Deploying a Scalable Big Data Platform to Enable a Food Safety Data Space

Mihalis Papakonstantinou, Manos Karvounis, Giannis Stoitsis, and Nikos Manouselis

Abstract The main goal of this chapter is to share the technical details and best practices for setting up a scalable Big Data platform that addresses the data challenges of the food industry. The amount of data that is generated in our food supply chain is rapidly increasing. The data is published by hundreds of organizations on a daily basis, in many different languages and formats making its aggregation, processing, and exchange a challenge. The efficient linking and mining of the global food data can enable the generation of insights and predictions that can help food safety experts to make critical decisions. All the food companies as well as national authorities and agencies may highly benefit from the data services of such a data platform. The chapter focuses on the architecture and software stack that was used to set up a data platform for a specific business use case. We describe how the platform was designed following data and technology standards to ensure the interoperability between systems and the interconnection of data. We share best practices on the deployment of data platforms such as identification of records, orchestrating pipelines, automating the aggregation workflow, and monitoring of a Big Data platform. The platform was developed in the context of the H2020 BigDataGrapes project, was awarded by communities such as Elasticsearch, and is further developed in H2020 The Food Safety Market project in order to enable the setup of a data space for the food safety sector.

Keywords Big Data · Artificial Intelligence · Food safety · Data platforms · Food recall prevention · Food risk prediction · Data space

M. Papakonstantinou · M. Karvounis · G. Stoitsis (✉) · N. Manouselis
Agroknow, Maroussi, Greece
e-mail: Mihalis.papakonstadinou@agroknow.com; manos.karvounis@agroknow.com; stoitsis@agroknow.com; nikosm@agroknow.com

E. Curry et al. (eds.), *Data Spaces*, https://doi.org/10.1007/978-3-030-98636-0_11

1 Introduction

The food system is rapidly changing, becoming more and more digitized. Data is being generated in all entities in the food supply chain. To better understand emerging risks and protect the consumers, it is essential to be able to combine, process, and extract meaning from as much data as possible.

Food safety and certification requirements throughout the supply chain have become stricter, and authorities are continuously in activity to develop and promote more selective methods of monitoring and controlling our food. That has led to a major increase in public data from official sources of food incidents across many countries that focus on a broad range of products and ingredients. A food incident can relate to an issue that could or is expected to impact multiple government jurisdictions. A food recall is an action taken by a food business to remove unsafe food from distribution, sale, and consumption. All food businesses must be able to quickly remove food from the marketplace to protect public health and safety.

When a food safety incident occurs somewhere in a global supply chain, simply gaining access to that top-level information is not enough to truly understand the repercussions. What about the product brand that was recalled? Which was the company behind the recall? And if you dive even deeper you also need to make sure you understand the company's involvement in the incident. Was it the manufacturer or the packer? Or was it possibly the importer or the trader? In other words, accessing comprehensive, reliable data in real time—and being able to properly analyze and harness that data—is critical for ensuring food safety in the increasingly complex, dynamic, and international supply chains.

Further to the food safety issues, more challenges could concern the monitoring of food fraud incidents around the world in almost real-time, large-scale data analysis in order to reveal patterns; predictions on whether someone in the supply chain has substituted, misbranded, counterfeited, stolen, or enhanced food in an unapproved way; and finally detection of increased probability of food fraud. We could further analyze all these challenges and find ourselves with even more, such as the identification of correlations between fraud incidents with price changes, the probability of fraud increase for suppliers based in countries with high corruption scores, and how the weather phenomenon is linked to an increase of food adulteration incidents in an area, amidst many more. To face the above challenges, Big Data platform that collects and processes many different data types is necessary in order to assess and predict risks [1].

During the last 15 years, very important fraud issues like the "2013 horse meat scandal" [2] and the "2008 Chinese milk scandal" (Wen et al 2016) have greatly affected the food industry and public health. One of the alternatives for this issue consists of increasing production, but to accomplish this, it is necessary that innovative options be applied to enhance the safety of the food supply chain [3]. For this reason, it is quite important to have the right infrastructure in order to manage data of the food safety sector and provide useful analytics to food safety experts.

