Situation-Aware Linked heTerogeneous Enriched Data

D2.3: Report on Situation-aware data enrichment

Work package	WP 2
Task	Task 2.3
Due date	31/08/2023
Submission date	31/08/2023
Deliverable lead	Kybeidos
Version	1
Authors	Víctor González (UC), Laura Martín (UC), Jorge Lanza (UC), Juan Ramón Santana (UC), Pablo Sotres (UC), Maren Dietzel (Kybeidos), Gürkan Solmaz (NEC), Benjamin Hebgen (NEC), Ricardo Rodríguez (Amper), Amir Reza Jafari Tehrani (IMT)
Reviewers	Anja Summa (Kybeidos), Luis Sánchez (UC)

Abstract	This document, developed by the SALTED project, represents the D2.3 deliverable of the situation-aware data enrichment. The focus of this document is the definition of the architecture implementation and deployment, regarding the enrichment of data that has been carried out throughout the project. The close link to the applications of activity 4 results in a short description of applications as well. Furthermore, D2.3 includes an update on the architecture implementation and deployment reported in D1, D2.1 and D2.2.

This project is co-financed by the Connecting Europe Facility of the European Union under the Action Number 2020-EU-IA-0274.

This project is co-financed by the Connecting Europe Facility of the European Union under the Action Number 2020-EU-IA-0274.

Keywords	Data Modelling, Data Linking, Data Enrichment, Architecture, Security, Implementation, Applications

Table of Contents

Table of Figures

1 INTRODUCTION

1.1 SCOPE OF DOCUMENT

The main goal of the SALTED project is to add value to existing data by enriching them, and subsequently publish the enriched data in the European Data Portal (EDP) using the NGSI-LD information model. Furthermore, the project shows possibilities for applications using those homogenous, linked and enriched datasets using examples from the smart city and smart agriculture domain. This document D2.3 focuses on the implementation and current deployment status of the SALTED architecture defined in D2.1 [1]. Focus hereby is the implementation of the Data Enrichment Toolchain (DET), especially the components related to data enrichment. The document gives also updates regarding the last report D2.2 [2], which is mainly concerned with the DET components related to data linking.

1.2 TARGET AUDIENCE

This document is mainly intended for internal use, although it is publicly available in order to raise awareness of the SALTED project and its relation with the FIWARE ecosystem. Thus, the targeted audience is, besides the SALTED technical team including all partners involved in the delivery of work packages 1, 2 ,3 and 4, the scientific and technical community that are developing novel techniques on data management for extracting the potential value of these data according to the Findability, Accessibility, Interoperability, and Reusability (FAIR) principles. It especially serves as reference for the developers of the situation-aware applications in work package 4, since the enriching of datasets is often targeted towards a specific application. By extension, any application developer consuming data from a SALTED DET implementation should also find the document highly relevant. The document provides a thorough review of the key functionalities that the SALTED architecture and DET currently support, and gives first insights into the enriching steps that lay the basis for the applications developed within the project.

1.3 STRUCTURE OF THE DOCUMENT

Within the first substantive chapter, Section 2, the updates regarding the architecture, security and orchestration are described. Section 3 includes all updates regarding the implementation and deployment of the DET Injection Chains and the DET Linkers, respectively. Section 4 focusses on the new developments concerning the DET Injection Chains, especially the DET Enrichers. Section 5 gives a short summary on the development status of the project.

2 UPDATE: SECURITY AND ORCHESTRATION

2.1 SECURITY

In SALTED, we use a security architecture that was already defined in D2.2 [2]. It requires any party that wishes to access the Federator Broker and the Control Broker to authenticate against our KeyCloak instance, obtain a JSON Web Token, and use that token as proof of identity. This has the benefit of allowing us to have a tight control over any kind of authorisation, since there are sensitive operations that only specific users should be allowed to perform, while others should be open to the public. The downside of this approach is that it imposes some additional burden in terms of usability for external users since they are required to make an extra effort in order to simply enter the SALTED development environment.

In order to overcome this entry barrier, we have developed a Python package that greatly eases the communication with the KeyCloak instance and allows simple authentication and authorisation even with no knowledge of the underlying mechanisms. We have called this package *TokenHandler*.

Token Handler

The *TokenHandler* package is publicly available in the SALTED GitHub repository[1]. It can be easily installed (e.g. using pip) for Python 3. All it requires is an OAuth 2.0 token endpoint, a client ID, and a client secret as credentials. Internally, the package takes care of refreshing and updating the token when necessary, so users only need to retrieve it once with the *get_token* function.

The package is aimed at DET components and applications that require any kind of access to the Federator Broker, although it may also be used to handle communications with the Control Broker. However, the latter case is further covered in another Python package developed by the consortium that will be explained in the next section.

For more information refer to the README.md file located in the root of the repository, which provides all the necessary information and an example of use.

2.2 ORCHESTRATION

The Control Plane designed for SALTED, covered in depth in D2.2 [2], relies on a Control Broker implemented by an MQTT broker. DET components and applications have the possibility to provide a configuration interface, through which users or other applications can send MQTT messages to them. However, communication with the MQTT broker is not a trivial task, since it also requires authenticating against the SALTED KeyCloak instance. In order to address this complexity, the SALTED team has designed and implemented a Python package that greatly facilitates the authentication and communication with the Control Plane, the so-called *Control-Loop Handler*.

[1] https://github.com/SALTED-Project/TokenHandler

Control-Loop Handler

The *ControlLoopHandler* package is publicly available in the SALTED GitHub repository. We provide two distributions through the branches of the repository[2]: a downloadable script and a pip-installable version. Both implement the same functionality. The goal of the package is to ease the development of DET components and applications that use the control loop mechanism. This not only helps partners who are less familiar with these technologies, but also potential users that may wish to implement their own components and applications.

We recommend having a look at the README.md file located in the root of the repository, since it provides all the necessary information and examples of use, both from the perspective of the configurable component and the requesting party. We are currently using the Control Loop Handler in several components of our production deployment, such as the IoT Data Injection Chain (chapter 3.2) or the Geolocation Data Linker [2].

Local Partner Orchestration

MQTT is used for the orchestration of services within the SALTED infrastructure. The Control Broker is the central instance of the control plane, where services can be connected if they need to communicate platform-wide (especially for the secure JWT-secured communication between applications and pipelines). Apart from that, the orchestration of pipelines is partner-specific and dependent e. g. on the use case and the number of services integrated in the respective data enrichment toolchain. The following section gives an exemplary approach within the SALTED project, targeting the setup for the *Agenda Analytics* use case:

The use case consists of an injection toolchain that handles the automatic detection and integration of organization entities into the Scorpio Broker (sources: Open Street Map, Wikipedia), and an enrichment toolchain that crawls the web with different approaches, dependent on the agenda of interest, and generates matching scores based on that agenda (an agenda could be the Sustainable Development Goals but also any user defined set of targets). The two pipelines are formed by a series of services, namely *DiscoverAndStore, Mapping, Publish, Crawling,* and *AgendaMatching*. Those services are written in Python and use the FastAPI framework to allow for standalone usage. Additionally, they are all connected to a local MQTT Broker to enable automatic triggering (since this use case doesn't allow, as of now, triggering from the application side, no connection to the central Control broker is needed). The described setup is the backbone on which the *MQTT Trigger Service* is based, which allows for the flexible triggering of whole pipelines given the desired set of parameters for all services. Figure 1 and Figure 2 show the provided endpoints and an exemplary MQTT payload, which will be shortly explained in the following.

[2] https://github.com/SALTED-Project/ControlLoopHandler

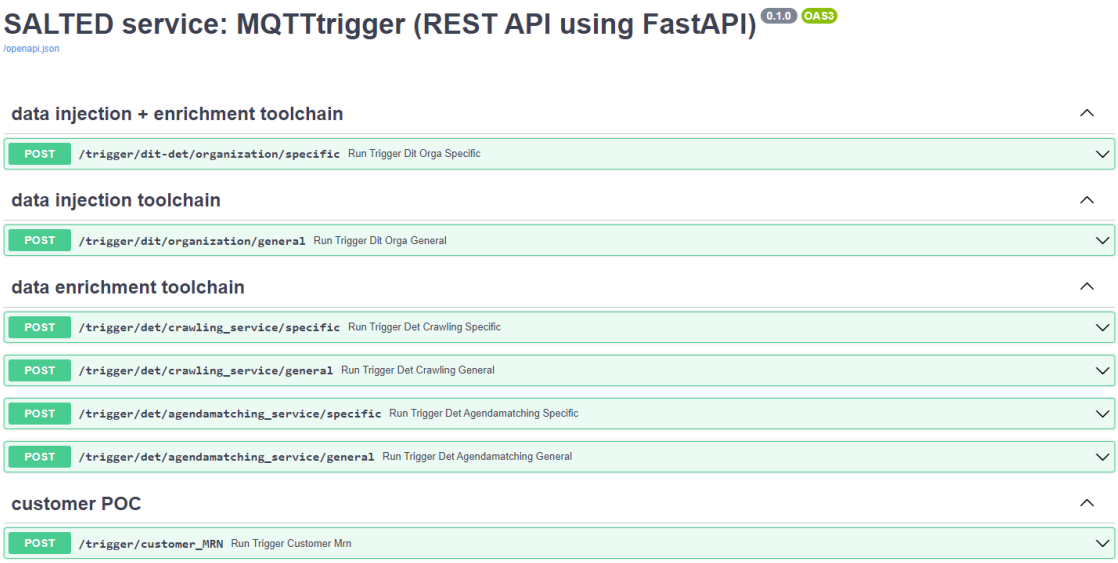


Figure 1. Agenda Analytics MQTT Trigger Service.

```
{
      "parameters": {
          "publish": {},
          "crawling": {
            "approach": "report",
            "custom_depth": "1",
            "actuality_days": "7",
            "keywords": "Nachhaltigkeitsbericht filetype:pdf",
            "target_agenda_entity_id": "urn:ngsi-ld:DistributionDCAT-AP:9bd03954-fa71-4fd7-a014-77b79b6534a0"
          },
          "agendamatching": {
            "agenda_entity_id": "urn:ngsi-ld:DistributionDCAT-AP:9bd03954-fa71-4fd7-a014-77b79b6534a0"
          }
        },
      "data": {<entitytype Organization>}
}
```

Figure 2. Agenda Analytics Exemplary MQTT Message.

As explained, the *MQTT Trigger Service* allows to start pipelines to inject data into the Scorpio Broker and enrich the data, so the end application can use an ever-growing data basis. As of now, it is e.g. possible to start the injection and enrichment process for one specific NGSI-LD entity of the type Organization. This is aimed to satisfy customer-specific demands towards Agenda Analytics (specific Organizations should be analysed): here all information needed for creating an NGSI-LD Organization must be supplied. Additionally, it is possible to start the general injection process, which tries to find and integrate all updates within the used data sources: here the organizations are automatically discovered and mapped to NGSI-LD. The same is possible for the enrichment process, which allows for automatic crawling or matching of all organizations represented within the broker or selected ones.

Within each trigger request the user is asked to provide the parameters that should be used by each service. Figure 3 shows an example for the injection and enrichment of one specific entity.

```
{
      "parameters": {
          "publish": {},
          "crawling": {
            "approach": "report",
            "custom_depth": "1",
            "actuality_days": "7",
            "keywords": "Nachhaltigkeitsbericht filetype:pdf",
            "target_agenda_entity_id": "urn:ngsi-ld:DistributionDCAT-AP:9bd03954-fa71-4fd7-a014-77b79b6534a0"
          },
          "agendamatching": {
            "agenda_entity_id": "urn:ngsi-ld:DistributionDCAT-AP:9bd03954-fa71-4fd7-a014-77b79b6534a0"
          }
        },
      "data": {<entitytype Organization>}
}
```

Figure 3. An example for the injection and enrichment of one specific entity.

3 UPDATE: DET INJECTION CHAINS

3.1 DATA MODELS

Due to the continuous integration of new data sources into the SALTED project, data models are continuously evaluated since they are used to map data into the NGSI-LD information model. The following part will give a short overview about additional NGSI-LD data models leveraged by SALTED since the D2.2, D2.1 and D1 reports [2] [1] [3].

3.1.1 DataService & DataServiceRun

When dealing with data, the topic of **Data Lineage**, i.e. how the data was created and modified, is crucial. With SALTED services being in active development, the need for data models came up that can represent a data service (e.g. Crawling Service) and a specific "run" of this service (e.g. Crawling that took place on the 31.05.23 at 10:00). The integration of this information into the broker allows applications to leverage this metadata to provide users full oversight regarding the sources of information that they consume. This ensures that SALTED is a trustworthy source of high-value data.

We used the DCAT-AP standard for this, which is already partly integrated into Smart Data Models. In particular the *DataService* (formerly *DataServiceDCAT-AP*) was extended by us, together with the warning that our added attributes do not belong to the official standard yet. This was possible, since the approach of the Smart Data Model initiative is that standards should nimbly adapt to reality and commonly used data models should prevail. Therefore we were able to actively contribute to this process by our use cases that try to integrate this standard into practice, instead of adding our own additional data model that may be too specific for others to use. The corresponding data model *DataServiceRun* was fully contributed by us, but also under the DCAT-AP umbrella repository[3,4].

