ESSnet Trusted Smart Statistics – Web Intelligence Network

Grant Agreement Number: 101035829 — 2020-PL-SmartStat

WP2: OJA and OBEC Software

Deliverable 2.6: Technical guidelines on the implementation of the scripts produced for OJA and OBEC statistics

Final version, 2025-03-28

Prepared by:

WP leader: Jacek Maślankowski (Statistics Poland, Poland, j.maslankowski@stat.gov.pl)

Contributors:

OBEC:

Ewelina Niewiadomska, Heidi Kuehnemann, Vilma Nekrašaitė-Liegė, Marta Sobieraj, Donato Summa, Kostadin Georgiev, Črt Grahonja, Jacek Maślankowski, Michał Bis, Johannes Gussenbauer

OJA:

Žydrūnas Eisinas, Johannes Gussenbauer, Alberto Columbano, Dominik D.P. Blatt, Nuška Brnot, Matej Divjak, Martine M.T. de Mooij-Schep, Erika Cerasti , Angela Pappagallo, Francesco Amato, Francesca Inglese, Giuseppina Ruocco, Galia Stateva, Kostadin Georgiev, Annalisa Lucarelli, Jacek Maślankowski, Pierre Villedieu, Yannis Bouachera, Renato Magistro, Giulio Massacci

Web Intelligence Network

Funded by the European Union

This document was funded by the European Union.

Contents

1. Introduction 4
2. Prerequisites 5
   2.1. OJA 5
   2.2. OBEC 5
3. Writing scripts for OJA 7
   3.1. Description of the datasets 7
   3.2. Preparation of the environment in the Datalab 9
   3.3. Connecting to Athena 11
   3.4. List all tables for OJA 12
   3.5. Typical query 13
   3.6. How to get a sample of rows from the latest table 14
   3.7. Getting data by countries 14
   3.8. Total number of unique OJAs 15
   3.9. Getting data for the first active year 15
   3.10. Getting OJAs by countries for the first active year and skills at hierarchy 0 16
   3.11. Calculate number of job ads by countries in the first quarter of 2023 17
   3.12. The script used to generate tables 17
4. OBEC Use case 19
   4.1. General outline on the statistical production 19
   4.2. Technical manual on the OBEC use case step-by-step 21

List of figures 32

List of tables 32

References 32

1. Introduction

The purpose of this guideline is to demonstrate the technical feasibility of accessing the resources of the Web Intelligence Hub (WIH) to execute scripts developed by members of the Web Intelligence Network (WIN). The guideline includes two specific use cases: Online-Based Enterprise Characteristics (OBEC) and Online Job Advertisements (OJA).

Online-Based Enterprise Characteristics (OBEC) is a concept that originated in 2016. It refers to the utilization of digital platforms, including websites and social media, by enterprises to showcase their business activities. OBEC encompasses not only the presence of a website hosted on servers owned or operated by the enterprise itself, but also includes websites hosted by third parties, such as those managed by parent companies or related enterprises within a corporate group. The OBEC use case focuses on enterprises that maintain a website and have a workforce of 10 or more employees.

Mostly it was on to support ICT Usage in Enterprises, which is a ESS survey. The population is strongly related to the question from this survey, i.e.:

Methodological manual for data compilers and users of the ICT survey:

A4. Does your enterprise have a website?

[Scope: enterprises with access to the internet, i.e. A1 > 0],

[Type: single answer (i.e. Tick only one); binary (Yes/No); filter question].

Online Job Advertisements (OJA) refers to data collected from various online job portals across European countries. This data is standardized and harmonized to enable the generation of comparable results across all countries using a single query.

2. Prerequisites

2.1. OJA

To work with Online Job Advertisements (OJA), it is essential to use Python or R for data access, and a knowledge of SQL is required for direct database interactions. This technical guideline focuses on Python, as it has been utilized across all use cases within the Web Intelligence Network. Access to the following resources is necessary for working with OJA data:

- **OJA Datalab**: https://portal.eurostat.datalab.ecdp.tech.ec.europa.eu/

  The following technical skills are required for effective data processing:

- **Python**: For scripting and data analysis.
- **SQL (Athena)**: For querying and managing data within the database.
- **Linux**: For operating system-level data management and processing tasks.

2.2. OBEC

To work with Online-Based Enterprise Characteristics (OBEC), access to the following resources is required:

- **Web Intelligence Hub Platform (WIHP, WIP) – Data Acquisition Service (DAS)**: https://prod.wihp.ecdp.tech.ec.europa.eu/
- **Datalab**: https://portal.eurostat.datalab.ecdp.tech.ec.europa.eu/

To execute scripts for OBEC analysis, the following files are necessary:

- A **JSON file** containing the URLs that need to be examined.
- The **WIN_OBEC Python package**, provided by Work Package 2.

For effective linkage of the results to the Business Register, the JSON file with URLs should include two datasets with the following attributes:

- **For Web Intelligence Hub Platform – Data Acquisition Service (web scraping tool with OpenSearch):**
  - *Anonymized ID*: A unique identifier for each enterprise.
  - *Enterprise URL*: The website address of the enterprise.
  - *Group*: The group, such as "OBEC", which is the group to which the user belongs and have at least developer rights.

- **For further processing in Datalab (JupyterLab with Python, RStudio):**
    - *Business Register ID*: A unique identifier linking the enterprise to the business register.
    - *Anonymized ID*: To maintain consistency across datasets.
    - *Other attributes*: Additional information needed for analysis.

The following technical skills are required for data processing:

- **Python**: For scripting, data analysis, and processing.
- **NoSQL (OpenSearch)**: For querying and managing data extracted from the WIP-DAS.
- **Linux**: For managing the data environment and performing server-side operations.

3. Writing scripts for OJA

3.1. Description of the datasets

The OJA DataLab is a system that comprises several tables, with the most important being the table containing the final, processed data derived from the raw data. Below is the list of key attributes available in this table.