There are several systems that are collecting and processing food safety information in order to support decision making in the food industry. This includes systems operated by public organizations like Rapid Alert System for Food and Feed [4] and commercial systems like HorizonScan (FERA [5]), gComply (Decernis [6]), and DigiComply (SGS [7]). These systems are based on a data collection and processing platform and are currently working on a production setup, serving thousands of users and providing access to hundreds of thousands of records. However, each of these systems is focusing on one specific data type and they are not combining and linking different data types.

In this chapter we present the design principles and the deployment details of a Big Data platform for the food safety sector that combines several different data types and can help food safety experts to make data-informed decisions. The data platform is designed to handle voluminous data that cannot be analyzed using the traditional data management techniques and warehousing. The proposed methodology and architecture was developed with an aim of increasing the scalability, availability, and performance of data platforms that need to handle processes such as data cleaning, data transformation, data unification, data enrichment, and data intelligence. The novelty aspects of our work consist of (a) the introduction of an architecture and a methodology that allows the processing and linking of highly heterogeneous data at a large scale and (b) an operational version of the first Big Data platform that links millions of food safety data records, and it provides food safety analytics and predictions that prove the scalability of the proposed approach. The methodology for data ingestion and processing and the architecture of the platform can be applied in other sectors in which processing of incidents announced by open data sources is very critical, such as pet food, cosmetics, medical devices, and pharmaceuticals. The chapter relates to the technical priorities: 3.2 Data Processing Architectures and 3.3 Data Analytics of the European Big Data Value Strategic Research and Innovation Agenda [8]. It addresses the horizontal concern of data processing architectures of the BDV Technical Reference Model and the vertical concerns of Big Data types and semantics. In addition, this work aims to maximize the contribution with the future European activity in AI and data by focusing on how a vertical organization like the food safety sector can be transformed, creating new opportunities [9].

2 Big Data Platform Architecture

The development of the Big Data platform for the food safety sector focused at targeting the needs of the food safety industry using Big Data processing, text mining, semantics, and Artificial Intelligence technologies. The platform is responsible for collecting, processing, indexing, and publishing heterogeneous food and agriculture data from a large variety of data sources [10]. The platform was designed using microservice architecture [11], with different technology components handling different aspects of the data lifecycle. All of the components

are interconnected using well-defined connectors and application programming interface (API) endpoints, each responsible for storing and processing different types of data (Fig. 1).

More specifically, the platform includes the following components:

- The **data sources ingestion** component, which connects to numerous data sources, extracts the data, and detects the changed data.
- The **data transformation** component, which performs data transformation to an appropriate format designed for performance optimization.
- The **storage** components, which feature various storage engine technologies that are used in numerous places throughout our architecture and are responsible for the physical archiving of data collections.
- The d**ata enrichment component** implemented using **machine learning (ML) and natural language processing (NLP)** technologies, which is responsible for hosting individual text mining, machine learning, and data correlation scripts that can be used in a variety of contexts as standalone pieces of code or as web services through the so-called intelligence APIs.
- The **data processing** components, which include a machine-readable interfaces (APIs) to the different types of data collected in the platform that is used in numerous places throughout our architecture. This part of the architecture is responsible for making data discoverable, but also for submitting new data assets back to the platform.
- The **monitoring** component, which is responsible for monitoring the data platform's smooth data digestion and performance.
- The **orchestration** component, which is responsible for the overall orchestration of all the processes that run in the Big Data platform.

3 Data Modeling

To address the dynamic requirements, the data model was designed in a way that allows the addition of new data types and new properties. To that direction we used a Functional Requirements for Bibliographic Records (FRBR)-inspired logic of metadata organization, so that there are basic and common metadata attributes describing each data asset, but also space for more customized attributes per data type (International Federation of Library Associations and Institutions 1998). The main goal was the adoption of a flexible and smart data model that is able to address dynamic requirements and support many different data types.