3.1.2 Distribution

Similar to the metadata data models referenced in the previous section, this already implemented data model was chosen to model the used agendas within the Agenda Analytics pipelines. Persisting the agendas in their actually used version was the last step in giving full transparency to any data consumer. Now the full data basis used to provide the application to the customer is accessible via a single access point, the SALTED DET.

It is possible that the continuous exploration and evaluation of data models shows that, e. g. an integration of the Dublin Core, as "de facto" standard for metadata[5], has more advantages in this case, but this decision is still ongoing. The overall goal remains: to have a data model, which can be used to model artefacts, like documents and objects, as NGSI-LD entities. Error:

[3] https://github.com/smart-data-models/dataModel.DCAT-AP/tree/master/DataService

[4] https://github.com/smart-data-models/dataModel.DCAT-AP/tree/master/DataServiceRun

[5] https://www.dublincore.org/specifications/dublin-core/

Reference source not found shows an example integration as it is currently used, building upon the existing *Distribution*[6] Smart Data Model (formerly *DistributionDCAT-AP*).


```
    "id": "urn:ngsi-ld:DistributionDCAT-AP:f7c6cdd0-3b55-412d-a3d9-0c0fef06fdfd",
  "type": "DistributionDCAT-AP",
  "description": {
      "type": "Property",
      "value": "This entity holds information about one specifc representation of an agenda."
  },
  "downloadURL": {
      "type": "Property",
      "value": [
      "http://10.1.1.216:8006/files/a5b1c1d9-4e81-4a10-9b07-cc22553cac17",
      "http://10.1.1.216:8006/files/8b0f909c-4ea6-4966-b6f6-0cdfd2ab2582",
      "http://10.1.1.216:8006/files/c1c89a53-9475-45f2-b648-4ddce2ef5c39",
      "http://10.1.1.216:8006/files/a38d7ddb-e883-4ec4-9ab1-6c02684080b2",
      "http://10.1.1.216:8006/files/d726a448-8abb-43fd-9de5-cbd0c5a0b2b3",
      "http://10.1.1.216:8006/files/805cd6cc-3adf-40ee-be79-001090599bc5",
      "http://10.1.1.216:8006/files/dde6003f-c0ed-4042-81e3-f4cae2449179",
      "http://10.1.1.216:8006/files/09d33ef4-f3d3-4eb7-a2c8-b7a65876c99e",
      "http://10.1.1.216:8006/files/a7d81c19-3565-4a0c-81f7-f955a8395c80",
      "http://10.1.1.216:8006/files/cf9c47ff-831f-4dc8-a10c-59e53a5876ec",
      "http://10.1.1.216:8006/files/8b74190b-b8a2-4e58-941f-117cb4da4a69",
      "http://10.1.1.216:8006/files/6d9f0e48-6f4f-4188-84d8-ac1cb761e8ef"
      ]
  },
  "format": {
      "type": "Property",
      "value": " text"
  },
  "dateCreated": {
      "type": "Property",
      "value": "2023-07-07 14:06:55.147319"
  },
  "name": {
      "type": "Property",
      "value": "Environmental Social Governance (de)"
  },
  "@context": [
      "https://raw.githubusercontent.com/smart-data-models/dataModel.DCAT-AP/master/context.jsonld",
      "https://smartdatamodels.org/context.jsonld"
  ]
}
```


Figure 4. Example Distribution for agendas (Agenda Analytics).

[6] https://github.com/smart-data-models/dataModel.DCAT-AP/tree/e3408b27ffb1872843a40c2b0e26ccfdd9ec4d06/Distribution

3.2 IoT Data Injection Chain

3.2.1 Overview

The IoT Data Injection Chain (IoTDIC) was thoroughly explained in the D2.2 [2] and its core functionality remains unchanged. However, for the sake of completeness and to make this document the final compilation and description of the SALTED DET, we consider that it is important to make a comprehensive list of all of these data sources. The list will also be classified by the nature of the data sources and include the Smart Data Models to which the data from each of the sources will be subsequently mapped.

Batch Datasets

- Barcelona Open Data
  - AirQualityObserved
  - BikeHireDockingStation
  - TrafficFlowObserved
- Bilbao Open Data
  - TrafficFlowObserved
- Murcia Open Data
  - AirQualityObserved
- Oslo Open Data
  - AirQualityObserved
  - TrafficFlowObserved
- Santander Open Data
  - BikeHireDockingStation
  - FleetVehicleStatus
  - ParkingSpot
  - SoundPressureLevel
  - Temperature
- Santander TTN
  - ParkingSpot
  - TrafficFlowObserved
- Valencia Open Data
  - TrafficFlowObserved
- Vitoria Open Data
  - TrafficFlowObserved
- Vizcaya Open Data
  - AirQualityObserved
- Madrid Open Data
  - TrafficFlowObserved
  - AirQualityObserved

Data Streams

- SmartSantander
  - AirQualityObserved
  - BatteryStatus
  - ElectroMagneticObserved

Additionally, the components of the IoTDIC (collector, mapper and curator) are now deployed inside Docker containers for the various advantages they provide.

3.2.2 Update Madrid Air Pollution Data

IoT Air Pollution Madrid Data

In our thorough investigation of Madrid's air pollution data, our primary aim was to enhance the dataset and usefulness. This process involved the conversion of data into the NGSI-LD

format, and also enrich the dataset which is depicted in the illustrative figure 5 below. The initial dataset, gathered from various air pollutant stations, encompassed a concentration of pollutants, such as CO, SO2, PM10, NO2, and NO. The process followed within the Injection Chain encompassed the following steps:

Handling Missing Values Through Interpolation

To ensure the integrity of the data, we took a careful approach, closely examining the dataset to spot any missing values. Through the application of interpolation techniques, we effectively bridged these gaps, estimating missing pollutant concentrations based on previous continuous pattern data points.

While we were looking at air pollution data in Madrid, we carefully studied the numbers for each month in the years 2019, 2020, and 2021. This data was collected every hour from different air pollution stations. Interestingly, some months had gaps in the numbers, which could make the data inconsistent. To fix this, we used an interpolation in Python to fill in these gaps.

Conversion to NGSI-LD Format for Improved Compatibility

Once the missing value issue was resolved, we took steps to enhance the dataset's compatibility and interoperability. Transforming the dataset into NGSI-LD format proved instrumental in achieving these objectives. This conversion streamlined data access and facilitated seamless integration with various tools and platforms.

Enriching with Air Quality Levels

A significant stride in data enrichment was the incorporation of air quality levels. We categorized pollution levels as good, bad, or moderate, providing a holistic understanding of the air quality dynamics in Madrid. This enhancement allowed for a clearer assessment of the pollution's severity and its potential implications.

Integrating Meteorological Data

Recognizing the interplay between air pollution and meteorological conditions, we further enriched the dataset by integrating meteorological data. Parameters such as humidity, temperature, wind speed, and pressure were added, offering valuable context for a more profound analysis of the correlation between air pollution and weather conditions.

Through a methodical combination of missing value analysis, interpolation, NGSI-LD format conversion, and the integration of air quality levels and meteorological data[7], it was possible to successfully enrich the dataset, resulting in a comprehensive and robust resource. This enhanced dataset proved to be a powerful tool, enabling accurate and insightful analyses that shed light on the intricacies of air pollution dynamics in Madrid. With meticulous data enrichment, we transformed the initial air pollution dataset into a valuable asset, empowering us to gain deeper insights into the complexities of Madrid's air quality.

[7] https://www.visualcrossing.com/weather/weather-data-services

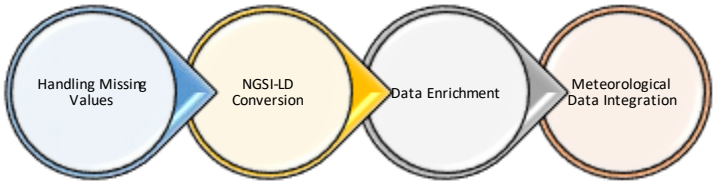


Figure 5. Madrid Air Pollution Data Conversion and Enrichment Process


```
{
    "id": "urn:ngsi-ld:AirQualityObserved:Madr
    "type": "AirQualityObserved",
    "dateObserved": {
        "type": "Property",
        "value": "2020-01-01 00:00:00/2020-01-
    },
    "airQualityLevel": {
        "type": "Property",
        "value": "Moderate"
    },
    "CO": {
        "type": "Property",
        "value": 1.2,
        "unitCode": "GP"
    },
    "O3": {
        "type": "Property",
        "value": 1.0,
        "unitCode": "GQ"
    },
    "temperature": {
        "type": "Property",
        "value": 35.3
    },
    "PM10": {
        "type": "Property",
        "value": 20.0,
        "unitCode": "GQ"
    },
    "C6H6": {
        "type": "Property",
        "value": 3.79999999,
        "unitCode": "GQ"
    },
    "NO2": {
        "type": "Property",
        "value": 85.0,
        "unitCode": "GQ"
    },
    "NO2": {
        "type": "Property",
        "value": 85.0,
        "unitCode": "GQ"
    },
    "NO": {
        "type": "Property",
        "value": 144.0,
        "unitCode": "GQ"
    },
    "SO2": {
        "type": "Property",
        "value": 8.0,
        "unitCode": "GQ"
    },
    "refPointOfInterest": {
        "type": "Relatinship",
        "value": "urn:ngsi-ld:PointOfI
    },
    "windDirection": {
        "type": "Property",
        "value": 356.0
    },
    "windSpeed": {
        "type": "Property",
        "value": 3.1
    },
    "source": {
        "type": "Property",
        "value": "https://aqportal.dis
    },
    "location": {
        "type": "GeoProperty",
        "value": {
            "type": "Point",
            "coordinates": [
                40.4168,
                3.7038
            ]
        }
    },
```


Figure 6. Example NGSI-LD Air Pollution Conversion and Enrichment

3.2.3 Update Madrid Traffic Data

In the context of updating the injection chain of IoT-sensed data, the following improvements are integrated so far:

Treatment of Missing Data

We identify the data which is missing and should be treated. At first, a few of the attributes of the sensor presenting the collected data were under consideration (i.e., Occupancy and sensor status). Occupancy is the time percentage a vehicle occupies the detector in 15 minutes; 50 percent of this means a time interval of the 7 minutes and 30 seconds. The explanation on these attributes, for instance, the Occupancy time interval, is mentioned in Madrid traffic data

structure and content document specification[8]. In order to interpolate the missing values, the pandas-supported utilities are integrated which enables the linear or polynomial interpolation method. Due to the scaled volume of the data, the simple functions are working to encounter or localize the missing values in the data. Next to the localization, the said interpolation methods are handling it to make the data presentation more consistent.

For instance, the captured statistics of a single sensor in Madrid for a month (April 2019) are 145 missing values, corresponding to the Occupancy attribute, which are ignorable due to the volumetric nature of the Madrid traffic data : 4 sensed tuples per hour by single sensor * 24 hours * 30 days(+1, -1), while the total missing values for all sensor in a single month may introduce the inconsistencies and problems to have a correlative view on data (8443 tuple values are missed in April, 2019 for the discussed attribute of the sensed data. However, the sensors were active during this period of time.). A short description of the snapshotting or localizing of the missing value for a single attribute can be captured using the codelet employing simple logic in Figure 7.

```
ocupacion_lst = re.findall(r'\d+', str (each_sensor_based_DF.loc[default_index]["ocupacion"]))
#print ('\n\n occupasion split value', )

#*************************************************************************************************
if ocupacion_lst != []:     # NO VALUE IN  THE SERIES ON LOCATION OF DEFAULT ROW INDEX FOR THE COLUMN  OF oCCUPANCY

    Occupancy_INT_VALUE = int(ocupacion_lst[-1])

    #print ("\n\n Occupancy_INT_VALUE",repr (Occupancy_INT_VALUE) )
    traffic_flow_dict_template["Occupancy"] = {"type": "Property", "value": Occupancy_INT_VALUE}
else :
    print ("\n\n here is empty occpancy value for the ---> index", default_index, rows)
    traffic_flow_dict_template["Occupancy"] = {"type": "Property", "value": None}
    print ("\n \n  iterpolation result", each_sensor_based_DF.loc[default_index,"ocupacion"].interpolate())
```

Figure 7. An example of missing data treatment

Large Scale Data Loading Using Chain

The other context of the improvement was to deal with large data loading for applying the different functions on it (i.e., a pipeline to make the data context-aware and mappable to the corresponding Smart Data Model). The simplification of the chain or its execution should not demand any highly capable resource or computational node at the user ends when the potential user intends to translate the volumetric data compliant to the NGSI–LD format. For this reason, one of the solutions is identified to introduce the lightweight child process or thread (referred to task A in figure 13), supported by Python, which will take care of the data to load according to the user specified delay in reading or writing of consecutive data tuples, chunks or aggregate of the tuples across the entity (sensor), aiming at analysis, transformation, and rest of the treatments. This computation aware data handling will continue till the writing of NGSI-LD formatted data into files. This makes the execution of the IoT data chain lightweight in terms of file input/output streaming. Especially when there is need to deal with 10 million tuples in single CSV file. For instance, these are the counts of one month (April, 2019) of the traffic sensed data in Madrid. An illustration for the ongoing and completed tasks, partially from the IoT chain development processes is illustrated in .