Table 1. Attributes available in processed data in the OJA DataLab

Column name	Data type	Description
oja_id	object	Internal ID of the online job offer
first_active_date	object	Date of posting the offer
first_active_day	int64	The day the offer was posted
first_active_month	int64	The month the offer was placed
first_active_year	int64	The year the offer was placed
last_active_date	object	Offer end date
last_active_year	int64	Offer end date
last_active_month	int64	The month the offer ends
last_active_day	int64	Offer end year
language	object	Language of the offer
occupation4d_id	object	ISCO 4-character code
occupation3d_id	object	ISCO 3-character code
occupation2d_id	object	ISCO 2-character code
occupation1d_id	object	ISCO 1-character code
occupation4d	object	ISCO Level 4 name
occupation3d	object	ISCO Level 3 name
occupation2d	object	ISCO Level 2 name
occupation1d	object	ISCO Level 1 name
skill_id	object	Skill ID
skill	object	Skill name
skill_hier3_id	object	Skill code, level 3
skill_hier3	object	Skill name, level 3
skill_hier2_id	object	Skill code, level 2
skill_hier2	object	Skill name, level 2
skill_hier1_id	object	Skill code, level 1
skill_hier1	object	Skill name, level 1
skill_hier0_id	object	Skill code, level 0
skill_hier0	object	Skill name, level 0
city_id	object	Location identifier
city	object	Town
nuts3_id	object	Code NUTS3
nuts3	object	Name NUTS3

nuts2_id	object	Code NUTS2
nuts2	object	Name NUTS2
nuts1_id	object	Code NUTS1
nuts1	object	Name NUTS1
country_id	object	Two-character country identifier according to ISO-3166
country	object	Country name
contract_id	object	Contract ID
contract	object	Contract name
education_id	object	Educational level identifier according to ISCED-11
education	object	Educational level according to ISCED-11
economic_activity2d_id	object	Economic activity code (activity classification) of the entity posting the offer, level 2
economic_activity2d	object	Economic activity (activity classification) of the entity posting the offer, level 2
economic_activity1d_id	object	Economic activity code (activity classification) of the entity posting the offer, level 1
economic_activity1d	object	Economic activity (activity classification) of the entity posting the offer, level 1
activity_sector_id	object	Code of the activity sector of the unit inhabiting the offer
activity_sector	object	Name of the sector of activity of the unit inhabiting the offer
salary_id	object	Pay Level ID
salary	object	Salary level
working_time_id	object	Job size identifier
working_time	object	Name of the job size
experience_id	object	Required Experience ID
experience	object	Name of required experience
source_id	object	ID of the source from which the offer was obtained
source_category	object	Category of the source from which the offer was obtained
source_category_id	object	Category identifier of the source from which the offer was obtained
source_country	object	The country in which the offer was published
source_stability	object	The stability level of the data source

Many of the attributes described above are derived from detection algorithms implemented within the OJA processing pipeline, leveraging machine learning techniques. These include classifications such as occupation (ISCO), educational attainment (ISCED), economic activity (NACE), among others. The underlying assumption is that, in most instances, these algorithms accurately identify and classify the relevant data within the OJA pipeline.

Data access is facilitated through Python or R environments, using SQL queries executed within the Amazon Web Services (AWS) cloud computing environment. The backend database is Amazon Athena, requiring that all queries conform to Athena's SQL syntax.

As detailed in the country-specific subchapters below, maintaining high data quality is crucial. An annotation exercise conducted in 2022 revealed that the statistics produced by the OJA DataLab may not yet meet the standards required for publication as official statistics. In some instances, however, limited aggregations of the data can be used to present findings as *experimental statistics*. A final decision regarding the use of OJA data as official statistics will depend on the outcomes of the 2024 OJA annotation exercise, in agreement with member countries of the Web Intelligence Network (WIN).

The following sections detail the specific applications of OJA data by individual countries within WIN, while also highlighting commonalities among them. For example, France, Lithuania, Slovenia, and the Netherlands have developed in-house systems for generating statistics from OJA data, leading them to focus on how WIN OJA data might complement their existing datasets. Austria and France provided comparative tables that align DataLab OJA results with their official Job Vacancy Statistics, and a similar analysis was carried out in Poland. In Poland's case, the OJA data displayed greater variability and, at times, diverged from the trends observed in official labor market statistics. Notably, Italy, Slovenia, and Bulgaria collaborated on a joint initiative to produce a set of indicators using a standardized methodology, enabling the presentation of comparable results across these countries.

3.2. Preparation of the environment in the Datalab

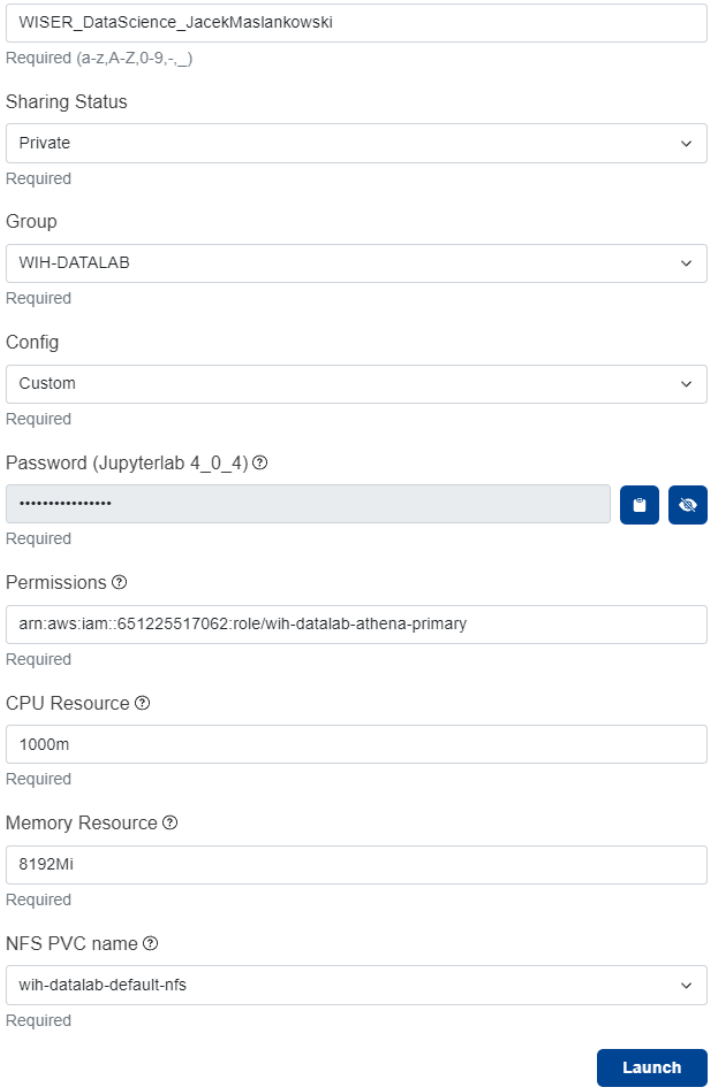
To begin working with the OJA DataLab, it is first necessary to establish a connection by logging in through the following URL:

- **OJA DataLab Login**: https://portal.eurostat.datalab.ecdp.tech.ec.europa.eu/

It is important to note that this URL may change as the DataLab platform evolves. Currently, users can log in through the EU login using either a QR code or a PIN. Since this is a standard login method for European Union services, we will not delve into the specifics here. However, users must contact the Web Intelligence Hub (WIH) authorities to obtain access credentials for the DataLab.

The initial parameters required for setting up the environment are provided in Figure 1 (not included here). These parameters are essential for configuring the data access environment, enabling users to interact with the OJA datasets once access is granted.