More specifically, the two advantages of such smart data model in a data platform are that the platform (a) can accommodate many different data types and entities and (b) can support different instances of the same data object that may be published by different data sources and in different formats which is very important when you need to deal with information duplication. The introduction of a common FRBR-based metadata schema to the platform made the process of integrating new data types much easier, in a way that also conveys a basic thematic and temporal view on

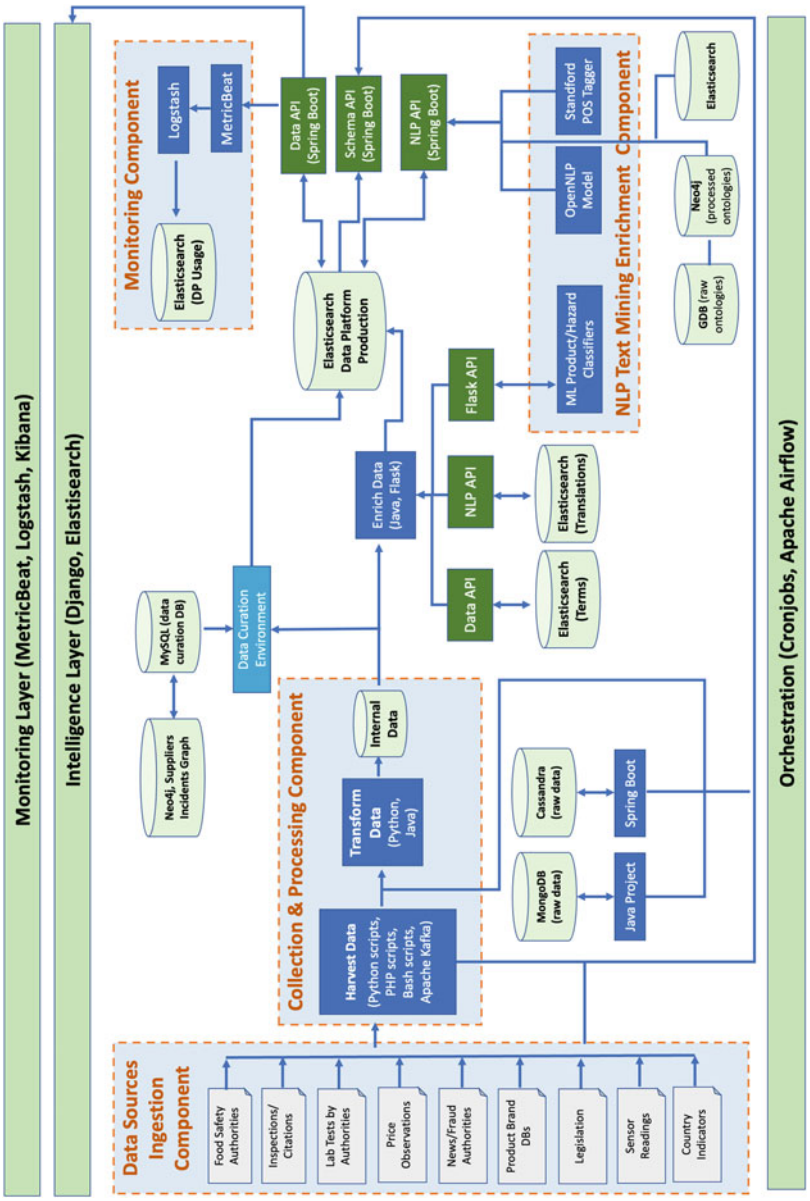

Fig. 1 Architecture of the Big Data platform

the underlying data assets. This data model was implemented using JSON, the open standard file format and data interchange format.

Each record in the data platform is structured according to a FRBR hierarchy (i.e., work, expression, manifestation, and item) where each FRBR level is described using a limited set of structural metadata elements complemented by faceted descriptions that can vary depending on the type of resource being described, its context, and the relevant FRBR level (example in the mind map below). The hierarchical data model that we developed includes a common part for all the different entities and data types with generic properties such as identifier, title, description, and date. The second part is specific to the data type and includes all the required properties that will enable data enrichment and linking (Fig. 2). The data model supports the use of standard and semantic vocabularies for several properties.