8 https://datos.madrid.es/FWProjects/egob/Catalogo/Transporte/Trafico/ficheros/Estructura_DS_Contenido_Trafico_Historico.pdf

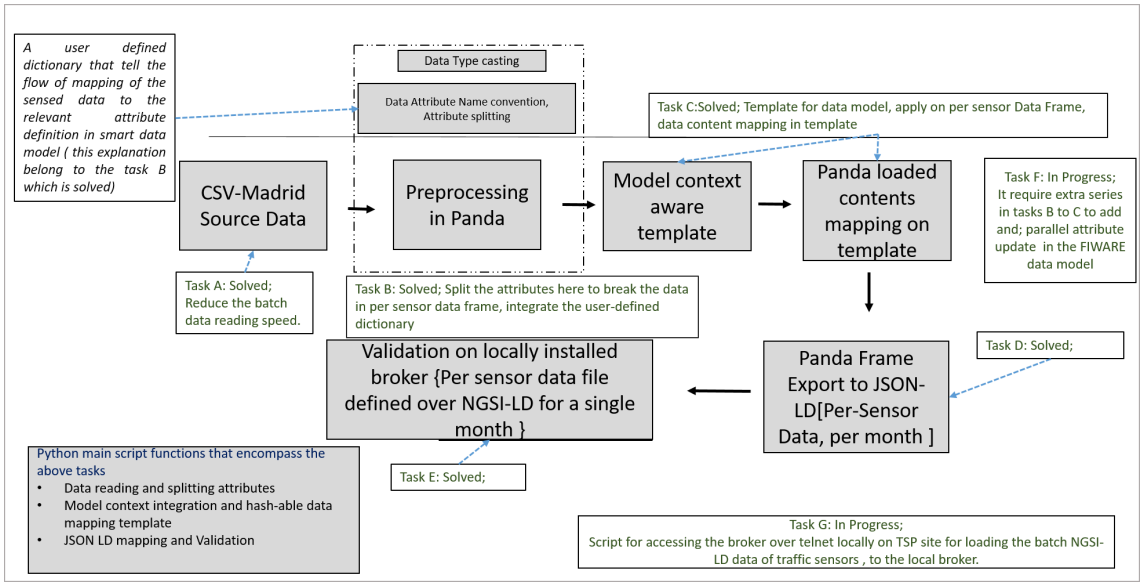


Figure 8. Ongoing tasks status across the IOT data injection chain

Data linking between traffic and air quality data

The linking is done based on the location (e.g., Madrid city, Dublin city, etc.) and specific data sets (Traffic and Air Quality). Traffic data obtained from traffic stations consists of information about traffic intensity. Air Quality data obtained from air stations consists of air pollutant data such as CO, NO, NO2, PM10, etc. and weather data such as temperature, humidity, etc. The linkage between any feature of Traffic data and any feature of Air Quality data can be calculated. To compute the linkage, the Pearson correlation coefficient and p value are used. The correlation can be used to validate which features in Air Quality data are associated with traffic intensity.

Data linking is provided as a service. Data is evaluated by querying the context broker and using Python tools on the time series data. Therefore, according to the request, the context broker is queried for the required data, and the correlation value can be calculated between any two features of two data sets, hourly, daily, monthly, etc. Traffic measurements are obtained every 15 minutes, and air data is obtained hourly. Therefore, the time is harmonized, so the traffic data is converted to hourly data.

According to Figure 9, first the required features to find the linking are selected, and then the time range. Afterwards, the context broker is queried to obtain the required data, and finally, the correlation and the P value are calculated between the selected features in the selected time range.

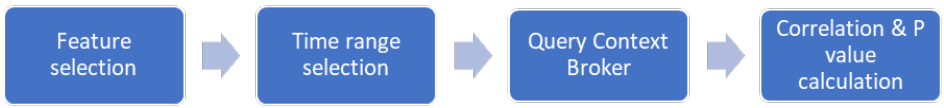


Figure 9. Data Linking Service (Traffic & Air Quality Data).

- Pearson Correlation Coefficient (r)

The Pearson Correlation Coefficient (r) is a measure of the linear correlation between two data sets. Since it is a normalized measurement covariance, the result is between -1 and 1. A value close to 1 suggests a positive correlation, a value close to -1 suggests a negative correlation, and a value close to 0 suggests no correlation between two features. The Correlation Coefficient is calculated according to below equation given below:

$$r = \frac{\sum_{i=0}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=0}^{n}(x_i - \overline{x})^2 \sum_{i=0}^{n}(y_i - \overline{y})^2}}$$

$$r = correlationcoefficient$$

$$x_i = valuesofthex - variable \in asample$$

$$\overline{x} = meanofthevaluesofthex - variable$$

$$y_i = valuesofthey - variable \in asample$$

$$\overline{y} = meanofthevaluesofthey - variable$$

- P value

P value is a statistical measurement used to validate a hypothesis against observed data. The p value is defined as the probability of obtaining results at least as extreme as the observed results of a statistical hypothesis test under the assumption that the null hypothesis is true. A smaller p value suggests that there is stronger evidence in favour of the alternative hypothesis.

Consider this example of the calculation of the Pearson Correlation Coefficient (r) and P Value. The interesting time range is 24 hours, and the time is synchronized hourly. The time range of interested and two features to be compared can be selected as required. In this example, NO_2 from Air Quality data is considered as x values and Traffic intensity from Traffic Flow data is considered as y values.

Table 1:Time series data for Traffic intensity and NO2 value

Time Stamp	$NO_2 value$ x	$TrafficIntensity$ y
1	85	558
2	69	2316
3	70	2516
4	61	1324
5	55	1091
6	48	786

7	46	793
8	48	702
9	48	433
10	43	290
11	35	334
12	40	671
13	45	1125
14	54	1517
15	57	1945
16	58	1366
17	67	1100
18	72	1667
19	83	1771
20	89	1874
21	92	1757
22	95	1539
23	98	1184
24	113	895

The mean values are as follows:

$$\overline{x} = 65.4$$

$$\overline{y} = 1231.4 r = 0.415$$

$$pvalue = 0.0437$$

Therefore, the Pearson correlation coefficient and p value are **0.415** and **0.0437**, respectively. The considerable correlation coefficient and lower p-value suggest good linkage between the two features.

3.3 WEB DATA INJECTION CHAIN

There are no updates regarding the web data injection chain. Anyhow the exploration of possible additional data sources continues, that could enrich the applications already implemented or contribute to the development of new ones.

3.4 SOCIAL MEDIA DATA INJECTION CHAIN

In the previous deliverables [2] [1] [3], we elaborated on how different entities can be constructed and the properties can be filled with all the relevant accessible data. Now, we have given utmost importance to data privacy, considering the latest privacy policies. These policies encompass aspects such as data anonymity, protection of sensitive information, and the proper usage and publication rights, drawing inspiration from the Twitter Developer document[9] .

Based on these considerations, we have made significant changes to how data is handled when sending it to the broker. Instead of including comprehensive details of tweets and users, we now transmit a condensed version of the data. This condensed version only contains essential information, such as tweet IDs, publish dates, and user IDs, while omitting sensitive details like tweet text, usernames, and other private information. This approach ensures that user privacy and data confidentiality are safeguarded.

Furthermore, we have implemented a collection process to maintain a record of how and why a particular set of tweets was collected. This enhancement improves data organization and traceability, making it easier to understand why certain data sets were collected and to access them later.

Addressing the enrichers, we have restructured the code for sentiment and translation analysis. After obtaining a tweet, we immediately perform translation, followed by sentiment analysis. The resulting sentiment labels and scores are the only pieces of information retained from the process. All textual content, including the original tweets, is no longer stored. This way, we maintain the necessary data for analysis while mitigating potential privacy risks associated with retaining sensitive information.

By adhering to these data curation practices, we ensure that the processed data remains compliant with privacy policies, respects user rights, and facilitates ethical data handling.

[9] https://developer.twitter.com/en/developer-terms/more-on-restricted-use-cases

3.5 AGRICULTURAL DATA INJECTION CHAIN

3.5.1 Data Collection

There are many smart agriculture sites in the world, each of them is producing and dealing with a lot of data. For instance, the region of Lorca in Murcia provides data about crops by means of the Spanish National Organization SIGPAC, relating the geographical area with crops and CO2 measurements. Related to this geographical area, a set of data with CO2 retained in crops and soil is managed through a dedicated injection chain that has been developed in SALTED.

CO2 data

Given the recent environmental policies derived from these serious threats caused by global climate change, practical measures to decrease net CO2 emissions have been established. Carbon sequestration is a major measure to reduce atmospheric CO2 concentrations within the short and medium term, in which terrestrial ecosystems play an essential role as carbon sinks. Quantification and monitoring of organic carbon reservoirs are needed to inform environmental management, adapt local policies, and assess potential impacts.

The CO2 data set is obtained by means of LIDAR (Light Detection and Ranging). LIDAR is an active system for remote detection based on laser sensors that allow to get on real time more geo-located measure densities than previous coverages available for reservoirs of CO2 both in the soil (below-ground carbon sequestration) and the vegetation (above ground carbon sequestration biomass). LIDAR get information with a density of 0.5 pixel/m2. As a result of this, it produces a digital image with continuous spatial information of 25m x 25m cell size. The data set file is obtained converting raster images with the carbon footprint to a geojson file, and it is updated from LIDAR system every 6 years since its start in 2015.

Characteristics of CO2 data

The geojson file provides a single CO2 value per line, referring to a polygon as a geographical parcel delimited by the lines that connect the points associated. The values don't contain a timestamp.

- Tn_C_ha: CO2 measure expressed is tons of carbon by the hectare
- Geometry: polygon coordinates for the CO2 measure

An example of a file with CO2 geojson data is shown in Figure 10.

{

"type": "FeatureCollection",

"name": "Stock_Banda1_Lorca_P30_4326_Alineado_geo",

"crs": { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },

"features": [

{ "type": "Feature", "properties": { "Tn_C_ha": 34 }, "geometry": { "type": "Polygon", "coordinates": [[[-1.82500101488498, 37.96917], [-1.82500101488498, 37.968336666666154], [-1.824167686062246, 37.968336666666154], [-1.824167686062246, 37.96917], [-1.82500101488498, 37.96917]]] } },

.....

{ "type": "Feature", "properties": { "Tn_C_ha": 28 }, "geometry": { "type": "Polygon", "coordinates": [[[-1.7033350067659, 37.423336666333846], [-1.703335006 7659, 37.422503333], [-1.702501677943167, 37.422503333], [-1.702501677943167, 37.423336666333846], [-1.7033350067659, 37.423336666333846]]] } }

]

}

Figure 10. Example CO2 geojson data

Land and Crop data

This data is provided by the SIGPAC public web organization (Servicio de Información Geográfica de Parcelas Agrícolas) which provides information on land divisions and crops in different regions in Spain. This data is updated with a low frequency (once a year) because the variations due to changes in crop and resizing of parcels are very infrequent. Data can be downloaded from its web site[10] .

The data is obtained through a REST API and therefore, we have labelled it as stream data. Since data is received in an asynchronous way, it causes us to use a completely different collection procedure. The effort made with these large data sets is to collect them, provide a homogeneous format, and curate the data in such a way that applications can access them with a well-defined and formed structure.

Characteristics of Land and Crop data

Crop values are provide per parcel, where each parcel is defined by a polygon or multipolygon using the geographical coordinates from each delimited area.

- dn_oid: land identification (e.g.: 1214602685)
- provincia: province number (e.g.: 30)
- polígono: polygon number e.g.: 27)
- parcela: parcel number (e.g.: 9004)
- recinto: enclosure number (e.g.: 1)
- dn_surface: polygon Surface expressed in square meters (e.g.: 6441.0315812299996)
- pend_media: ground incline medium (e.g.: 142)
- coef_rega: irrigation coefficient for the ground (e.g.: null)

[10] https://sigpac.mapama.gob.es/fega/visor/?provincia=30

- uso_sigpac: SIGPAC uses (e.g.: AG)
- grp_cult: crop group (e.g.: PT)
- sigpac_n: (e.g.: 1)
- geometry: geographical coordinates for the ground area

Figure 11 shows an example of a file with Land and Crop geojson data.


```
{
'type': "FeatureCollection",
'name': "SIGPAC_Lorca_20210104_4326_cod_GeoJ",
'crs': { "type": "name", "properties": { "name": "urn:ogc:def:crs:OGC:1.3:CRS84" } },
'features': [
{ "type": "Feature", "properties": { "dn_oid": 1214602685, "provincia": 30, "poligono": 27, "parcela":
'geometry': { "type": "MultiPolygon", "coordinates": [ [ [ [ -1.730890749094084, 37.865401176349884 ],
{ "type": "Feature", "properties": { "dn_oid": 1214602686, "provincia": 30, "poligono": 27, "parcela":
 "geometry": { "type": "MultiPolygon", "coordinates": [ [ [ [ -1.737792003730953, 37.87127752617166 ],
{ "type": "Feature", "properties": { "dn_oid": 1214602687, "provincia": 30, "poligono": 27, "parcela":
 "geometry": { "type": "MultiPolygon", "coordinates": [ [ [ [ -1.742741236216798, 37.857748772386387 ]
{ "type": "Feature", "properties": { "dn oid": 1214602688, "provincia": 30, "poligono": 27, "parcela":
```


Figure 11. Example (cutout) SIGPAC land and crop geojson data.

