis on it.

- **View analysis on a dataset:** to see the analysis carried out on a dataset, search on the main page the dataset of interest and click on it (if its state is *complete*).
- **Create a new filter for a dataset:** a Filter is a set of queries that is used to isolate tweets about a disinformation topic. Each query is a set of words connected through a logic of AND/OR/NOT, that represent a fake news.

To create a new filter, click the button “New Filters” in the section *Filters* of the results page of a dataset. Compile the following form with this fields:

- *tag*: an identifier for the new filter;
- *description*: a short description for the new filter;
- *title*: a short description for the following query;
- *query*: AND/OR/NOT logic query; the query follows [Google’s logic research](#) (e.g. space between two words is meaning AND, etc..)

For each word in the query, it is possible to add a list of synonyms, separate by a comma, that will be used during the research.

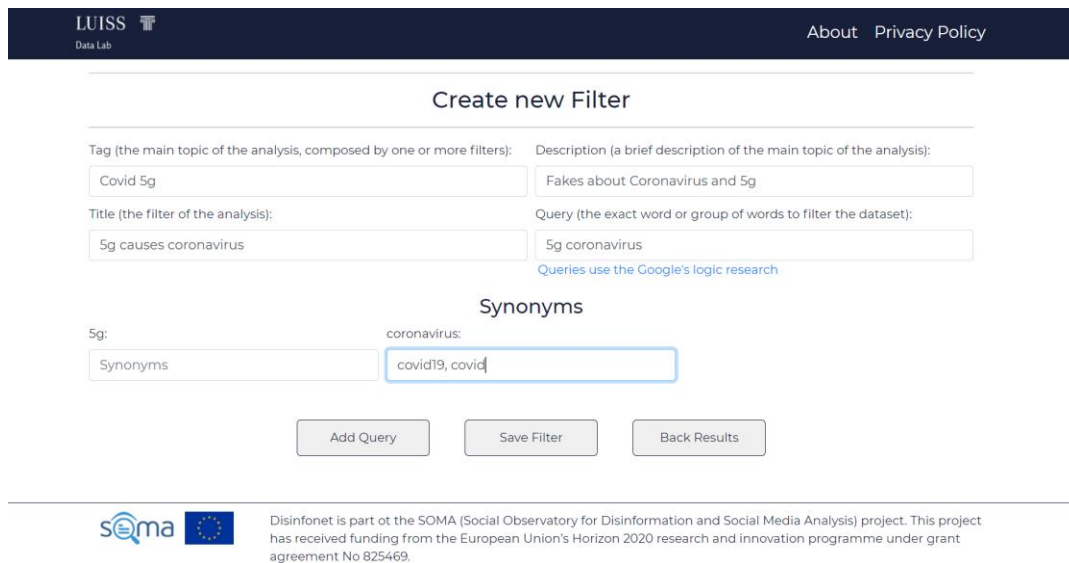


Figure 13: Create new filter interface

Through the button “Add Query”, is possible to add a new title and query for the current filter. The button “Save Filter” stores the just created filter, with all associated queries, on db. With the button “Back Result” you can go back to result without saving the new filter.

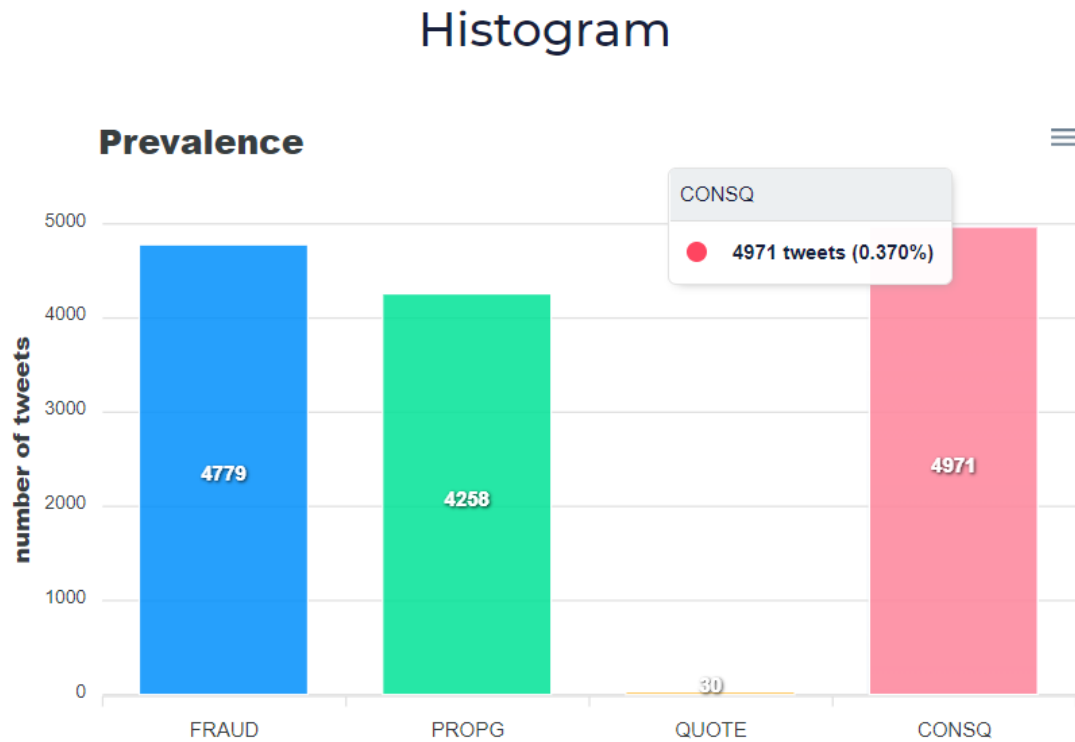
- **Create a classifier for a dataset:** you can create a new classifier through the button “New Classifier” in the section *Classifier* of the results page of a dataset.