4 Data Standards Used

To enable data interoperability and data linking, we have used in our data model standard and well-adopted semantic vocabularies for several properties such as location, ingredients, materials, hazards, and suppliers. More specifically for the products, ingredients, and materials, we have used the FoodEx2 ontology [12] and the Food and Agricultural Organization (FAO) Codex commodity categories [13]. For the case of hazards we used the classification that is adopted by the European Food Safety Authority in systems like the Rapid Alert System for Food and Feed. In addition to that, we have foreseen the integration of the Global Standards 1 (GS1) for suppliers, product brands, and traceability information [14].

Data linking is achieved by annotating all the different data types with the terms from the same semantic vocabulary. This means that the recalls for specific ingredients such as cocoa are automatically linked to lab test results, border rejections, price data, and trade data. The use of hierarchical semantic vocabularies allows the automated expansion of the queries and analytics to parent and child terms. This approach enables the development of interactive charts for the analytics that allow the drill-down from generic categories of products and hazards to specific categories and instances.

5 Software Stack Used in Data Platform

An important aspect of the design of the Big Data platform was the selection of the software tools and systems that can meet the requirements for Big Data processing. In this section we present the software stack used in the food safety data platform and we analyze the rationale for the selection of each component.

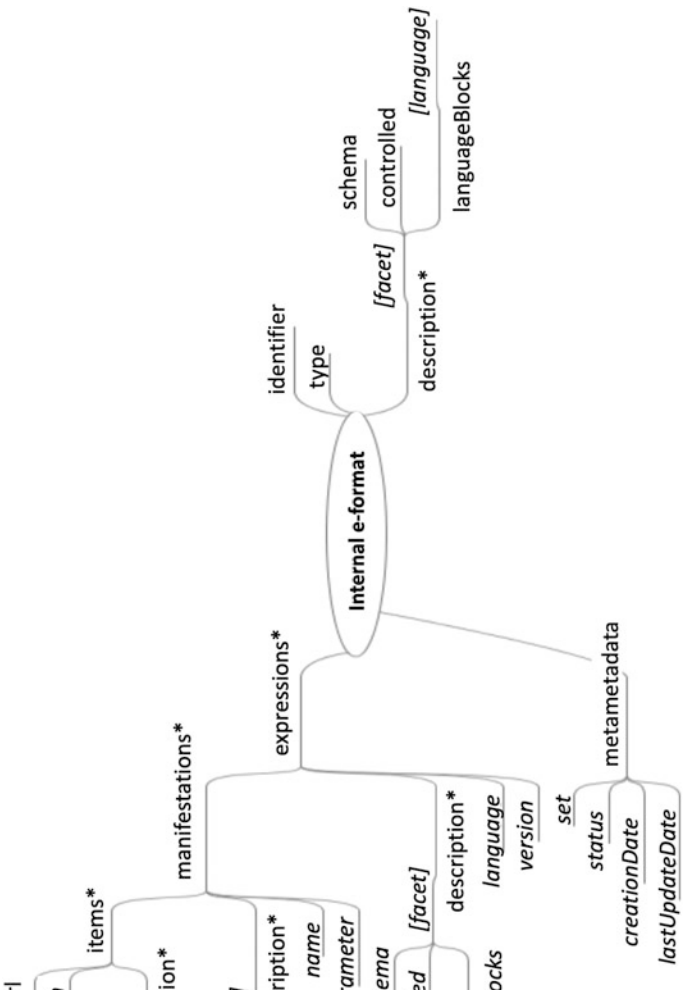

Fig. 2 The smart data model mind map for the internal data schema

5.1 Data Ingestion Components

Data ingestion is the first step in the Big Data platform for building data pipelines and also one of the toughest tasks in Big Data processing. Big Data ingestion involves connecting to several data sources, extracting the data, and detecting the changed data. It's about moving data from where it originated into a system where it can be stored, processed, and analyzed. Furthermore, these several sources exist in different formats such as images, OLTP data from RDBMS, CSV and JSON files, etc. Therefore, a common challenge faced at this first phase is to ingest data at a reasonable speed and further process it efficiently so that data can be properly analyzed to improve business decisions.