3.5.2 Data Mapping

AirQualityObserved

Smart Data Models defined within the FIWARE activities cover many aspects related to air quality. For representing the CO2 reservoirs information, the choice is to use the *AirQualityObserved* data model because that model fits best the available data.

This model allows us to represent different pollutants (CO2 in this case) and their measure and to associate them to a location and to a timestamp (timestamp applies to all data in the data file). This organization is illustrated by this snapshot in Figure 12 taken from the *AirQualityObserved* specification[11].

[11] https://github.com/smart-data-models/dataModel.Environment/blob/master/AirQualityObserved/doc/spec_ES.md

```
{
    "id": "urn:ngsi-ld:AirQualityObserved:Madrid-AmbientObserved-28079004-2016-03-15T11:00:00",
    "type": "AirQualityObserved",
    "CO": 500,
    "CO_Level": "moderate",
    "NO": 45,
    "NO2": 69,
    "NOx": 139,
    "SO2": 11,
    "address": {
        "addressCountry": "ES",
        "addressLocality": "Madrid",
        "streetAddress": "Plaza de Espa\u00f1a",
        "type": "PostalAddress"
    },
    "airQualityIndex": 65,
    "airQualityLevel": "moderate",
    "areaServed": "Brooklands",
    "dateObserved": "2016-03-15T11:00:00/2016-03-15T12:00:00",
    "location": {
        "coordinates": [
            -3.712247222222222,
            40.423852777777775
        ],
        "type": "Point"
    },
    "precipitation": 0,
    "refPointOfInterest": "urn:ngsi-ld:PointOfInterest:28079004-Pza.deEspanya",
    "relativeHumidity": 0.54,
    "reliability": 0.7,
    "source": "http://datos.madrid.es",
    "temperature": 12.2,
    "typeOfLocation": "outdoor",
    "windDirection": 186,
    "windSpeed": 0.64,
    "@context": [
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
        "https://raw.githubusercontent.com/smart-data-models/dataModel.Environment/master/context.jsonld"
    ]
}
```


Figure 12. Example of AirQualityObserved entity smart data model.

This data model is flexible and extensible in such a way to allow the formatting of the air quality geojson data file described in the previous chapter into a well-formatted data set. The considered data will be organized per location and type of pollutants.

For the carbon footprint encountered in the different data sets, SALTED has defined a description of it conformant to the data model and capable of representing the value, the polygon and location in which it was measured and a timestamp from the whole measures.

AirQualityObserved Smart Data Model has the following mandatory properties:

- Id[]: entity identifier
- dateObserved [string]: observation date and time (format ISO8601 UTC)
- location []: geojson reference to the element.
- type [string]: entity type

For the Land and Crop Carbon Footprint for Smart Agriculture, the CO2 and area served properties are included (cf Figure 13):

- CO2 [number]: CO2 observed
- areaServed [string]: area the measure belongs to.

```
[
    {
        "id": "urn:ngsi-ld:AirQualityObserved:lorca:DB880BB9083DFB0FC42F49AEE2029678",
        "type": "AirQualityObserved",
        "areaserved": {
            "type": "Property",
            "value": "LORCA"
        },
        "co2": {
            "type": "Property",
            "value": 0.3,
            "unitCode": "28"
        },
        "dateObserved": {
            "type": "Property",
            "value": "2016-09-30T00:00:00"
        },
        "location": {
            "type": "GeoProperty",
            "value": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            -1.860000825439784,
                            37.94250333331707
```


Figure 13. AirQualityObserved entity example resulting from the Injection Chain with CO2 data

AgriCrop

For representing the Land and Crop information, the choice is to use the *AgriCrop* data model because that model fits best the available data.

AgriCrop entities contain a harmonized description of a generic crop. For each kind of crop described in SIGPAC file for the observed area, a new entity is injected to the NGSI-LD Context Broker.

AgriCrop Smart Data Model has the following mandatory properties. Figure 14 shows an example for a AgriCrop entity resulting from the Injection Chain:

- Id[]: entity identifier
- name [string]: crop name
- type [string]: entity type


```
{
    "id": "urn:ngsi-ld:AgriCrop:4F3C7B50C6D2579ABEB05FBA29A32FA6",
    "type": "AgriCrop",
    "name": {
        "type": "Property",
        "value": "frutos secos y olivar"
    },
    "@context": [
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.3.jsonld"
    ]
},
```


Figure 14. AgriCrop entity example resulting from the Injection Chain.

AgriParcel

Smart Data Models defined within the FIWARE activities cover many aspects related to land and crop. For representing this information, the choice is to use the *AgriParcel* data model because that model fits better the available data. This entity contains a description for a generic parcel. It is mainly associated to agriculture category and IoT related applications. An example is showed in the Figure 15 (the figure stretches over multiple pages):


```
{
    "id": "urn:ngsi-ld:AgriParcel:72d9fb43-53f8-4ec8-a33c-fa931360259a",
    "type": "AgriParcel",
    "area": {
        "type": "Property",
        "value": 200
    },
    "belongsTo": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:AgriFarm:f67adcbc-4479-22bc-9de1-cb228de7a765"
    },
    "category": {
        "type": "Property",
        "value": "arable"
    },
    "createdAt": "2017-01-01T01:20:00Z",
    "cropStatus": {
        "type": "Property",
        "value": "seeded"
    },
    "description": {
        "type": "Property",
        "value": "Spring wheat"
    },
    "hasAgriCrop": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:AgriCrop:36021150-4474-11e8-a721-af07c5fae7c8"
    },
    "hasAgriParcelChildren": {
        "type": "Relationship",
        "object": [
            "urn:ngsi-ld:AgriParcel:26ba4be0-4474-11e8-8ec1-ab9e0ea93835",
            "urn:ngsi-ld:AgriParcel:2d5b8874-4474-11e8-8d6b-dbe14425b5e4"
        ]
    },
    "hasAgriParcelParent": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:AgriParcel:1ea0f120-4474-11e8-9919-672036642081"
    },
```

```
"hasAgriSoil": {
    "type": "Relationship",
    "object": "urn:ngsi-ld:AgriSoil:429d1338-4474-11e8-b90a-d3e34ceb73df"
},
"hasDevice": {
    "type": "Relationship",
    "object": [
        "urn:ngsi-ld:Device:4a40aeba-4474-11e8-86bf-03d82e958ce6",
        "urn:ngsi-ld:Device:63217d24-4474-11e8-9da2-c3dd3c36891b",
        "urn:ngsi-ld:Device:68e091dc-4474-11e8-a398-df010c53b416",
        "urn:ngsi-ld:6f44b54e-4474-11e8-8577-d7ff6a8ef551"
    ]
},
"lastPlantedAt": {
    "type": "Property",
    "value": {
        "@type": "DateTime",
        "@value": "2016-08-22T10:18:16Z"
    }
},
"location": {
    "type": "GeoProperty",
    "value": {
        "type": "Polygon",
        "coordinates": [
            [
                100,
                0
            ],
            [
                101,
                0
            ],
            [
                101,
                1
            ],
            [
                100,
                0
            ]
        ]
    }
},
"modifiedAt": "2017-05-04T12:30:00Z",
"ownedBy": {
    "type": "Relationship",
    "object": "urn:ngsi-ld:Person:fce9dcbc-4479-11e8-9de1-cb228de7a15c"
},
"relatedSource": {
    "type": "Property",
    "value": [
        {
            "application": "urn:ngsi-ld:AgriApp:72d9fb43-53f8-4ec8-a33c-fa931360259a",
            "applicationEntityId": "app:parcel1"
        }
    ]
},
"seeAlso": {
    "type": "Property",
    "value": [
        "https://example.org/concept/agriparcel",
        "https://datamodel.org/example/agriparcel"
    ]
},
"@context": [
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld",
    "https://raw.githubusercontent.com/smart-data-models/dataModel.Agrifood/master/context.jsonld"
]
}
```


Figure 15. Example of AgriParcel entity Smart Data Model.

AgriParcel Smart Data Model has the following mandatory properties:

- Id[]: entity identifier
- area [string]: crop name
- location []: geojson reference to the element.
- type [string]: entity type
- hasAgriCrop []: relation that reference the crop for this parcel

Furthermore, for the Land and Crop Carbon Footprint (**LCCF**) use case, it is also included the CO2 and area served properties:

- category [string]: type of parcel (for instance: **hacienda, pradera, viñedo, huerto, cultivo mixto, tierras bajas, tierras altas, retirada de tierras, silvicultura, humedal**).
- hasAirQualityObserved[]: relationship that reference the "AirQualityObserved" entities related to this object

Figure 16 shows an example of entity injected to the NGSI-LD Context Broker.


```
{
    "id": "urn:ngsi-ld:AgriParcel:lorca:SIGPAC_Lorca_20210104_4326:8612608F61927004585A4069569BCE9A",
    "type": "AgriParcel",
    "area": {
        "type": "Property",
        "value": 37965.90040273,
        "unitCode": "MTK"
    },
    "category": {
        "type": "Property",
        "value": "CA: viales"
    },
    "hasAgriCrop": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:AgriCrop:98C987F77116079AF1A73CCCDE00058E"
    },
    "hasAirQualityObserved": {
        "type": "Relationship",
        "object": [
            "test_0",
            "test_1",
            "test_2"
        ]
    },
    "location": {···
    },
    "@context": [
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context-v1.3.jsonld"
    ]
```


Figure 16. AgriParcel entity with land and crop data example resulting from the Injection Chain.

3.5.3 Data Curation

The curation phase may take care of the following aspects of the dataset: the resolution of some problems related to the lack of data or their correctness in a short period; the broader issue of harmonizing the actual dataset to the rules and the expected quality of measurement in the large; and the evaluation of the goodness and usability of the formatted data in

applications. The first aspect deals with the recovery of data in small periods of time and the interpolation of missing data in a short period of time. The second aspect will cope with the necessary sampling of data, the availability of data that can be semantically comparable to data of the same time of other data sets or previous data of the same source or the alignment to a system of measures. In other terms, the possibility of linking and comparing the transformed data with other ones. The third aspect is related to an evaluation of the value and usability of the transformed data.

Other aspect for data curation is repeated data and wrong data. For instance, if the coordinates of a polygon are wrong and the polygon is not closed. For Land and Crop Carbon Footprint use case, curation data in SALTED solution is focused on these last aspects: repeated data and coordinates errors. Once the data is read from files and mapped to the Smart Data Models, the application looks for repeated data to remove them from the list before the injection phase. In this way, it avoided the possibility of setting the same parcel to the NGSI-LD Context Broker (i.e. Scorpio). For the second point, before the injection phase, polygons are checked to fix the coordinates. For each polygon the application checks if the polygon is not closed. In that case, the coordinates of the last point are fixed to get a closed polygon.

3.5.4 Data Linking

The addition of specific metadata that is linked to the datasets to incorporate data quality assessment evaluation should be structured in a hierarchical way in order to evaluate and assess the most granular element (e.g., the sensor/detector) and to associate with it an evaluation related to a minimum period of time (e.g., one day). For data linking information, Amper solution set the relation between the crop parcels and the CO2 polygons.

These relations allow determine the relation between the amount of CO2 retained with the different crop types, which give us information about the crops that can help to reduce CO2 and their contribution to reduce the climate change. To set the relation of AgriParcel model with AirQualityObserved model, a change in AgriParcel Smart Data Model has been performed in order to include the new relation “hasAirQualityObserved”. This new attribute is a “Relationship” type with the different AirQualityObserved entities related (cf Figure 17).


```
"hasAirQualityObserved": {
        "type": "Relationship",
        "object": [
          "urn:ngsi-ld:airqualityobserved_entity_0",
          "urn:ngsi-ld:airqualityobserved_entity_1",
          "urn:ngsi-ld:airqualityobserved_entity_2"
        ]
      }
```


Figure 17. Example of AirQualityObserved entities linked to AgriParcel entity.

To get this relationship, we have made use of different algorithms to get the overlaps between the SIGPAC parcels and the CO2 polygons. Due to the great amount of data to be analysed, this process can take some time. This relationship will help to show later in demonstrative LCCF application different aspects of the study.

4 DET ENRICHERS

4.1 OVERVIEW

Injection Chains, specified in D2.2, are the starting element of DETs of the SALTED project. They are the data pipeline processes that handle discovery and collection of heterogeneous data, NGSI-LD mapping, data curation and finally injection of this data into a Linked-Data Context Broker with the purpose of enabling smart agriculture and smart city applications.

Enrichment Chains, on the other hand, have the goal to identify and provide additional information that can be inferred from the data. This augmented information is necessary for the advancement of the smart agriculture and smart city applications targeted.