The classification process is composed by more phases:

1. The hashtag graph is constructed and the pageranks are computed to order the nodes; Now it is possible to select and delete, among the 30 top hashtags, the misleading ones to clean the dataset.
2. The hashtag graph is rebuilt without the deleting ones selected in the previous phase, and a clustering algorithm is applied. You can select 2 clusters from those computed, to polarize all tweets between this. For each cluster you must specify a label and a description.
3. Now the training/prediction process can start. The new defined classifier can be seen in the *Classifier* section on the result page, with state *computing*. After that the training/predict process is terminated (it can last several hours depending on the size of the dataset), the classifier passes in the state *complete* and the related analysis can be viewed.

All charts are interactive and show the following analysis:

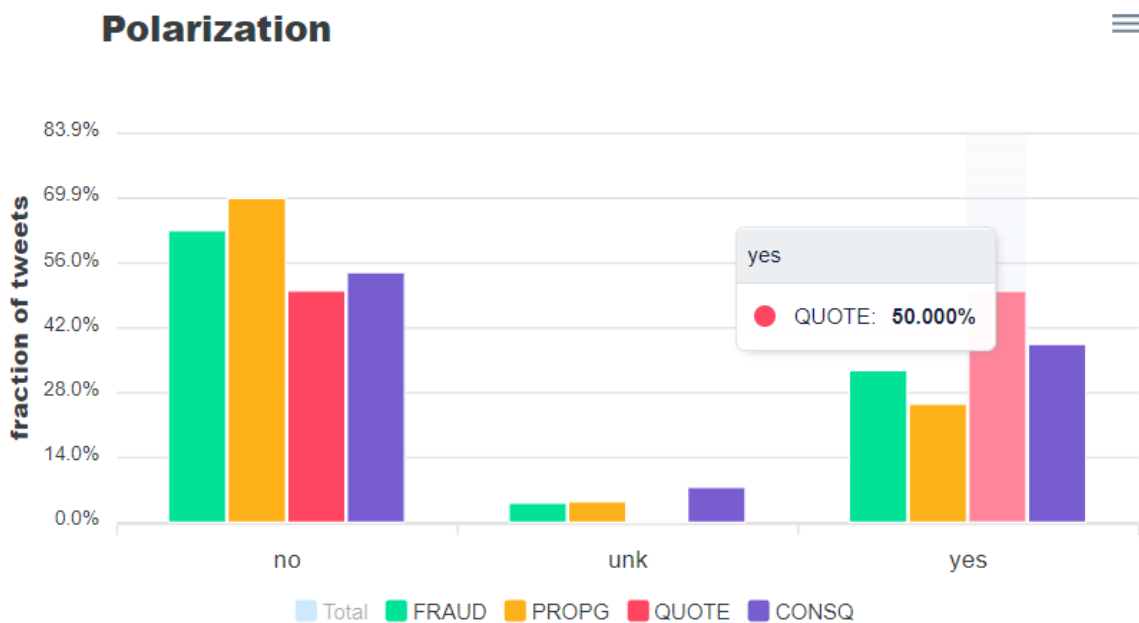
- **Histogram Prevalence:** An interactive histogram that highlights the prevalence of each group of filtered tweets (Figure 14). For each filter it is possible to view the quantity and percentage of the total. Using the drop-down menu at the top right of the graph, you can download a screenshot of the graph in .svg or .png format, or the .csv of the data.