Figure 1. Parameters used in configuration of the OJA Datalab



To set up the environment in the OJA DataLab, follow these steps:

1. **Name of the Datalab**:
    - While there are no strict naming conventions, it is recommended to use a name that is easily associated with your organization or the country creating the environment. This helps in identifying the workspace more effectively.
2. **Sharing Status**:
    - **Private**: Only the creator (single user) has access to this environment.
    - **Public**: Allows sharing of the environment with other users.

    For most users working with OJA data within the Web Intelligence Hub (WIH), it is advisable to set the group to **WIH-DATALAB**.

3. **Configuration**:
    - Set the **config** to **custom**. This means that the environment's resources, such as CPU and memory allocation, can be tailored to specific needs.
4. **Password**:
    - A password will be generated automatically during the setup process, which ensures secure access.
5. **Permissions**:
    - These should be left unchanged and must match the permissions shown in the reference figure. This ensures that the environment operates correctly and remains aligned with the access requirements of the WIH.
6. **CPU Resource Configuration**:
    - This parameter defines how many CPU resources will be allocated. For example, setting **1000m** indicates that one full CPU will be utilized.
7. **Memory Resource**:
    - It is recommended to allocate a memory resource of no more than **8 GB**. This is specified as **8192Mi** in the configuration.
8. **NFS PVC Name**:
    - Select the **NFS PVC (Persistent Volume Claim) name** from the provided list. For users in this setup, it should be **wih-datalab-default-nfs**.

These settings ensure that your Datalab environment is correctly configured for accessing and analyzing OJA data, while also maintaining compatibility with the Web Intelligence Network's standards.

3.3. Connecting to Athena

To access the OJA data from the SQL database using Python, you can use the following script in a new Jupyter notebook within the DataLab environment. This script sets up a connection to the Athena database and allows you to query the data.

```
#!pip install pyathena

from pyathena import connect

from pyathena.pandas.util import as_pandas

cursor = connect(
```

```
    region_name="eu-west-1",
    s3_staging_dir="s3://wih.aws-athena-query-results.eu-west-
1/",
    work_group="primary"
    ).cursor()
cursor.execute("""
            SHOW TABLES IN AwsDataCatalog.wih_oja_latest;
            """, {'param': 'a string'})
tables = as_pandas(cursor)
tables
```

To begin using the pyathena package for accessing data from the OJA DataLab's Athena database, you will first need to install the package. Here's how you can do that, followed by an updated script that connects to the environment and lists all tables from the wih_oja_latest schema, which is the recommended schema for users to access.

1. **Step 1: Install the pyathena Package**

In your Jupyter notebook, you can install the pyathena package by running the following command:

pip install pyathena

2. **Step 2: Connect to the Athena Environment and List Tables**

After installing the package, you can use the following script to connect to the Athena environment and list all tables in the wih_oja_latest schema:

3.4. List all tables for OJA

To always retrieve the most recent list of tables in the wih_oja_latest schema, you can execute the following SQL command:

SHOW TABLES IN wih_oja_latest;

Explanation:

- **Command**: SHOW TABLES IN wih_oja_latest;
    - This SQL command instructs Athena to list all tables currently available within the specified schema (wih_oja_latest).

- Executing this command will provide an up-to-date view of all tables, ensuring you have the latest information.

```
cursor.execute("""
          SHOW TABLES IN AwsDataCatalog.wih_oja_latest;
          """, {'param': 'a string'})
tables = as_pandas(cursor)
tables
```

The aforementioned command executes an SQL query that retrieves the list of tables from the specified schema and stores the results in a Pandas DataFrame. Therefore, it is essential to import the Pandas package prior to executing this command, as it provides the necessary functionality for data manipulation and storage within the Python environment.

3.5. Typical query

A typical query executed within the environment pertains to the wih_oja_latest schema, specifically targeting the wih_oja_blended table. This query is illustrated in the code provided below.

```
# 0. FIRST QUERY
# latest dataset SCHEMA: wih_oja_latest
# all the variables available in the table (i.e. blending all
the information available - "wih_oja_blended")
cursor.execute("""
     SELECT *
     FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended
     LIMIT 1;
    """, {'param': 'a string'})
documents = as_pandas(cursor)
column_names = documents.columns.values.tolist()
column_names
```

The query presented above will retrieve all columns available within the wih_oja_blended table. A comprehensive list of these columns has been previously detailed in the preceding subchapter of this guideline.

3.6. How to get a sample of rows from the latest table

A prevalent example of a query involves retrieving data directly from the wih_oja_blended table. The example provided below demonstrates the process of extracting this data and storing it within a Pandas DataFrame for subsequent analysis and manipulation.

```
# 1. Get a sample of rows for all variables

cursor.execute("""

    SELECT *

    FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended LIMIT
1000;

    """, {'param': 'a string'})


my_sample = as_pandas(cursor)

my_sample
```

As previously indicated, not all data is stored in the DataFrame due to the inclusion of the LIMIT 1000 clause in the SQL query. It is advisable to avoid executing SQL queries that yield outputs comprising tens of thousands of rows, as this may result in excessive memory consumption. Querying the whole table can create large cost, that can be avoided by the LIMIT clause.

3.7. Getting data by countries

To retrieve data categorized by country, it is essential to utilize the country attribute, which is a designated column within the wih_oja_blended table. A typical query that lists occupations along with their corresponding country names is presented below.

```
# 2. FIRST PRACTICAL QUERY

cursor.execute("""

    SELECT distinct occupation4d, country, country_id

    FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended LIMIT
10;

    """, {'param': 'a string'})

documents = as_pandas(cursor)

documents
```

The aforementioned table contains two columns pertinent to country identification: one is designated as country, and the other as country_id. The country_id column utilizes a two-character code to represent each country; for instance, "PL" corresponds to Poland, "DE" represents Germany, and "IT" signifies Italy, among others. The query provided above will yield a list of unique occupations associated with the countries in which these occupations were advertised.

3.8. Total number of unique OJAs

The script employed to calculate the total number of unique objects is presented above. It is important to note that the query must consistently utilize the DISTINCT keyword for the oja_id column. This column is not inherently unique, as it is utilized to represent multiple skills associated with each occupation.

```
# calculate number of unique OJA ids

cursor.execute("""

     SELECT country, COUNT(DISTINCT oja_id)
     FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended
     GROUP BY country LIMIT 100;

    """, {'param': 'a string'})

nr_ads  = as_pandas(cursor)

print(nr_ads)
```

The query presented above can be employed to compute the total number of unique identifiers within the dataset.