To achieve this, different services are developed, each offering a specific functionality for enhancing the data with a use case in mind. The following sections describe the enrichment services created for each targeted application, namely:

- **Agenda Analytics:**
  - Scope: Matching of given agendas to the activities of companies and public administration. The obtained matching scores can be used for detailed reporting and visualization.
  - Enrichers: Crawling Enricher, Agenda Matching Enricher
- **Sentiment App:**
  - Scope: Adding two components for translating the textual data and applying sentiment analysis useful for future applications concerning the general sentiment of people regarding different topics.
  - Enrichers: Translation Enricher, Sentiment Analysis Enricher
- **LCCF: Land and Crop Carbon Footprint**
  - Scope: Parcels without CO2 data related. Add calculated CO2 data to those parcels using data from neighbour parcels with the same crop type.
  - Enrichers: Gap Calculator
- **CLIFF: City Liveability Index**
  - Scope: Perform quality and value augmentation on the cities and its regions using available indices.
  - Enrichers: City Liveability Index
- **Application agnostic enrichers**
  - Enrichers: Reverse Geocoding Enricher, Insight Data Enricher

4.2 CRAWLING ENRICHER (AGENDA ANALYTICS)

The Agenda Analytics use case involves two Enrichers: Crawling and AgendaMatching. Their basic functionality is already described in D2.2 [2], which is why this report will only sum up the basic functionality and the adaptions made regarding this two (starting with the Crawling in this section and the Agenda Matching within the following section). The following figure shows the setup of those services within the DET of Agenda Analytics. Both enriching services have entities within the NGSI-LD Context Broker (i.e. Scorpio) as starting point, mainly organizations. They both generate *DataServiceRun* entities which are published within the broker, which persist their results. They are either used for transparency reasons or are used by the services displayed in Figure 18.

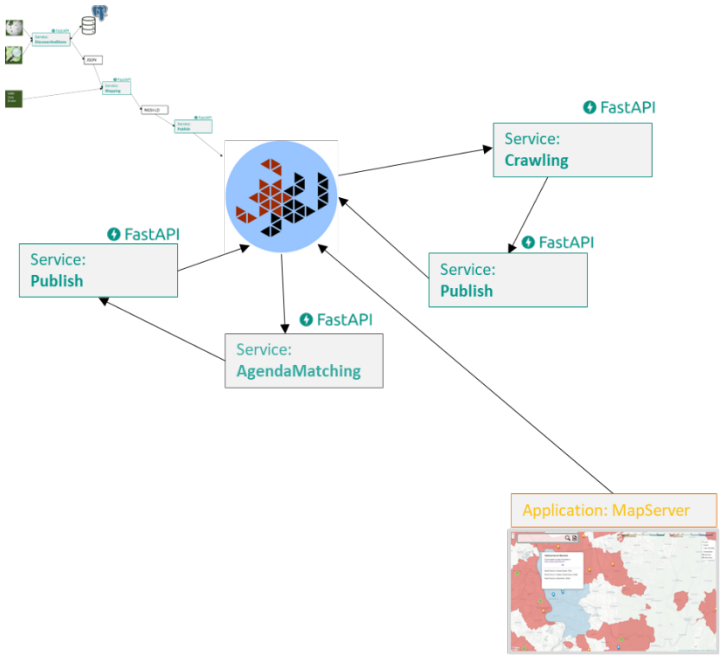


Figure 18. Agenda Analytics Enriching.

In the Crawling Enricher, each Organization entity in the broker is augmented with information regarding a specific agenda. With e.g. the SDGs (Sustainable Development Goals) as agenda the service searches the public web for information on the sustainability efforts of a specific organization. It uses the NGSI-LD entity of an organization and a keywords parameter to specify the data to be crawled, such as "sustainability report pdf". Further parameters specify the actuality of already existing crawling results, that would prevent the service of running again, and the agenda for which the crawling runs. Figure 19 shows the service's representation within the Scorpio Broker using the *DataService* data model:

```
{
            "id": "urn:ngsi-ld:DataServiceDCAT-AP:Salted-Crawling",
            "type": "DataServiceDCAT-AP",
            "alternateName": {
                "type": "Property",
                "value": "Crawling"
            },
            "dataProvider": {
                "type": "Property",
                "value": "Kybeidos GmbH"
            },
            "dataServiceDescription": {
```

```
            "type": "Property",
            "value": [
                "Service that crawls documents for a specific NGSI-LD entity of type Organization"
            ]
        },
        "dateCreated": {
            "type": "Property",
            "value": "2023-06-09T08:00:00Z"
        },
        "dateModified": {
            "type": "Property",
            "value": "2023-06-09T08:00:00Z"
        },
        "description": {
            "type": "Property",
            "value": "Data service for the SALTED Data Enrichment Toolchain"
        },
        "endPointDescription": {
            "type": "Property",
            "value": [
                "FAST API end point without authentication",
                "REST API compliant"
            ]
        },
        "endPointURL": {
            "type": "Property",
            "value": [
                "https://salted-det.hdkn.eu:8004/docs#"
            ]
        },
        "seeAlso": {
            "type": "Property",
            "value": [
                "https://salted-project.eu/"
            ]
        },
        "servesDataset": {
            "type": "Property",
            "value": [
                "NGSI-LD entities of type DataServiceRun"
            ]
        },
        "title": {
            "type": "Property",
            "value": [
                "Salted Crawling"
            ]
        },
        "assetProvider": {
            "type": "Property",
            "value": [
                "https://fastapi.tiangolo.com/",
                "https://github.com/DigitalPebble/storm-crawler/"
            ]
        },
        "contactPoint": {
            "type": "Property",
            "value": {
                "name": "contact point for Salted Crawling",
                "email": "maren.dietzel@kybeidos.de"
            }
        },
        "configuration": {
            "type": "Property",
            "value": [
                "keywords",
                "actuality_days",
                "target_agenda_entity_id"
            ]
        },
        "@context": [
            "https://raw.githubusercontent.com/smart-data-models/dataModel.DCAT-AP/master/context.jsonld",
            "https://smartdatamodels.org/context.jsonld"
        ]
    }
```


Figure 19. DataServiceDCAT-AP for Crawling (Agenda Analytics).

Through the ongoing evaluation of the quality of the results in conjunction with the technical performance (e. g. storage capacity and time needed), the possibilities of allowing the crawling

of text and the specification of depth to crawl using additional parameters are tested at the moment.

4.3 AGENDA MATCHING ENRICHER (AGENDA ANALYTICS)

The second Enricher, AgendaMatching, processes the crawled information and performs compliance matching. It takes the entity ID of an organization as input and uses an additional parameter that allows for the specification of the agenda that is to be used. Figure 20Figure 20 shows the service's representation within the NGSI-LD Context Broker using the *DataService* data model:


```
{
            "id": "urn:ngsi-ld:DataServiceDCAT-AP:Salted-AgendaMatching",
            "type": "DataServiceDCAT-AP",
            "alternateName": {
                "type": "Property",
                "value": "AgendaMatching"
            },
            "dataProvider": {
                "type": "Property",
                "value": "Kybeidos GmBH"
            },
            "dataServiceDescription": {
                "type": "Property",
                "value": [
                    "Service that matches documents identified by Salted-Crawling (type DataServiceDCAT-AP) the for a specific
                     NGSI-LD entity of type Organization with the SDG goals"
                ]
            },
            "dateCreated": {
                "type": "Property",
                "value": "2023-06-09T08:00:00Z"
            },
            "dateModified": {
                "type": "Property",
                "value": "2023-06-09T08:00:00Z"
            },
            "description": {
                "type": "Property",
                "value": "Data service for the SALTED Data Enrichment Toolchain"
            },
            "endPointDescription": {
                "type": "Property",
                "value": [
                    "FAST API end point without authentication",
                    "REST API compliant"
                ]
            },
            "endPointURL": {
                "type": "Property",
                "value": [
                    "https://salted-det.hdkn.eu:8005/docs#"
                ]
            },
            "seeAlso": {
                "type": "Property",
                "value": [
                    "https://salted-project.eu/"
                ]
            },
            "servesDataset": {
                "type": "Property",
                "value": [
                    "NGSI-LD entities of type DataServiceRun",
                    "NGSI-LD entities of type KeyPerformanceIndicator"
                ]
            },
            "title": {
                "type": "Property",
                "value": [
                    "Salted AgendaMatching"
                ]
            },
            "assetProvider": {
                "type": "Property",
                "value": [
                    "https://fastapi.tiangolo.com/"
                ]
            },
            "contactPoint": {
                "type": "Property",
                "value": {
                    "name": "contact point for Salted AgendaMatching",
                    "email": "maren.dietzel@kybeidos.de"
                }
```

```
        },
        "configuration": {
            "type": "Property",
            "value": “target_agenda_entity_id"]
        },
        "@context": [
            "https://raw.githubusercontent.com/smart-data-models/dataModel.DCAT-AP/master/context.jsonld",
            "https://smartdatamodels.org/context.jsonld"
        ]
    }
```

Figure 20. DataServiceDCAT-AP for Agenda Matching (Agenda Analytics).

The further development of this services focusses on the performance, since the matching process takes a significant amount of time, when the agendas and matching texts are long, which is pretty usual for e. g. sustainability reports.

4.4 SENTIMENT ANALYSIS ENRICHER (TRAFFIC SENTIMENT APP)

As part of the SALTED WP4 developments it has been decided to develop a Sentiment Analysis Enricher which focuses on sentiment analysis of tweets related to traffic in the city of Madrid.

By leveraging the power of natural language processing, we aim to extract valuable insights from social media data and gain a comprehensive understanding of public sentiments regarding traffic conditions.

We will further explain the details and challenges of the application in the upcoming WP4 deliverable.

To achieve sentiment analysis, we recognized the need for two enricher components: a translation component and a sentiment analysis component.

The presence of diverse languages on social media platforms, particularly Twitter, poses a challenge for our project. Developing and utilizing separate sentiment analysis models for each language would be both impractical and time-consuming. By implementing a translation component, we aim to mitigate the problem of multilingual data. It allows us to convert tweets from various languages into a common language (English). This enables us to apply a single sentiment analysis model across all tweets.

We will describe the details of the enricher components in the following subsections.

4.4.1 Translation Component

To address the multilingual nature of tweets, we implemented a translation component within our application. We experimented with two different approaches to achieve effective translation capabilities.

Initially, we utilized the translation model provided by Hugging Face named Helsinki-NLP/opus-mt-en-es model[12]. This model specializes in translating text from Spanish to English, which was the primary language used in tweets about the city of Madrid. The advantage of this approach is that it runs on our own hardware, allowing us to leverage the available computational power efficiently. However, a limitation of this approach is its restricted language support, as it only works for Spanish. To achieve a more comprehensive translation component, it would be beneficial to incorporate support for additional languages. Although we no longer utilize this model in our current implementation, we have retained the corresponding code as comments, to be used if the need arises.

[12] https://huggingface.co/Helsinki-NLP/opus-mt-en-es

```
# # First Approach

# from transformers import pipeline
# # Initialize the translation pipeline
# translator = pipeline('translation', model='Helsinki-NLP/opus-mt-es-en')

# # Get the text to translate from the tweet entity
# text_to_translate = tweet["hasText"][0]["value"]
# # text_to_translate = 'Cómo has estado?'
# translated_text = translator(text_to_translate, max_length=280)[0]['translation_text']
```

Figure 21. Code for translation from Spanish to English using Transformers' translation pipeline.

To overcome the limitations of the first approach and enable translation across multiple languages, we explored the Deep-Translator library[13]. This library provides access to various well-known translation services. By utilizing the Deep-Translator library, we gain the flexibility to choose among different translators based on our requirements. The option we selected is Google Translator. This choice was made due to Google Translator's reputation for accuracy and its extensive language coverage. By setting the source language to "automatic", we can now translate tweets from any language to English, expanding the scope of our translation component beyond a single language.

```
from deep_translator import GoogleTranslator

# Get the text to translate from the tweet entity
text_to_translate = tweet["hasText"][0]["value"]

# Use any translator you like, in this example GoogleTranslator
translated_text = GoogleTranslator(source='auto', target='en').translate(text_to_translate)
```

Figure 22. Code for automatic translation from any language to English using Google Translator.

As part of the application's workflow, the translated text obtained from the translation component is further processed and presented in the form of an *SMAnalysis* entity following the NGSI-LD specification from the *SocialMedia* data model[14] introduced in the previous deliverables (cf. Figure 23).

[13] https://pypi.org/project/deep-translator/

[14] https://github.com/smart-data-models/dataModel.SocialMedia

```python
from datetime import datetime
# Create an Analysis entity to store the translation result
analysis = {
    "id": f"urn:ngsi-ld:Analysis:{generatedAnalysisId}",
    "type": "SMAnalysis",
    "analyzedAt": {
        "type": "Property",
        "value": {
            "@type": "DateTime",
            "@value": datetime.now().isoformat()
        }
    },
    "hasAnalysisType": {
        "type": "Property",
        "value": "Translation"
    },
    "hasAnalysisValue": {
        "type": "Property",
        "value": translated_text
    },
    "isAnalysisOf": {
        "type": "Relationship",
        "object": tweet['id']
    },
    "@context": [
        "https://raw.githubusercontent.com/smart-data-models/dataModel.SocialMedia/master/context.jsonld"
    ]
}
```

Figure 23. Creation of translation analysis entity in NGSI-LD.

It is important to note that the current implementation is undergoing modifications due to privacy considerations. These changes may lead to the merging of the translation and sentiment components, resulting in a revised approach where a separate entity solely for translation may no longer be necessary.