The figure provides the prevalence of all available filters in the dataset.

Figure 14:Histogram Prevalence

- Histogram Polarization:** an interactive histogram that highlights the polarization of each group of filtered tweets with respect to the selected classifier (Figure 15). For each group of filtered tweets, it is possible to see the percentage subdivision derived from the classification, compared to the total number for that filter. Through the legend below the graph, it is possible to select/deselect a group of filtered tweets to highlight/hide them in the graph. Using the drop-down menu at the top right of the graph, you can download a screenshot of the graph in .svg or .png format, or the .csv of the data.



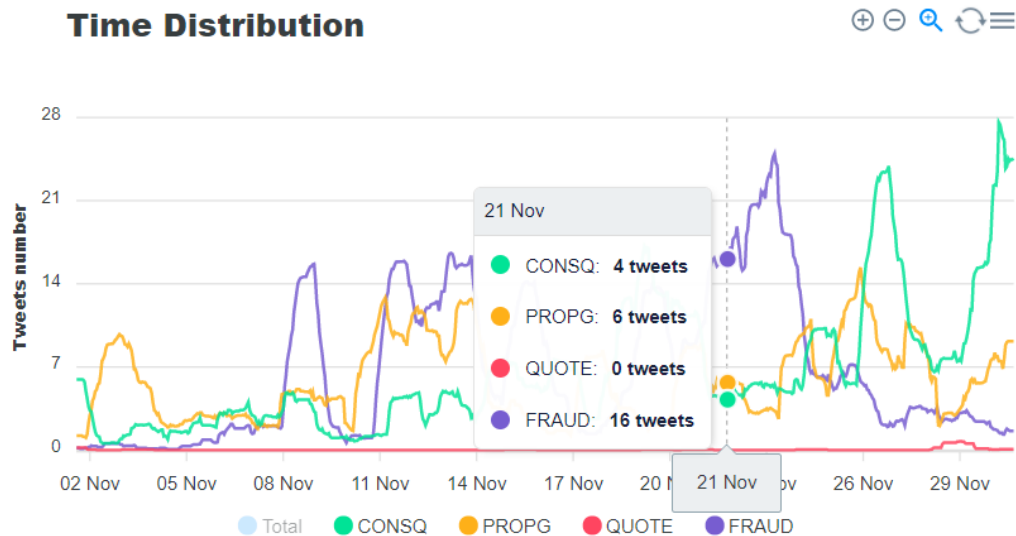
The figure provides the polarization of tweets for each available filter.

Figure 15:Histogram Polarization

- Time Distribution:** an interactive graphic that shows the time distribution of each group of tweets by filter created (Figure 16). By placing the mouse on a point on the graph, the number of tweets made that day will be displayed for each filter. It is possible to zoom the figure by selecting a time window inside the graph, or by using the two buttons zoom in and zoom out in the menu bar at the top right. To reset the zoom just select the appropriate button in the menu bar. Through the legend below the graph, you can select/deselect a group of filtered tweets to highlight/hide them in the graph. Using the drop-down menu, at

the top right of the graph, you can download a screenshot of the graph in .svg or .png format.

Time Distribution



The figure provides the one-day rolling mean of each obtained class in the considered period of time, compared with the overall trend.

Figure 16: Time Distribution

- Map:** in the map section (Figure 17) a navigable map is shown which shows the geographical distribution, where possible, of the filtered tweets. Through the menu at the top right, it is possible to select a specific filter to be displayed. It is possible to zoom the map by scrolling with the mouse or by using the + and - buttons located at the top left of the graph. By clicking on a specific tweet shown on the map, the user who created it and the text of the tweet will be shown. It is also possible to download a screenshot of the map using the button with the download symbol located at the top left.

Map



The figure provides the geographic distribution of tweets, based on the percentage of geotagged tweets.

Figure 17:Map

8. Conclusions

The goal of this deliverable is to describe the main features of the Disinfonet tool. Starting from what has been described in the document D3.3: Data Intelligence toolkit description the activity has seen the study of algorithms and architectural models to be implemented in the definition of the tool. Through the use of the Scrum framework, functional requirements, platform navigation flows, user experience and user interface, logical architecture have been defined. Through the use of example datasets, the first work has been realized, which represents the basis to start testing the prototype.

Based on these results, development continued with the following objectives:

- possible correction of errors in the development code;
- functionality test;
- bug fixes;
- graphic update of the platform;
- improvement of UX / UI components;
- testing of new input data sets;
- tests with external users;

Following the results obtained, a new version of the software update was released, capable of meeting the functional requirements and guaranteeing high operational performance of the Disinfonet toolkit.