3.9. Getting data for the first active year

A critical component of the queries is the date-related WHERE condition. To filter the data based on specific dates, it is necessary to utilize attributes such as first_active_date, first_active_month, last_active_year, last_active_month, and last_active_day. The query presented below demonstrates the methodology for determining the number of job advertisements that have been published according to the year of their initial activation.

```
# calculate number of unique OJA ids grouped by
first_active_year

cursor.execute("""
```

```
    SELECT COUNT(DISTINCT oja_id) AS ads, first_active_year
    FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended
    GROUP BY(first_active_year) LIMIT 10;
   """, {'param': 'a string'})
nr_ads  = as_pandas(cursor)
nr_ads
```

As indicated above, the results of the query will be restricted to the first ten rows. While we have enumerated all job advertisements based on their initial activation year, the corresponding last active year remains unspecified.

3.10. Getting OJAs by countries for the first active year and skills at hierarchy 0

The tables recommended by the Web Intelligence Network include references to selected occupations and territorial aggregations. Furthermore, it is possible to extract skills associated with specific occupations. The query presented below will display the relevant skills along with their corresponding first active year.

```
# 6. No duplicates - use distinct
cursor.execute("""
    SELECT COUNT(DISTINCT oja_id) AS ads, first_active_year, skill_hier0
    FROM      AwsDataCatalog.wih_oja_latest.wih_oja_blended
    GROUP BY(first_active_year, working_time, skill_hier0)
    LIMIT 1000000;
   """, {'param': 'a string'})
nr_ads_by_year_worktime_skill = as_pandas(cursor)
nr_ads_by_year_worktime_skill
```

However, the utilization of skills is not recommended by the Web Intelligence Network, as it has not been possible to verify all identified skills. It is important to recognize that a machine learning algorithm is employed for skill identification, which may result in inaccuracies in the identification process, leading to the possibility that certain skills are not correctly recognized.

3.11. Calculate number of job ads by countries in the first quarter of 2023

To compute the number of job advertisements by country for the first quarter of 2023, the query provided below can be utilized.

```
cursor.execute("""
    SELECT country, count(distinct oja_id) as number_of_ojas
    FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended
    WHERE (first_active_month in (1,2,3) or last_active_month
in (1,2,3))
    AND (first_active_year=2023 or last_active_year=2023)
    GROUP BY country ORDER BY 2 LIMIT 100;
    """, {'param': 'a string'})
documents = as_pandas(cursor)
documents
```

The query presented above will display the number of job advertisements categorized by country, while restricting the results to the first three months of 2023, thereby representing the first quarter of the year.

3.12. The script used to generate tables

Finally, the scripts employed to generate the tables recommended by the Web Intelligence Network align with the guidelines outlined in Deliverable 2.4. This deliverable specifically advocates for the listing of the most accurate occupations categorized by territorial disaggregation.

```
cursor.execute("""
    SELECT country, nuts3, occupation4d, count(distinct
oja_id) as number_of_ojas
    FROM AwsDataCatalog.wih_oja_latest.wih_oja_blended
    WHERE (first_active_month in (1,2,3) or last_active_month
in (1,2,3))
    AND (first_active_year=2024 or last_active_year=2024)
```

```
    AND occupation4d_id in

    ('OC2113')

    GROUP BY country, nuts3, occupation4d ORDER BY 1,2,3 LIMIT
10;

    """, {'param': 'a string'})

documents = as_pandas(cursor)

documents
```

The query presented above will reveal the number of job advertisements that occurred in the first quarter of 2024, categorized by country, NUTS-3 region, and occupation. An illustrative example of the resultant data is provided in the table below.

Table 2. Result of the suggested query to present the final set of tables

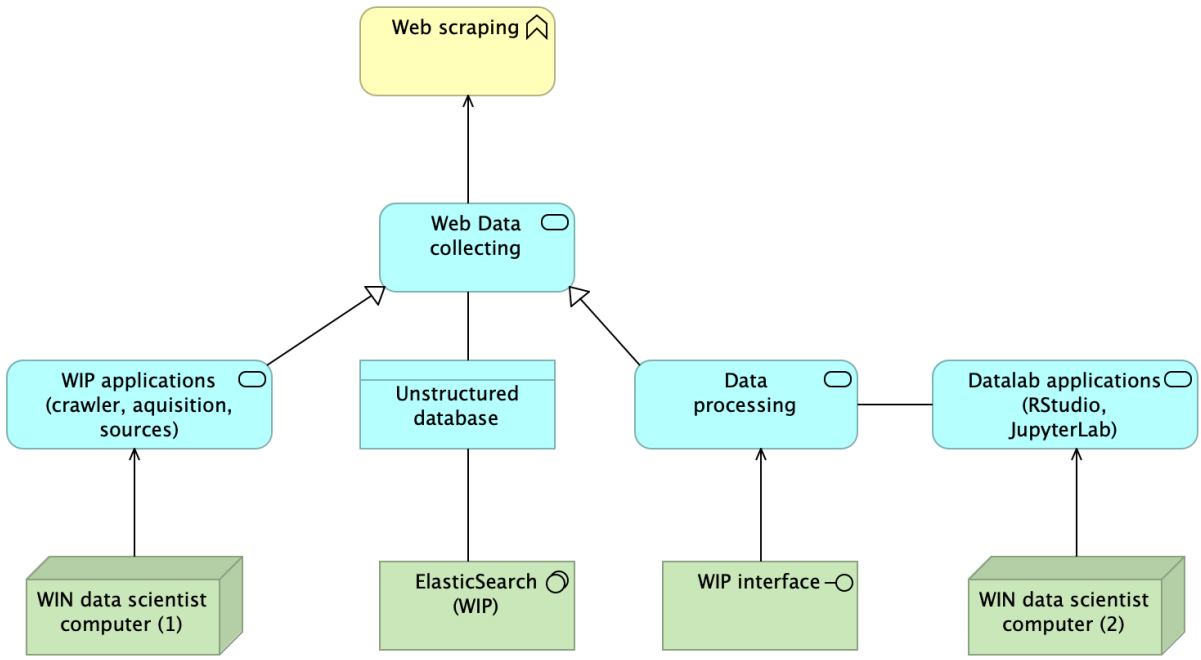
No	Territorial unit: country and NUTS 3		Occupation	number_of_ojas
1	Belgique/België	Arr. Antwerpen	Chemists	6
2	Belgique/België	Arr. Charleroi	Chemists	13
3	Belgique/België	Arr. Gent	Chemists	5
4	Belgique/België	Arr. Hasselt	Chemists	6
5	Belgique/België	Arr. Huy	Chemists	4

4. OBEC Use case

4.1. General outline on the statistical production

The business functions and technical components utilized within the Web Intelligence Platform Data Acquisition Platform (DAS) and the Online-Based Enterprise Characteristics (OBEC) Datalab are illustrated in the figure below.

Figure 2. Technical components and business functions of WIP-DAS and OBEC Datalab



Source: WP2 Deliverable 2.2. Second Interim Progress Report, 2023.