4.4.2 Sentiment Analysis Component

To determine the sentiment behind the translated text, we employ the sentiment analysis pipeline from Hugging Face[15], a well-known platform for natural language processing models and libraries. The sentiment analysis pipeline utilizes a pre-trained model that has been fine-tuned on a large dataset to accurately classify sentiments in text[16].

While specific details about the model used may vary depending on the implementation, Hugging Face provides a wide range of sentiment analysis models that can be chosen based on the project's requirements and performance benchmarks[17].

In our sentiment-analysis pipeline, we utilize the default sentiment analyser model provided by Hugging Face for sentiment analysis tasks on English text, namely distilbert-base-uncased-finetuned-sst-2-english[18]. This model is a fine-tune checkpoint of DistilBERT-base-uncased, fine-tuned on SST-2. The model has demonstrated strong performance in sentiment classification

[15] https://huggingface.co/

[16] https://huggingface.co/docs/transformers/main_classes/pipelines

[17] https://huggingface.co/models?pipeline_tag=text-classification&sort=downloads&search=sentiment

[18] https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english

tasks, achieving high accuracy in distinguishing between positive and negative sentiments. The experimental results show the accuracy of 91.3 on the dev set.

```
model_details = sentiment_analyzer.model.config
model_details
```


```
DistilBertConfig {
  "_name_or_path": "distilbert-base-uncased-finetuned-sst-2-english",
  "activation": "gelu",
  "architectures": [
    "DistilBertForSequenceClassification"
  ],
  "attention_dropout": 0.1,
  "dim": 768,
  "dropout": 0.1,
  "finetuning_task": "sst-2",
  "hidden_dim": 3072,
  "id2label": {
    "0": "NEGATIVE",
    "1": "POSITIVE"
  },
  "initializer_range": 0.02,
  "label2id": {
    "NEGATIVE": 0,
    "POSITIVE": 1
  },
  "max_position_embeddings": 512,
  "model_type": "distilbert",
  "n_heads": 12,
  "n_layers": 6,
  "output_past": true,
  "pad_token_id": 0,
  "qa_dropout": 0.1,
  "seq_classif_dropout": 0.2,
  "sinusoidal_pos_embds": false,
  "tie_weights_": true,
  "transformers_version": "4.30.2",
  "vocab_size": 30522
}
```


Figure 24. Sentiment analysis model configuration.

The sentiment analysis pipeline allows us to extract sentiment information from the translated text. It takes care of the model loading and inference processes, simplifying the integration of sentiment analysis into our application. The output includes the sentiment label, which can be either positive or negative, along with a confidence score indicating the model's level of certainty regarding the assigned sentiment.

```
import pandas as pd
from transformers import pipeline

# specific_model = pipeline(model="finiteautomata/bertweet-base-sentiment-analysis")
# specific_model(data)

sentiment_analyzer = pipeline('sentiment-analysis')
```

Figure 25. Code for using Transformers' sentiment analysis pipeline.

In our specific implementation, we focus on capturing only positive and negative sentiments. However, if desired for a particular application, incorporating the neutral sentiment can be achieved by setting a threshold on the confidence level of the sentiment analysis model. By adjusting this threshold, we can fine-tune the classification and include neutral sentiments based on the desired confidence level.

```
translated_text = "The traffic is insane today!"
text_sentiment = sentiment_analyzer(translated_text)
text_sentiment
```


```
[{'label': 'NEGATIVE', 'score': 0.9131529927253723}]
```


Figure 26. Sentiment analysis example.

After obtaining the sentiment label and confidence score for each translated text, we encapsulate this information in the form of an *SMAnalysis* entity, adhering to the NGSI-LD specification from the *SocialMedia* data model.

```python
from datetime import datetime
# Create an Analysis entity to store the translation result
analysis = {
    "id": "urn:ngsi-ld:Analysis:Sentiment123",
    "type": "SMAnalysis",
    "analyzedAt": {
        "type": "Property",
        "value": {
            "@type": "DateTime",
            "@value": datetime.now().isoformat()
        }
    },
    "hasAnalysisType": {
        "type": "Property",
        "value": "Sentiment"
    },
    "hasAnalysisValue": {
        "type": "Property",
        "value": text_sentiment[0]['label']
    },
    "hasConfidence": {
        "type": "Property",
        "value": text_sentiment[0]['score']
    },
    # "isAnalysisOf": {
    #     "type": "Relationship",
    #     "object": tweet['id']
    # },
    "@context": [
        "https://raw.githubusercontent.com/smart-data-models/dataModel.SocialMedia/master/context.jsonld"
    ]
}
```

Figure 27. Creation of SMAnalysis entity in NGSI-LD.

4.5 GAP CALCULATOR (LAND AND CROP CARBON FOOTPRINT)

In the linking phase, we have developed a solution that sets the relationship between *AgriParcel* entities and *AirQualityObserved* entities in order to provide information about the kind of crops that retain more or less CO2 quantities. After linking data, there are cases of *AgriParcel* entities that have not any relationship with *AirQualityObserved* entities (no polygons overlapping). For these cases, the enrichment solution option that have been developed is the "Gap Calculator" (GC). The gaps can be produced due to the kind of the parcel (buildings, roads, watercourses, etc.) or due to errors in the data read from LIDAR that don't provide CO2 records for an specific area. This second option is the one Amper solution will take in account for applying the enrichment procedure by means of gap calculator.

The solution is implemented by means of an algorithm that obtains the neighbour parcels and check the *AirQualityObserved* entities related. So, that getting the carbon values of the parcels with the same kind of crop and using the average of those values, a new *AirQualityObserved* entity would be created and related to the parcel. GC uses the *hasAirQualityObserved* and *location* properties of *AgriParcel* NGSI-LD entity to:

- check whether *AgriParcel* entity has some relationship
- obtain neighbour parcels
- calculate average CO2 value from *AirQualityObserved* related to neighbour *AgriParcel* entities

The GC extracts the NGSI-LD entities from the NGSI-LD Context Broker (i.e. Scorpio) using the corresponding REST queries in its API. This application is written in C# language. The GC process every *AirQualityObserved* related to neighbour *AgriParcel* entities of the same crop type that the parcel under analysis and calculate the average value of CO2. Once this value is calculated, a new *AirQualityObserved* entity is created and injected back to the Scorpio Broker with this CO2 value and using the same location of the *AgriParcel* to be linked with. Furthermore, the *AgriParcel* entity is updated with the new *AirQualityObserved* relationship.

4.6 CITY LIVEABILITY INDEX (CLIFF)

One of the ways of data quality and value augmentation in the DET is proposed through the new concept “City Liveability Index” (CLI) which will be visualized on its application called “City Liveability Index Flexible Frontend” (CLIFF). We shortly refer to this enrichment as CLIFF. On the other hand, this deliverable only focuses on the data enrichment, whereas the frontend aspects are left for the application-related deliverables. The concept of the CLIFF is shortly introduced in [4]. This section includes this introduction as well as concrete enrichment examples for smart cities and selected areas (e.g., urban areas) of the cities.

CLIFF focuses on the quality of data and value augmentation on the cities and its regions. The quality of the data includes the availability of the data for any given area as well as the granularity of the data in the temporal domain. The value augmentation includes implementing “indices”. The indices are calculated by openly defined methods [4] and accepted practices such as the practices of sustainable development. An index of the city combines multiple indices that belong to smaller areas of the cities. For instance, the smaller area can be represented as a polygon and the index can be applied to the smaller area as well as to the complete city.

The “liveability” of the city depends on the indices that may include risk, sustainability, mobility, and citizen engagement. For instance, mobility may include information regarding the traffic situation in the city, and sustainability may include the air quality in the city and its regions.

Figure 28. City Liveability Index applied in different smart cities in Europe. The data sources can be data from sensors that relate to the city environments, as well as open data sources (e.g., documents) depending on the application use case requirements.

Figure 28 illustrates the high-level concept of CLIFF, which consists of gathering data from cities and making them available as “city liveability indices” and publishing them in the EDP through the FIWARE NGSI-LD Scorpio Broker. Having CLIFF from multiple cities would enable comparing different cities through their indices and underlying factors behind the indices can be examined.

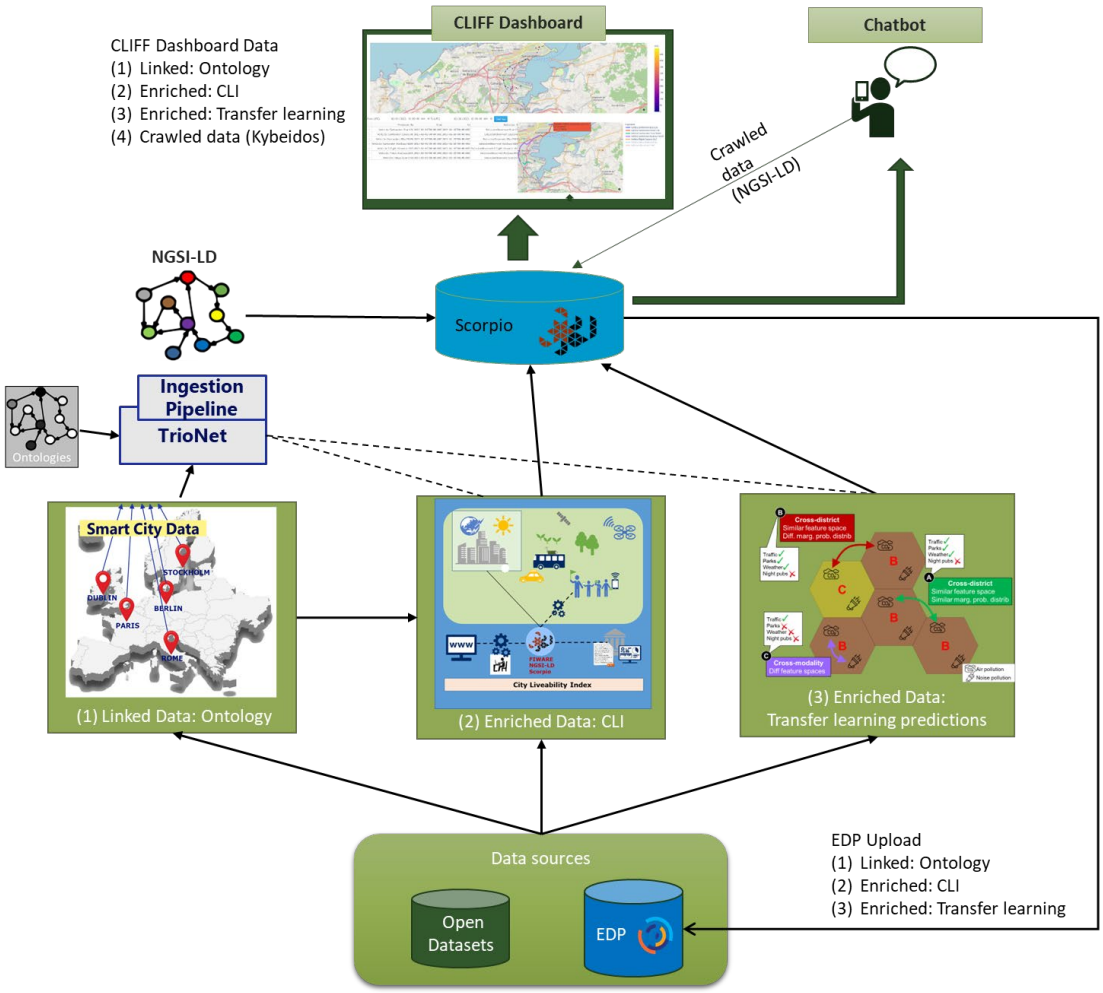


Figure 29. Different ways of linking and enriching the data in CLIFF, starting from the data sources (open sources, EDP) to the Scorpio Broker and frontend visualizations and interactions. The data can be linked in (1) to the ontology and published as NGSI-LD in the Scorpio Broker. The data can be enriched through the city-level or smaller areas in the city, as well as the indices in (2). Lastly, data can be enriched by transfer learning and predictions for the areas with limited data availability. Figure is also included in [4].

Figure 29 (from [4]) illustrates the CLIFF components considering the technical realization based on available datasets such as openly accessible smart city datasets or data from European Data Portal (EDP). These datasets can be linked (on the left side) through a method such as semantic linking by TrioNet and ingested as NGSI-LD data.

Similarly, the value-augment data will be provided by the City Liveability Indices (CLI) that are described above. Lastly, the quality of the data can be improved for the districts where the data are limited or suffering from problems such as low resolution. For those scenarios, techniques such as transfer learning are used to provide accurate predictions (on the right side). The NGSI-

LD data is hosted in Scorpio and accessed by different interfaces such as the flexible frontend and the chatbot (on top). The data can be also made available back in the EDP system.