The data ingestion layer includes a set of crawling and scraping scripts for collecting information from the data sources. For each data source a different script was developed. These scripts vary in form. We utilize Scrapy, Python scripts, a custom Java project, as well as bash scripts to help in the collection of the tracked data sources. Regardless of their type, these collections of scripts take as input a starting URL and some rules and store the matching documents in the file system as output. Cron jobs are used to check every few mins whether new notifications have been published on the web site of the agency/authority. For storing the fetched data, a NoSQL database, namely, MongoDB, is used.

Apache Kafka is used to collect data streams such as environmental data and weather data from sensors. Apache Kafka is a messaging framework that is distributed in nature and runs as a cluster in multiple servers across multiple datacenters. Moreover, Kafka allows the real-time subscription and data publishing of large numbers of systems or applications. This allows streamlined development and continuous integration facilitating the development of applications that handle either batch or stream data. An important factor in data ingestion technology, especially when handling data streams, is the fault tolerance capability of the chosen technology. Kafka ensures the minimization of data loss through the implementation of the leader/follower concurrency architectural pattern. This approach allows a Kafka cluster to provide advanced fault-tolerant capability, which is a mandatory requirement for streaming data applications.

5.2 Collection and Processing Components

In the Big Data platform, each data source, depending on its format, is collected in a different way and might have a different form as mentioned in the above ingestion component. The raw data is harvested and then transformed using Python and Java code into the internal data model (schema) that was analyzed in Sect. 3. The main goal is for all the collected different types of data to have the same data structure.

5.3 Storage Components

The storage layer deals with the long-term storage and management of data handled by the platform. Its purpose is to consistently and reliably make the data available to the processing layer. The layer incorporates schemaless persistence technologies that do not pose processing overheads either when storing the data or retrieving them. Therefore, the storing and retrieving complexity is minimized. The software components that were used for the storage of the Big Data are analyzed below.

MongoDB is a distributed database which treats and stores data as JSON documents. Thus, data can have different fields and the data structure is essentially alive since it can be changed over time. Also, MongoDB provides ad hoc queries, supporting field query, range query, and regular expression searches. Moreover, MongoDB has fault-tolerant and load balancing capabilities by providing replication and sharing of the main database. In the data platform it is used to store the data fetched by the crawlers of the data sources.

Elasticsearch is a distributed database, providing a full-text search engine based on Lucene. The distributed nature of Elasticsearch allows near real-time search in all kinds of documents. The indices of Elasticsearch can be divided into shards, hence supporting automatic rebalancing and routing. Moreover, the indices can be replicated to support efficient fault tolerance. Furthermore, Elasticsearch encapsulates out-of-the-box methods for establishing connections with messaging systems like Kafka, which makes integration easier and allows the faster development of real-time applications. In our case Elasticsearch is used in many places throughout the stack. More specifically, we use it for text mining purposes by taking advantage of its analyzer capabilities, for the storage and aggregation of all the production-ready data, and for storing application performance metrics.

MySQL is a relational database management system (RDBMS), which provides a robust implementation of the SQL standard. The data platform integrated software stack also provides the phpMyAdmin user interface to monitor and query the MySQL RDBMS through a web user interface. In the data platform the MySQL database is used in the data curation environment to manage and manually enrich the records by food safety domain experts.

GraphDB is an RDF triplestore compliant with the core Semantic Web W3C specifications (RDF, RDFS, OWL). It acts as a SAIL over the RDF4J framework, thus providing functionalities for all critical semantic graph operations (storing, indexing, reasoning, querying, etc.). The query language used is the implementation of the SPARQL 1.1 specifications, while connectors with Elasticsearch and Lucene are incorporated in the system. In the Big Data platform it was used for storing the geonames ontology, which is queried through an endpoint of the internal API used for the enrichment to identify countries based on country, city, or region names, in the languages supported by geonames.

Neo4j is a native graph storage framework, following the property graph model for representing and storing data, i.e., the representation model conceptualizes information as nodes, edges, or properties. Accessing and querying the underlying

data is achieved via the usage of the open-sourced Cypher query language, originally developed exclusively for Neo4j. In the food safety Big Data platform, Neo4j was used for storing the processed semantic vocabularies for products and hazards.