The data pipeline illustrates the process of data collection and analysis, which can be delineated into six distinct steps:

1. Preparing a dataset of enterprise URLs, accompanied by anonymized Business Register numbers.
2. Uploading the dataset of URLs into the Data Acquisition Platform (WIP-DAS).
3. Modifying the necessary parameters and initiating the web crawler.
4. Cloning the scripts responsible for generating statistics from GitHub repositories.
5. Executing the cloned scripts from step 4 to access data from the WIP-DAS for the websites that have already been scraped.
6. Processing the data to generate output files (in CSV format) containing statistical indicators.

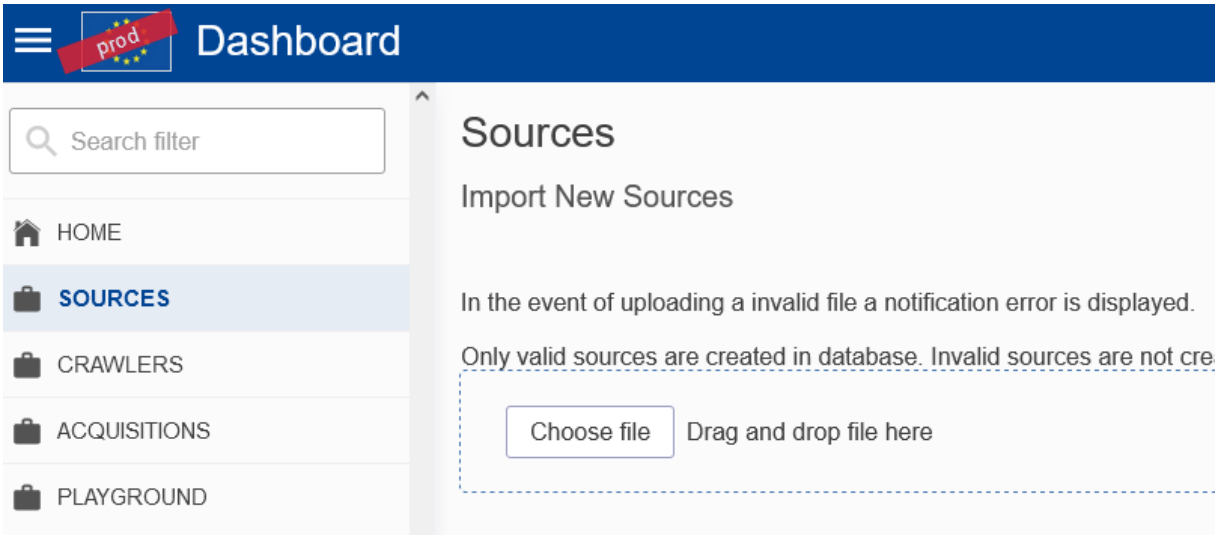
In the initial step (1), a JSON file must be prepared and subsequently loaded into the WIP-DAS.

An example of such a file is provided below:

```
{
"sources":[
 {
  "name": "PL_000001",
  "url": "https://stat.gov.pl",
 "group": "/OBEC"
 },
 {
  "name": "DE_000001",
  "url": "https://destatis.de",
 "group": "/OBEC"
 },
 {
  "name": "IT_000001",
  "url": "https://istat.it",
 "group": "/OBEC"
 }
]
}
```

This dataset is uploaded to the WIP-DAS through the dedicated interface provided by the platform (step 2). The figure below illustrates a screenshot demonstrating the process of defining the data source within the WIP-DAS.

Figure 3. Importing JSON files in the WIP-DAS



Source: WP2 Deliverable 2.2. Second Interim Progress Report, 2023.

The next step (Step 3) is to modify the necessary parameters of the crawler, such as defining the actions to be taken when encountering errors during website scraping, and then initiating the crawler. Data that has already been scraped and is available for extraction can be viewed in the OpenSearch repository.

The subsequent step (Step 4) in the OBEC pipeline involves cloning the repository of libraries necessary for executing the software that detects OBEC characteristics. For instance, the software responsible for detecting social media presence can be found at the following GitHub repository: https://github.com/jmaslankowski/WP2_OBEC_Starter. To clone this repository, the appropriate git cloning command must be executed within the OBEC Datalab environment.

The following step (Step 5) involves executing the scripts cloned from the GitHub repository. These scripts, written in Python, are designed to be run through Jupyter Notebook or JupyterLab. The example provided will enumerate all social media links present on the website, categorizing them by various social media channels, including Facebook, Twitter, YouTube, LinkedIn, Instagram, Xing, and Pinterest.

The final step of the OBEC pipeline (Step 6) is to retrieve the results generated in Step 5 for the purpose of aggregating the data and producing statistical indicators related to the social media presence of enterprises. Currently, our focus is on assessing the data quality aspects of this use case, particularly in identifying potential issues related to incorrect linkages between social media profiles and the corresponding enterprise URLs.

4.2. Technical manual on the OBEC use case step-by-step

The data collection process for our use case can be delineated into four fundamental steps, as outlined below:

1. **Preparation of a list of URLs** associated with the Business Register.
2. **Uploading the dataset of URLs** into the WIP-DAS.
3. **Modifying the necessary parameters of the crawler** and initiating the data acquisition process.
4. **Accessing the data** from the Datalab.

4.2.1. The use of anonymized Business Register to feed the data sources in the WIP-DAS

As previously mentioned, the initial step involves preparing anonymized Business Register numbers. This step is crucial because the Business Register should not be shared on the WIP-DAS, given that it may be accessible to other users. While the specific names assigned to these anonymized identifiers are not critical, it is essential that they remain unique, as they will be referenced in subsequent scripts. It is important to note that these identifiers must facilitate the

linkage between the list of URLs and the Business Register. This linkage is vital for enabling the aggregation of data by various attributes derived from the Business Register, such as the NACE code.

The JSON file should be defined this way:

```
{
"sources":[
  {
    "name": ”PL_00000001",
    "url": ”https://www.stat.gov.pl",
   "group": "/OBEC"
  },
  {
    "name": ”PL_00000002",
    "url": ”https://ug.edu.pl”,
   "group": "/OBEC"
  },
  {
    "name": ”PL_00000003",
    "url": ”https://gdansk.stat.gov.pl”,
   "group": "/OBEC"
  },
  {
    "name": ”PL_00000004",
    "url": ”https://szczecin.stat.gov.pl”,
   "group": "/OBEC"
  },
  {
    "name": ”PL_00000005",
    "url": ”https://wzr.ug.edu.pl”,
```