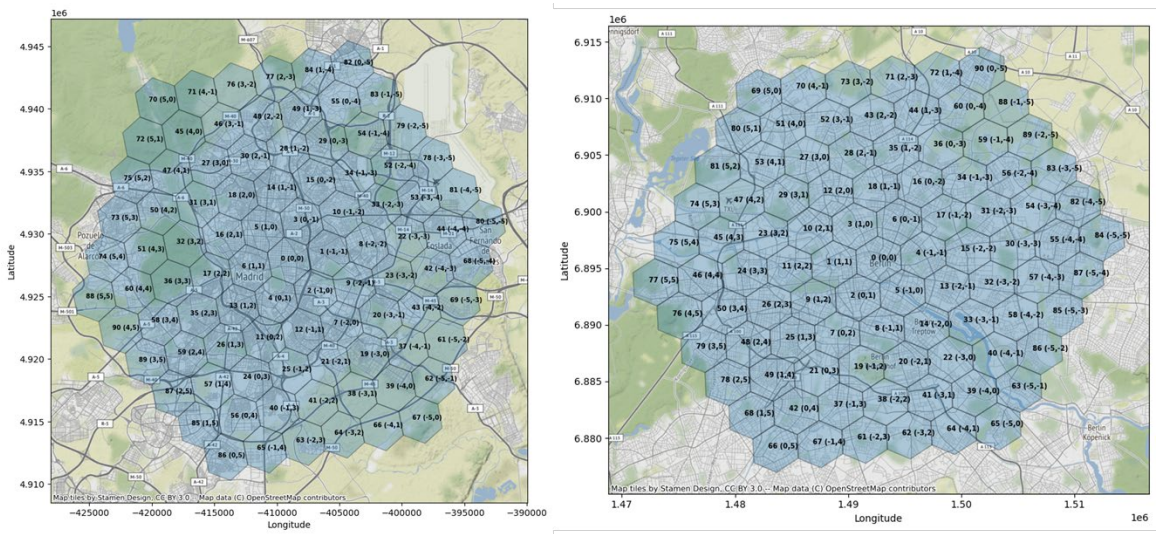


Figure 30. Visualizing the city maps and the smaller areas in the cities for later use in the CLIFF enrichment. Left: Madrid city, right: Berlin city. The cities are divided in the smaller hexagons. The sizes of the hexagons are adjustable with a parameter.

Figure 30 visualizes maps of **the two target cities to realize the CLIFF concept: Madrid and Berlin. The cities are divided into equal-sized hexagons to be able to calculate indices not only on the city level but also in the smaller areas such as city districts. The hexagons are adjustable by their sizes through a parameter.**

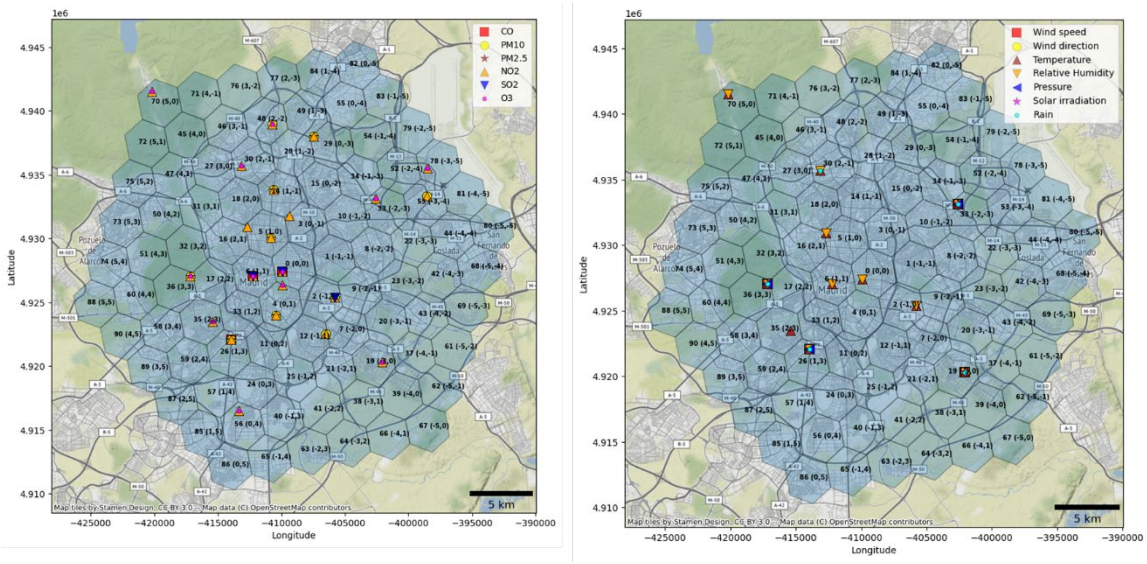


Figure 31. The sensor locations for air quality and weather in the Madrid city area.

Figure 31 visualizes some of the locations of the air quality and weather sensors in the Madrid city area. As can be seen, the sensors do not exist in every hexagon, whereas the information such as wind speed and direction could be applied in the vicinity. Similarly, the air quality data exist from various hexagons, whereas in certain regions it could be interpolated or predicted. The city liveability indices related to air quality and weather can be calculated both in the

hexagon-level and the city-level. Similar calculations can be applied and compared with the city of Berlin.

The enrichment includes the following aspects:

- **Sensor entities:** The mapping of sensor data as "sensor entities" to the NGSI-LD for sharing with the Scorpio Broker and the user interfaces. Each sensor is considered as a separate NGSI-LD entity.
- **Hexagon pre-processed entities:** The mapping of the data from sensors to the relevant hexagons that represent smaller regions in a city. Each hexagon may have various sensor data available. The hexagons are represented as NGSI-LD entities. The preprocessing includes filtering, data cleaning, and interpolation.
- **City liveability index applied on hexagons:** Applying of the city liveability indices to calculate measurable KPIs for the given hexagon area. This step includes the calculation of the indices from the openly accepted standards. Namely, the Sustainable Development Goals-related indices by the SDG portal[19] that can be calculated by the sensor data are chosen.
- **City liveability index applied on cities:** The mapping of the data from hexagons to the entire city based on the indices. This step includes simple aggregation of the indices from the hexagon-level to the entire city.
- **Prediction of missing data:** This step includes training machine learning models from the hexagons where the data and ground-truth exist and transferring the knowledge coming from the machine learning to the hexagons where the information is missing. There are various ways of transfer learning. CLIFF implements a district-similarity-based application of transfer learning.

Dataset name	Description	City
Madrid Air Quality	NO2/NO/CO/CO2/O3 from stations in Madrid every 15 minutes	Madrid
Madrid Traffic	Street-wide density every 15 min	Madrid
Madrid Noise	Noise from different stations	Madrid
Madrid Weather	Temp., wind speed/direction, pressure, RH, irradiation, rain	Madrid
Berlin Weather	Hourly temperature, rain, wind speed/direction, pressure	Berlin
Berlin Air Quality	PM10, PM2.5, NOx, O3, CHB, CHT, CO	Berlin
Berlin Traffic	Street-wide hourly vehicle counts with different vehicle types	Berlin
City Building Data	Building height/type dataset	Berlin, Madrid

Figure 32. Table of datasets related to CLIFF's initial concept realization from Madrid and Berlin.

Figure 32 includes an initial list of datasets that are found to realize the CLIFF application. The datasets include information such as air quality, traffic, and weather information. In addition to those, data from all buildings are collected to estimate population densities.

The SDG-Portal website includes a set of indicators that are based on the UN's Sustainable Development Goals (SDGs). These SDGs provide indicators for cities based on the yearly environmental and socioeconomic data. CLIFF considers [5] as an open standard to apply the

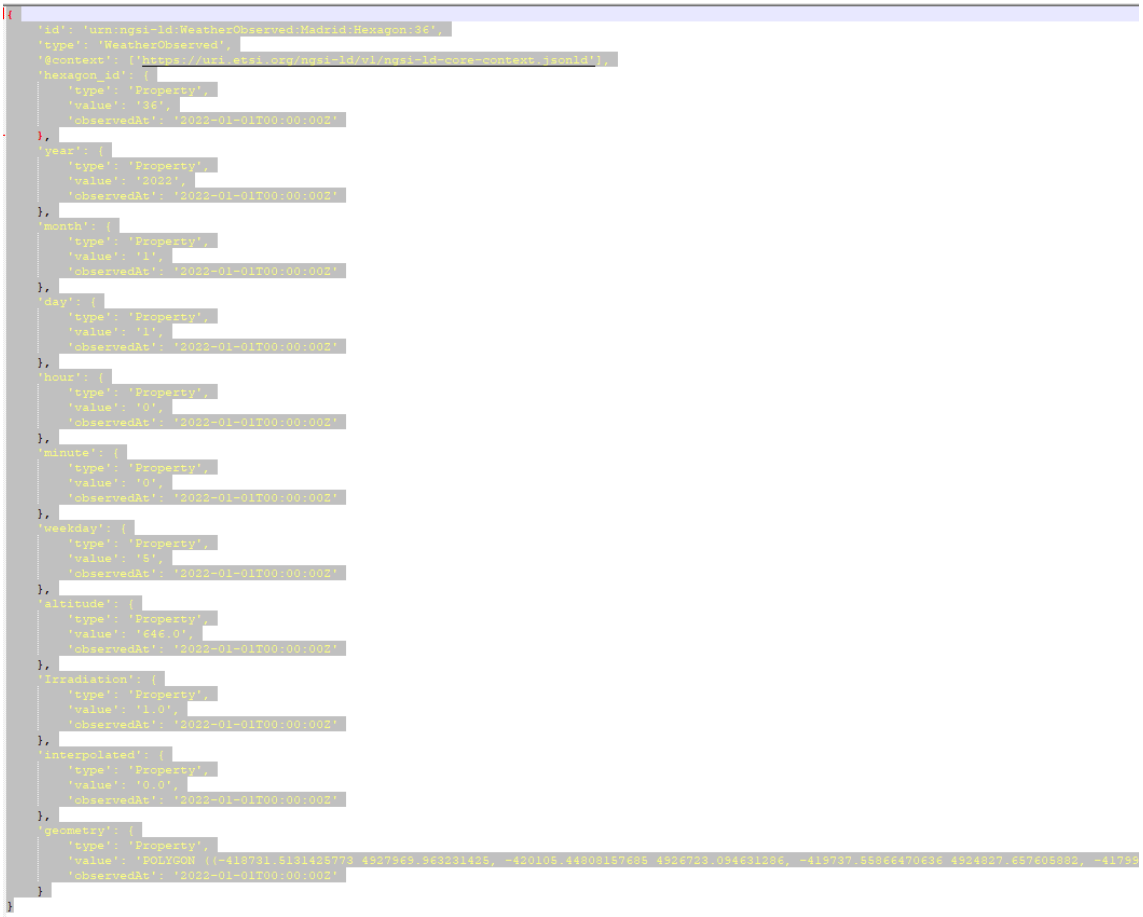
[19] SDG-Portal. https://sdg-portal.de/en/

indices also on higher granularity both in spatial domain (the smaller (higher granularity) areas) and temporal domain, based on sensor data, e.g., hourly timeseries data. As most of the indices are related to the yearly socioeconomic information, we conducted a study on the report [5], where the indices that can be obtained in a higher granularity and the ones that are relevant to the CLIFF datasets are chosen. Out of many SDG metrics, about 10 of them are chosen as feasible for CLIFF. CLIFF aims to implement a few of the most relevant KPIs, such as the ones related to air quality and traffic to the cities of Madrid and Berlin. These indicators can be obtained hourly for every hexagon (hexagon-level indices) as well as for the entire cities (city-level indices). The indices can be easily calculated given the datasets available in the enriched format, available in every hexagon and with higher temporal granularity.

At the end of this section, we include some example NGSI-LD entities based on CLIFF. The example entities that are sensor-based and hexagon-based are coming from real data sources, whereas the entities related to the indices are given as a template format. These example NGSI-LD entities for Air Quality, Weather observations, and Buildings can be found Figures 33-39 both in the forms of sensor entities and hexagon entities, representing smaller regions of the cities. Lastly, the indices are including the relationships such that a city (or an interested district) has a direct relation to the hexagons.

All entities are modelled using the NGSI-LD as well as Smart Data Models[20] for the entity ID and type fields.