Apache Cassandra is a NoSQL storage engine designed to handle large amounts of write requests. Being a NoSQL engine it can easily handle model updates. It is designed to be easily configurable and deployed in a multi-node, distributed manner. In the food safety data platform Apache Cassandra is used to store numerical data such as country indicators and sensor readings.

5.4 Data Processing Components

These components are used throughout our data platform in order for our data to travel and communicate with other components.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies, and several common framework-related tools. Extensions are updated far more frequently than the core Flask program. As part of the data platform, the Flask framework is used as a wrapper access layer on top of the machine and deep learning models trained for the classification of food recalls.

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. As part of the data platform, Django is used for the development of incident prediction and risk assessment APIs. On the one hand, food incidents data is used to train a deep learning prediction model to predict food incidents in the future. On the other hand, the main goal of the risk assessment module is to help food safety experts to identify the ingredients with unacceptable hazard risk.

5.5 Data Enrichment Components

Data enrichment in the Big Data platform for the food safety sector is achieved by applying machine curation processes and human curation processes.

In the case of machine curation, the data enrichment components are used to autotag the data with hazards, products, and country terms. An internal API

endpoint, a custom Java component for the processing of the notifications, and a set of Python scripts are used for the enrichment of the collected data.

The internal API has endpoints used for the textual processing of the notifications and identification of possible product and hazard terms. Using these endpoints we can (a) check the existence of a specific term against product and hazard vocabularies; (b) search for possible terms into text using fuzzy search, stemming, and N-grams; (c) get a stemmed version of text, without stopwords or without specific types of words (e.g., verbs); and (d) identify products in given brand names, using machine learning techniques trying to predict the product term based on the information already stored and frequently updated by human curation system.

The Java component is used to process and enrich the notifications collected by data collection component. It uses the internal API along with other endpoints to try to annotate each notification with the hazard and product terms it may involve. This component operates in any combination of the following ways:

- Controlled, in which only specific fields of the notification are used as possible input to the enrichment methods
- Smart, in which specific words and phrases usually associated with hazard terms (e.g., contains, due to, contaminated with) are used to make focused enrichment
- Country conversion, in which country names are converted into their respective ISO codes and vice versa
- Product enrichment, in which the complete text of the notification is sent to external services (OpenCalais) and the brand name to the internal API for the annotation with possible product terms
- Translate, in which the chosen text possibly containing hazard and product terms is sent to an external service, to translate it into English so it can be used by the enrichment methods

The set of Python scripts is used in two ways:

- The classification of product brands with their respective products. This extracts the information present on the curation system along with the tagged vocabulary terms, and using a set of Python packages used for machine learning (scikit-learn, NLTK) generates a predictive model used by the internal API for the annotation of text.
- The automatic approval of machine annotated hazard and product terms from the textual processing of each notification. Employed by the curation system, this script takes into account the machine-generated term, along with the reason behind the annotation, and using a set of Python libraries (pyjarowinkler, Porter Stemmer), it calculates the string distance between the two and automatically approves or not the generated term.

Furthermore, enrichment is employed for the identification of the country of origin for a recall, news item, outbreak, etc. This is achieved using the geonames ontology, imported into a GraphDB database which is queried through an endpoint of the internal API used for the enrichment. This endpoint identifies countries based on country, city, or region names, in the languages supported by geonames.

A machine learning API has been created for the enrichment of data platform's entities using machine learning techniques. This API uses two different models based on the annotation that will be attempted:

- One is using the title and textual description of the recall and is trained to identify the hazard which caused the recall.
- The other one is also using the title and description of the recall and identifies the products.

For both of them the SGDClassifier was used, along with a TFIDF vectorizer. This API has endpoints for the generation of the model with a new train dataset and for the identification of hazard and product terms and is built using Flask framework for Python.