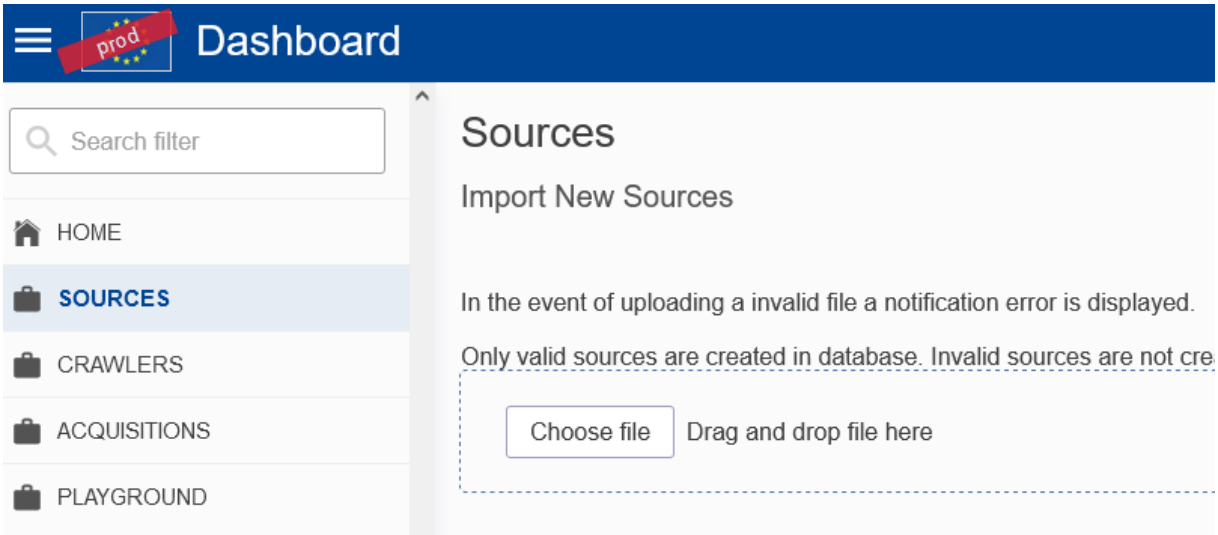
```
    "group": "/OBEC"
  }
]
}
```

The file prepared this way should be saved in JSON format. In fact this is the list of URLs as sources separated by comma. The group used in the file is OBEC.

4.2.2. Uploading the dataset of URLs into WIP-DAS

Once the file referenced in the previous step has been prepared, it can be uploaded to the dashboard of the WIP-DAS. To initiate this process, it is necessary to navigate to the "Sources" tab and select the file intended for upload to the platform.

Figure 4. Choosing the file to be imported to the WIP-DAS.

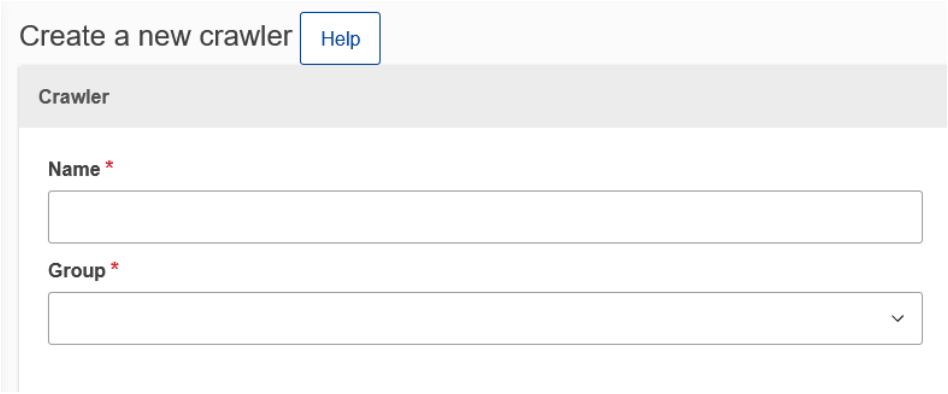


After successful import of the file, the report of importing the file will be displayed.

4.2.3. Modifying necessary parameters of the Crawler and starting the acquisition process

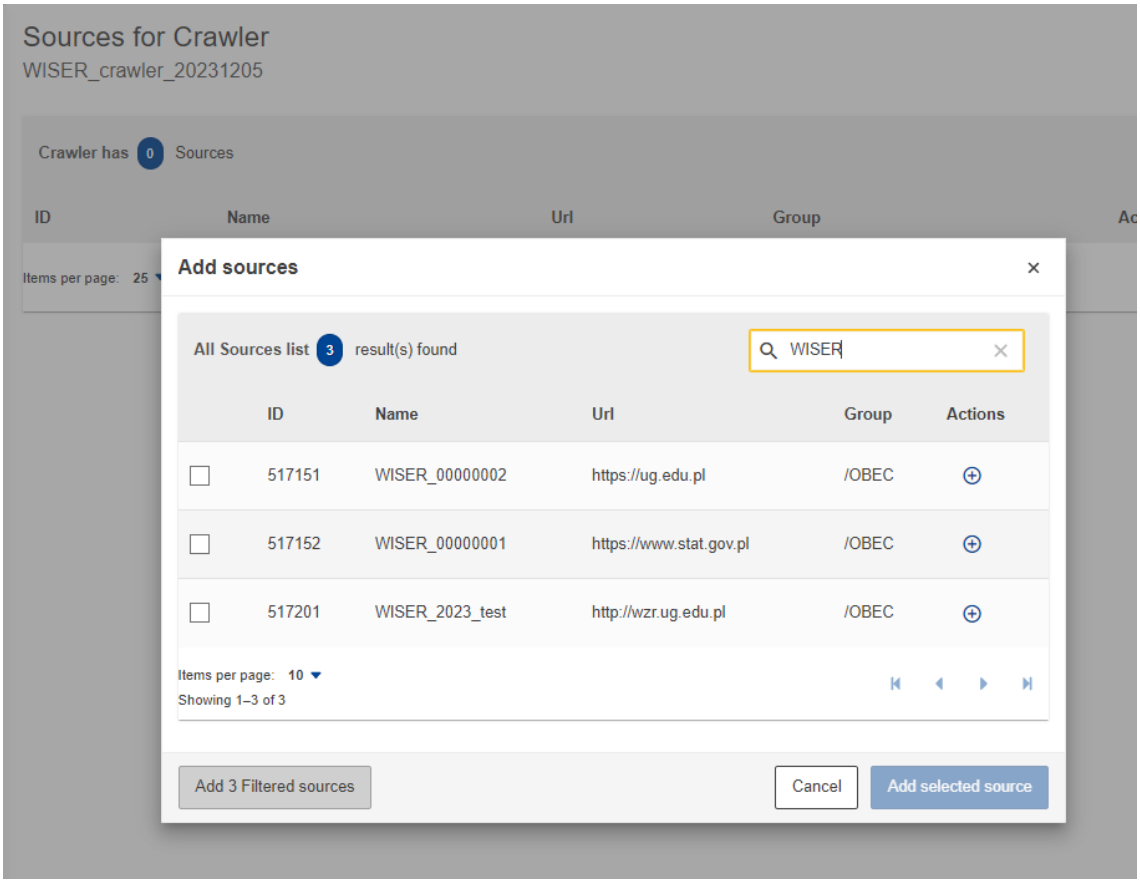
The next step involves the creation of a crawler that will be utilized during the data acquisition phase. The crawler must be assigned a unique identifier, and it should be categorized under the appropriate group, which in this instance is OBEC.