[20] https://github.com/smart-data-models



```
{
    'id': 'urn:ngsi-ld:WeatherObserved:Madrid:Hexagon:36',
    'type': 'WeatherObserved',
    '@context': ['https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld'],
    'hexagon_id': {
        'type': 'Property',
        'value': '36',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'year': {
        'type': 'Property',
        'value': '2022',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'month': {
        'type': 'Property',
        'value': '1',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'day': {
        'type': 'Property',
        'value': '1',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'hour': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'minute': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'weekday': {
        'type': 'Property',
        'value': '5',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'altitude': {
        'type': 'Property',
        'value': '646.0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'Irradiation': {
        'type': 'Property',
        'value': '1.0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'interpolated': {
        'type': 'Property',
        'value': '0.0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'geometry': {
        'type': 'Property',
        'value': 'POLYGON ((-418731.5131425773 4927969.963231425, -420105.44809157685 4926723.094631286, -419737.55866470636 4924827.657605882, -41795
        'observedAt': '2022-01-01T00:00:00Z'
    }
}
```


Figure 33. Weather observed for Hexagon 36 in Madrid.

```
{
    'id': 'urn:ngsi-ld:Building:building-Madrid:Hexagon:56',
    'type': 'Building',
    '@context': ['https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld'],
    'building_id': {
        'type': 'Property',
        'value': 'v0.1-ESP.8.1.3.10_1-111'
    },
    'height': {
        'type': 'Property',
        'value': '2.5'
    },
    'age': {
        'type': 'Property',
        'value': '2000.0'
    },
    'building_type': {
        'type': 'Property',
        'value': 'non-residential'
    },
    'id_source': {
        'type': 'Property',
        'value': 'ES.SDGC.BU.000900200VK36F'
    },
    'type_source': {
        'type': 'Property',
        'value': '3_industrial'
    },
    'geometry': {
        'type': 'Property',
        'value': 'POLYGON ((-414530.44149115286 4916181.52627402, -414527.6721902944 4916198.773553342,
    },
    'hexagon_id': {
        'type': 'Property',
        'value': '56'
    },
    'hexagon_x': {
        'type': 'Property',
        'value': '0'
    },
    'hexagon_y': {
        'type': 'Property',
        'value': '4'
    },
    'lat': {
        'type': 'Property',
        'value': '40.344900410360125'
    },
    'location': {
        'type': 'GeoProperty',
        'value': {
            "coordinates": [40.3449, -3.723747],
            "type": "Point"
        }
    },
    'lon': {
        'type': 'Property',
        'value': '-3.7237465718819323'
    }
}
```


Figure 34. A Building NGSI-LD entity from Madrid.

```
{
    'id': 'urn:ngsi-ld:WeatherObserved:MadridIrradiation:ID:24',
    'type': 'WeatherObserved',
    '@context': ['https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld'],
    'station_id': {
        'type': 'Property',
        'value': '24',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'geometry': {
        'type': 'Property',
        'value': 'POINT (-417152.48156697437 4927070.647350454)',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'year': {
        'type': 'Property',
        'value': '2022',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'month': {
        'type': 'Property',
        'value': '1',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'day': {
        'type': 'Property',
        'value': '1',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'hour': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'minute': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'weekday': {
        'type': 'Property',
        'value': '5',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'altitude': {
        'type': 'Property',
        'value': '646.0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'lat': {
        'type': 'Property',
        'value': '40.41254442249261',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'location': {
        'type': 'GeoProperty',
        'value': {
            "coordinates": [40.412544, -3.758228],
            "type": "Point"
        },
        'observedAt': '2022-01-01T00:00:00Z'
    }
```


Figure 35. Weather sensor for irradiation in Madrid.

```
{
    'id': 'urn:ngsi-ld:AirQualityObserved:Madrid:NO2:ID:8',
    'type': 'AirQualityObserved',
    '@context': ['https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld'],
    'station_id': {
        'type': 'Property',
        'value': '8.0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'hexagon_id': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'altitude': {
        'type': 'Property',
        'value': '672.0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'year': {
        'type': 'Property',
        'value': '2022',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'month': {
        'type': 'Property',
        'value': '1',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'day': {
        'type': 'Property',
        'value': '1',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'hour': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'minute': {
        'type': 'Property',
        'value': '0',
        'observedAt': '2022-01-01T00:00:00Z'
    }
    'NO2': {
        'type': 'Property',
        'value': '35.0',
        'observedAt': '2022-01-01T00:00:00Z'
    }
    'location': {
        'type': 'GeoProperty',
        'value': {
           "coordinates": [40.421553, -3.682316],
           "type": "Point"
        },
        'observedAt': '2022-01-01T00:00:00Z'
    }
```


Figure 36. Air quality NO2 sensor entity with ID:8 from Madrid.

```
{
    'id': 'urn:ngsi-ld:AirQualityIndex:Madrid:section4',
    'type': 'CityLivabilityIndex',
    '@context': ['https://fiware.github.io/data-models/context.jsonld'],
    'location': {
        'type': 'GeoProperty',
        'value': {
            'type': 'Point',
            'coordinates': [-417152.48156697437, 4927070.647350454]
        },
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'AirQualityIndex': {
        'type': 'Property',
        'value': 7,
        'used formula': {
            'type': 'Property',
            'value': 'no2 * area' #some real formula
        'providedBy': [
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityObserved:Madrid:hexagon1'
            },
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityObserved:Madrid:hexagon2'
            },
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityObserved:Madrid:hexagon3'
            }
        ]
    }
}
```

Figure 37. Air quality index of Madrid district 4 that consists of 3 hexagons: Hexagon 1, 2, and 3. (template example)

```
{
    'id': 'urn:ngsi-ld:citylivablityindex:Madrid',
    'type': 'CityLivabilityIndex',
    '@context': ['https://fiware.github.io/data-models/context.jsonld'],
    'location': {
        'type': 'GeoProperty',
        'value': {
            'type': 'Point',
            'coordinates': [-417152.48156697437, 4927070.647350454]
        },
        'observedAt': '2022-01-01T00:00:00Z'
    },
    'AirQualityIndex': {
        'type': 'Property',
        'value': 7,
        'providedBy': [
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityIndex:Madrid:section4'
            },
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityIndex:Madrid:section9'
            },
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityIndex:Madrid:section3'
            }
        ]
    },
    'PublicTransport': {
        'type': 'Property',
        'value': 4.7,
        'providedBy': [
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityObserved:Madrid:Trainstation:36'
            },
            {
                'type': 'Relationship',
                'object': 'urn:ngsi-ld:AirQualityObserved:Madrid:Busstop:37'
            }
        ]
    }
    '
}
```

Figure 38. Liveability indices of Madrid in terms of Air Quality and Traffic (e.g., Public Transportation).

4.7 APPLICATION-AGNOSTIC ENRICHERS

In SALTED, we have not only envisioned Data Enrichers that work towards a specifical goal, but also those that follow a horizontal approach. This means that the following Enrichers to be described in this section can be applied to several of the project's applications and use cases, or even to any kind of data modelled as Smart Data Models. The reasoning behind the implementation of these kinds of Enrichers is to provide domain-agnostic data quality enhancement, which may be used even outside of the scope of the project.

4.7.1 Reverse Geocoding Data Enricher

The Reverse Geocoding Data Enricher (RGDE) is able to enrich any NGSI-LD entity that features a *location* property. The use of this property is widely extended, to the point where all Smart Data Models include it by default, since it allows to pinpoint the exact coordinates of an object or event represented by an entity.

The RGDE uses the *location* property to obtain precise information about the position of an entity. This information may include the country, the region, the city, the postal code and/or the street number; and will do so if applicable. The operation that obtains this knowledge from a set of coordinates is known as reverse geocoding.

To obtain these values, the RGDE makes use of the third-party open-source service Nominatim[21], which itself relies on OpenStreetMap data.

The RGDE extracts the NGSI-LD entities directly from the NGSI-LD Context Broker (i.e. Scorpio), combining queries and subscriptions. It is written in Python, and uses the *geopy.geocoders* library for communication with the Nominatim API, the *json* library for data handling, the *datetime* and *pytz* libraries for traceability, and the *flask* and *waitress* libraries to set up the REST server for notifications.

Every NGSI-LD entity is processed individually. The RGDE reads the *location* value of the entity and performs a reverse geocoding request to the Nominatim API. The newly acquired information is then mapped into the *address* and *areaServed* properties that can be found by default in most Smart Data Models. The enriched entity, which now includes both new properties, is reinjected into the Scorpio Broker via an HTTP POST request to the */entityOperations/upsert* endpoint. An example of these properties included in a sample entity is shown in Figure 39.

[21] https://nominatim.org/

```
"address": {
    "type": "Property",
    "value": {
        "addressCountry": "España",
        "addressLocality": "Santander",
        "addressRegion": "Cantabria",
        "postalCode": "39001",
        "streetAddress": "Calle de Salvador Hedilla"
    },
    "observedAt": "2023-05-25T10:08:56.433950Z"
},
"areaServed": {
    "type": "Property",
    "value": "1, Calle de Salvador Hedilla, Correos, Santander, Cantabria, 39001, España",
    "observedAt": "2023-05-25T10:08:56.433950Z"
},
```


Figure 39: Extra properties added by the RGDE

The RGDE is deployed in an Ubuntu 20.04.5 LTS Virtual Machine, inside a Docker container, and we perform a periodic health check on it to make sure it is up and running.

4.7.2 Insight Data Enricher

The Insight Data Enricher (IDE) can enrich certain NGSI-LD entities according to the Smart Data Model they represent. Certain models have properties that consist of a numerical value and, if needed, metadata such as a unit or a timestamp. In some cases, these numerical values have a non-intuitive meaning, which makes them inconvenient for human reading and comprehension. For instance, an air quality measurement may present the user with an *no2* value of *67.7* and unit code *GP*. This has a high chance to not be understood by the end user.

The IDE adds new metadata to these kinds of properties in order to make life easier for the users. This is implemented as a string-typed sub-property that provides a short description of the value being enriched. In the aforementioned example, the string *"dangerous"* is added to the *67.7* value to explain in simple terms that it is not a safe level of *no2*.

The thresholds that separate two adjacent levels (e.g. *safe* versus *unsafe*) have been chosen according to well-known, respected sources. For example, air quality thresholds rely on guidelines published by the World Health Organization, like [6]. These sources are also added as metadata for traceability.

The IDE extracts the NGSI-LD entities directly from the Scorpio Broker, combining queries and subscriptions. It is written in Python, and it uses the *json* library for data handling, the *datetime* and *pytz* libraries for traceability, and the *flask* and *waitress* libraries to set up the REST server for notifications.

Every NGSI-LD entity, and in fact every eligible property, is processed individually. The IDE reads the value of the property and compares it to the pre-defined thresholds. The new sub-property is generated and added to the original value. The enriched entity, which now includes one or more new sub-properties, is reinjected into the Scorpio Broker via an HTTP POST request to the */entityOperations/upsert* endpoint. An example of these properties included in a sample entity is shown in Figure 40Figure 40.

```
"co": {
    "type": "Property",
    "value": 0.1,
    "observedAt": "2023-06-19T14:31:40Z",
    "unitCode": "GP",
    "concentrationLevel": {
        "type": "Property",
        "source": {
            "type": "Property",
            "value": "https://apps.who.int/iris/handle/10665/345329"
        },
        "value": "Safe",
        "observedAt": "2023-06-19T14:31:57.655634Z"
    }
},
"no2": {
    "type": "Property",
    "value": 115.0,
    "observedAt": "2023-06-19T14:31:40Z",
    "unitCode": "GQ",
    "concentrationLevel": {
        "type": "Property",
        "source": {
            "type": "Property",
            "value": "https://apps.who.int/iris/handle/10665/345329"
        },
        "value": "Dangerous",
        "observedAt": "2023-06-19T14:31:57.655634Z"
    }
},
```


Figure 40: Properties with extra sub-properties added by the IDE

The IDE currently supports the *AirQualityObserved*, *SoundPressureLevel*, *Temperature*, and *TrafficFlowObserved* Smart Data Models. It is deployed in an Ubuntu 20.04.5 LTS Virtual Machine, inside a Docker container, and we perform a periodic health check on it to make sure it is up and running.

5 CONCLUSIONS

In this document, we presented the D2.3 deliverable on the SALTED situation-aware data enrichment, as well as the current implementation and deployment of the architecture. The scope of the document therefor encompasses updates on Architecture, Security, Orchestration, DET Injection Chains, and a description of the DET Enrichers developed until now.

After the introduction in section one the second section updates on the architecture's security and orchestration aspects, highlighting the development of the helper python packages (*TokenHandler* and *ControlLoopHandler*) for easier integration within the partners' pipelines. Further an exemplary implementation of an orchestration pipeline is explained using the Agenda Analytics use case. The third section provides updates on the DET Injection Chains. Specifically, it outlines new data models used (namely *DataService* & *DataServiceRun* and *Distribution*) and presents updates on the IoT Data Injection Chain (Madrid Air Pollution and Traffic datasets), the Social Media Injection Chain and the Agricultural Injection Chain, overall demonstrating the architecture's versatility in handling different data sources.

The DET Enrichers section introduces various components, such as Crawling Enricher and Agenda Matching Enricher from Agenda Analytics, Translation and Sentiment Analysis Enricher from Traffic Sentiment App, Gap Calculator from Crops & CO2 App, and City Liveability Index from the CLIFF App. Additionally, Application-Agnostic Enrichers, including Reverse Geocoding Data Enricher and Insight Data Enricher, are discussed.

Overall, this deliverable showcases the advancement of the SALTED architecture and its DET components, supporting the project's objectives of efficient data modelling, linking, and enriching for meaningful insights. Especially the enrichers play a crucial role in advancing the development of the projects regarding the end-user applications, which showcase the true value of the SALTED infrastructure for data-driven applications.

6 BIBLIOGRAPHY

[1] SALTED, “D2.1: Report on Data Linking and Enrichment Architecture,” 2022.

[2] SALTED, “D2.2:Report on data modelling and linking,” 2023.

[3] SALTED, “D1.1: Report on semantic features extraction for heterogeneous data sources contextualization,” 2022.

[4] e. a. Luis Sanchez, “Data Enrichment Toolchain: A Data Linking and Enrichment Platform for Heterogeneous Data,” *under submission,* 2023.

[5] B. S. E. A. (HRSG.), “SDG-Indikatoren für Kommunen, 3. Auflage,” 2022. [Online]. Available: https://www.bertelsmann-stiftung.de/de/publikationen/publikation/did/sdg-indikatoren-fuer-kommunen-all-1. [Accessed 10 7 2023].

[6] W. H. Organization, “WHO global air quality guidelines: particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide,” 2021.