Figure 5. Creating a new crawler in the WIH-DAS



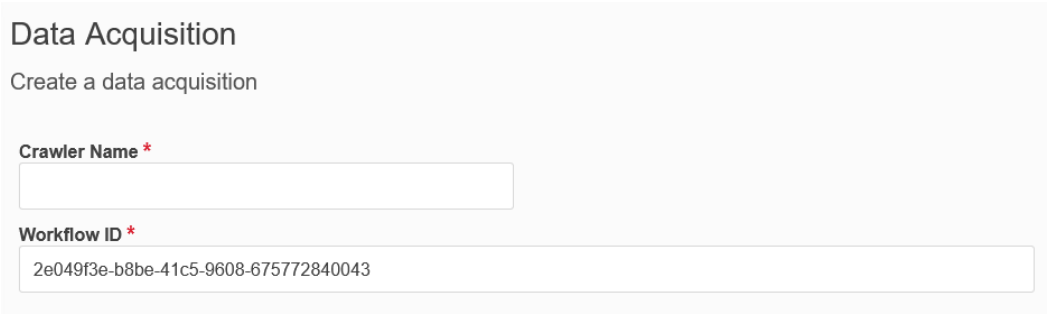
Following the creation of the crawler, additional URLs can be incorporated by selecting the "Add Sources" option, as illustrated in the figure below. Data services may be added individually or through batch import, utilizing the common segment of the URL names. In this instance, the shared portion of the names is "WISER," as depicted in the figure below.

Figure 6. Adding new URLs to the crawler in the WIP-DAS



Once the crawler has been defined, data acquisition can be initiated by selecting the "Acquisitions" option and creating a new workflow. Utilizing the default values will facilitate the retrieval of data from the landing page associated with each URL listed in the uploaded file. To enhance the depth of the data collection process, it is possible to specify the parameter for depth, allowing for a more comprehensive exploration of the website.

Figure 7. Starting the data acquisition in the WIP-DAS

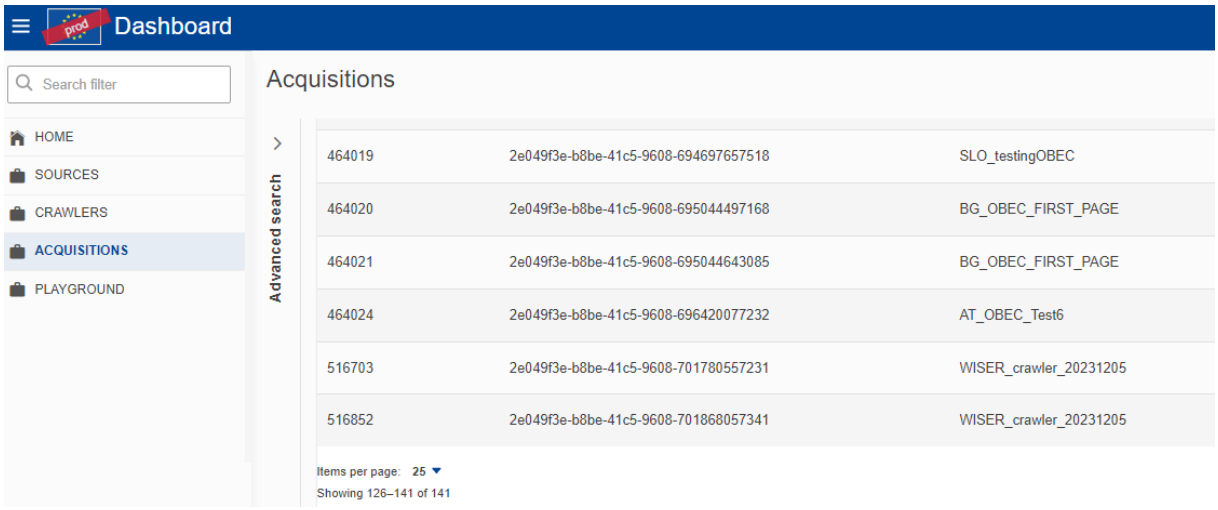


The workflow ID is generated automatically and will subsequently be referenced in the Python code to access the data collected by the crawler. Once the data has been successfully acquired, the crawler will be automatically terminated to finalize the data collection process. The termination of the crawler can also be done manually, even the data collection process is not finished.

4.2.4. Accessing the data from Datalab

The script developed by the work package tool for processing data to generate indicators on OBEC utilizes the data available within the platform. When initiating data analysis, it is essential to identify the acquisition ID that contains the data intended for processing. The list of data acquisitions and their corresponding crawler names can be accessed through the "Acquisitions" tab located on the left side of the WIP-DAS.

Figure 8. The list of acquisitions and associated crawlers in the WIP-DAS



The scripts developed for OBEC were authored in Python, necessitating access to the Datalab for execution. To reconfigure these scripts, it is imperative to know the acquisition ID associated with the specific crawler name of interest. The initial step involves updating the acquisition ID to the one relevant to the crawler intended for processing, specifically by modifying the parameter labeled query_content_doc.

All requisite algorithms for processing the data in accordance with OBEC standards have been encapsulated within the package named WIN_OBEC. In the code below, the class SocialMediaPresence is utilized, featuring its method searchSocialMediaLinks.

```
import WIN_OBEC as obec
import sys
import json
import requests
import re
import os
import pandas as pd
import json
opensearch_url = "https://...."
def get_result(query):
    payload={}
```

```
    headers = {}

    response = requests.request("GET", opensearch_url + query,
headers=headers, data=payload)

    return response.text
```

query_content_doc = "/content_2e049f3e-b8be-41c5-9608-726654126214_769098/_search?pretty=true"

```
smp=obec.SocialMediaPresence()

res = json.loads(get_result(query_content_doc))

# execute the searchSocialMediaLinks function for the first 50
URLs

for i in range(0,50):

    try:

        url=res['hits']['hits'][i]['_source']['fetched_url']

        htmlfile=res['hits']['hits'][i]['_source']['html']

        print(i,url)

        print("*"*100)

        smp.searchSocialMediaLinks(url,htmlfile)

        print("*"*100)

    except:

        break
```

The key components of the script outlined above include:

- **Importing the WIN_OBEC package**: This package provides the necessary functionalities for processing data related to OBEC.
- **Utilizing query_content_doc**: This variable serves as a reference to the data acquisition phase, facilitating data access.
- **Employing methods such as searchSocialMediaLinks, searchEcommerce, or searchLanguage**: These methods are designed to calculate indicators in accordance with Deliverable 2.4, which specifies the recommended tables for statistical production.

The classes within the script, such as SocialMediaPresence, do not feature a constructor. To instantiate a new object, one merely references the class. However, within the functions—specifically searchSocialMediaLinks—it is crucial to include two arguments: the first being the

URL and the second being the HTML file obtained from the WIP-DAS. The execution of the aforementioned script yields a list of URLs along with their corresponding social media channels, which is stored in a file named WP2_date.csv.

The output of the script is presented in the figure below.

Figure 9. Output of the OBEC script - the use of the searchSocialMediaLinks function

```
**********************************************************************************************************
1 https://www.proxiad.com/emplois/developpeur-net-f-h/
**********************************************************************************************************
The length of the scrapped content: 213386 characters
Number of links on website: 61
https://facebook.com/ProxiadGroup/
https://www.linkedin.com/company/proxiadfrance/
https://twitter.com/proxiadG
https://www.instagram.com/proxiad_group/
https://www.youtube.com/channel/UCYGV3gtqb6dZuu1UkH4kp7g
https://web.facebook.com/ProxiadGroup/
https://www.linkedin.com/company/proxiadfrance/
https://twitter.com/proxiadG
https://www.instagram.com/proxiad_group/
https://www.youtube.com/channel/UCYGV3gtqb6dZuu1UkH4kp7g
Total number of unique social media links found: 6
**********************************************************************************************************
2 https://www.ppd.com/who-we-are/location/australia/
**********************************************************************************************************
The length of the scrapped content: 182779 characters
Number of links on website: 176
https://www.linkedin.com/company/ppd/
https://twitter.com/ppdcro
https://www.facebook.com/PPDCRO
https://www.instagram.com/ppdcro/
https://www.youtube.com/PPDCRO
Total number of unique social media links found: 5
**********************************************************************************************************
3 https://asstroibg.com/produtki/m53/
**********************************************************************************************************
The length of the scrapped content: 147956 characters
Number of links on website: 82
https://www.facebook.com/asto.metal/
```

The output generated by this script provides critical insights into the scraped content. It includes the total number of links present on the website, as well as a character count of the website's content. Additionally, the script presents the number of unique social media links along with a comprehensive list of these links.

Other applications include, but are not limited to, evaluating whether a website addresses e-commerce-related issues. This script is grounded in the methodological framework outlined in WIN Deliverable 2.4. Additionally, as detailed in the same Deliverable, another use case involves the multi-language analysis of websites, the technical aspects of which are analogous to those discussed in this chapter.

For large-scale data processing Amazon OpenSearch is used which is a typical NoSQL index based database, specifically designed to search for terms in large textual databases.

The challenges and solutions associated with OBEC include the following:

- The target population may be unidentified or difficult to define.
- Obtaining a comprehensive list of relevant URLs may pose logistical challenges in certain ESS (European Statistical System) countries.
- Divergent regulatory requirements across countries may complicate the process; for example, some jurisdictions mandate the inclusion of tax identification numbers on business websites, while others do not.

From a legal perspective, the following issues are pertinent:

- A formal web scraping policy must be established to ensure compliance with legal frameworks.
- There may be evolving regulations governing the sharing of URL datasets and other associated data by National Statistical Institutes (NSIs).

Regarding methodology:

- Machine learning techniques may be suboptimal for this task due to their lower accuracy compared to the use of gold standard records.
- Text mining approaches are employed as part of the data analysis process to extract relevant information from web content.

Glossary

Amazon OpenSearch: A fully managed, scalable search and analytics service based on the open-source OpenSearch project. It is used to search, analyze, and visualize large volumes of data in real time. OpenSearch is commonly used for log and event data analysis.

AWS (Amazon Web Services): A comprehensive and widely adopted cloud platform provided by Amazon, offering a broad range of cloud computing services such as storage, compute, networking, machine learning, and analytics. AWS is used by organizations to scale and innovate their infrastructure and applications.

AWS Athena: A serverless, interactive query service provided by Amazon Web Services (AWS) that allows users to analyze data stored in Amazon S3 using standard SQL queries. Athena is designed to process large amounts of data without the need to manage infrastructure.

OpenSearch: A distributed search and analytics engine used for large-scale data retrieval and analysis. It is often used for full-text search, log analytics, and real-time data exploration.

JSON (JavaScript Object Notation): A lightweight, human-readable data format used for representing structured data. JSON is commonly used for transmitting data between a server and a web application, and it is easy to parse and generate in many programming languages.

Linux: An open-source, Unix-like operating system kernel that forms the basis for various operating systems, such as Ubuntu, CentOS, and Red Hat. Linux is known for its stability, security, and versatility, commonly used in server environments and for development.

NoSQL: A category of database management systems that do not use the traditional table-based relational model. NoSQL databases are designed to handle large amounts of unstructured or semi-structured data, and they provide flexibility for scalability, high availability, and performance. Examples include document stores, key-value stores, wide-column stores, and graph databases.

Python:A high-level, interpreted programming language known for its simplicity and readability. Python is widely used in fields such as web development, data analysis, artificial intelligence, machine learning, and automation.

URL (Uniform Resource Locator): A reference to a resource on the internet, commonly known as a web address. A URL specifies the protocol (e.g., HTTP or HTTPS), domain name, and path to the resource.

Web Scraping: The process of extracting data from websites using automated scripts or tools. Web scraping typically involves parsing HTML content, navigating web pages, and collecting data such as text, images, or links.

List of figures

Figure 1. Parameters used in configuration of the OJA Datalab 10
Figure 2. Technical components and business functions of WIP-DAS and OBEC Datalab ... 19
Figure 3. Importing JSON files in the WIP-DAS 20
Figure 4. Choosing the file to be imported to the WIP-DAS. 23
Figure 5. Creating a new crawler in the WIH-DAS 24
Figure 6. Adding new URLs to the crawler in the WIP-DAS 24
Figure 7. Starting the data acquisition in the WIP-DAS 25
Figure 8. The list of acquisitions and associated crawlers in the WIP-DAS 26
Figure 9. Output of the OBEC script - the use of the searchSocialMediaLinks function 28

List of tables

Table 1. Attributes available in processed data in the OJA DataLab 7
Table 2. Result of the suggested query to present the final set of tables 18

References

Web Intelligence Network: WP2 Deliverable 2.1. First Interim Progress Report, 2022.
Web Intelligence Network: WP2 Deliverable 2.2. Second Interim Progress Report, 2023.
Web Intelligence Network: WP2 Deliverable 2.3. Third Interim Progress Report, 2024.
Web Intelligence Network: WP2 Deliverable 2.4. Third Interim Progress Report, 2025.