For the human curation part the Drupal 7.0 is used as the data curation system of the data platform. The curation system includes:

- Feed importers to import the information that is collected and enriched
- Data curation workflow with specific roles and access rights
- Drupal rules for enriching and publishing information
- Drupal exporter to publish the curated and reviewed information

5.6 Monitoring Components

This component provides insights and visibility into the health and status of data platform's data clusters by tracking specific metrics in real time and sending alerts or notifications when readings exceed or fall below the set thresholds. Data collected from monitoring our data clusters can be stored, analyzed, and displayed in business intelligence and analytics dashboards and reports. The following software tools were used to monitor the health of the platform.

- **Logstash** is an open-source data collection engine with real-time pipelining capabilities. Data flows through a Logstash pipeline in three stages: the input stage, the filter stage, and the output stage. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Clean and democratize all your data for diverse advanced downstream analytics and visualization use cases.
- **Kibana** is an open-source data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases. It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support. Also, it provides tight integration with Elasticsearch, which makes Kibana the default choice for visualizing the data stored in Elasticsearch.
- Finally, we employ **Metricbeat**, i.e., the proposed tool for monitoring the scalability of Big Data platform, over the Elastic Stack to monitor and report our chosen metrics.

5.7 Intelligence Layer

The data platform includes an intelligence layer that is responsible for implementing the risk assessment and risk prediction algorithms using machine learning and deep learning methods [15].

Incident prediction module: Food incident data of the last 30 years is used to train a deep learning prediction model to predict food incidents in the future. The incidents' dataset becomes available through the data API of the data platform. To that direction, a request to the data API is sent with the product (ingredient or material) for which we want to predict the incidents of the next 12 months. We can build prediction models for specific hazards and specific regions by filtering the result set with the country of interest and the hazard.

For the implementation of the prediction models that are based on the historical food safety incidents, Prophet is used [16]. Prophet is a procedure for forecasting time-series data based on an additive model where nonlinear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend and typically handles outliers well.

Risk assessment module: For the implementation of risk estimation, we used a mathematical model that is based on the frequency of the incidents and the severity of the identified hazards in the incidents. The risk is estimated for all the possible product (ingredient, raw material) hazard pairs. Considering that the data platform includes more than 12,000 products (ingredients and materials) and more than 4300 hazards, the risk estimation should be conducted for a large number of pairs.

The risk assessment module can be used in the following ways:

- A request to the risk API to calculate the risk for a specific ingredient, origin, and hazard. This option can be used every time that we need a risk estimation for a specific time point.
- A request to risk API to generate a batch of risk time series. This option can be used to estimate the risk for a specific time period, e.g., evolution of the risk during the last 10 years.

6 Operational Instance of the Data Platform

An operational version of the data platform for the food safety records was deployed using the analyzed architecture and Big Data and Artificial Intelligence technologies. The main source of the datasets in the Big Data platform is the open data published by the national authorities, international systems, and food safety portals. The data platform includes records of news, food recalls, border rejections, regulation, fraud cases, country indicators, price data, sensor data,

supplier data, product brands, and inspections. More specifically, as of March 2021, the operational version of the Big Data platform includes the following data types:

- **Food Incidents**: 412,308 food recall warnings on a big variety of products with a growth rate of 6.3% per year. The data is announced on a daily basis by each food safety authority worldwide. The data has a variety of types (food recalls happening at market level, border rejections, information for attention) and with specific attributes following a different schema for each provider.
- **Food Inspections**: 227,603 inspections with a growth rate of 1.46% per year. The data is announced on a 2–5-month basis. The data has a variety of types with specific attributes following a different schema for each provider.
- **Lab Tests**: 102,187,114 laboratory analysis results that come from 34 national monitoring programs with a growth rate of 14.68% per year. The dataset is announced on a yearly basis by each food safety authority worldwide. The data are of a specific type with specific attributes and in an xls format following a different schema, depending on the provider.
- **Suppliers**: 702,578 different company suppliers with a growth rate of 1.46% per year. The data is announced on a daily basis.
- **Price**: 415,670 prices with a growth rate of 14.9% per year. The data is announced on a daily basis.
- **Country Indicator**: 38,061,648 country indicators with a growth rate of 5.14% per year. The data is announced on a yearly basis.
- **News:** 68,971 news records with a growth rate of 6.57% per year. The data is announced on a daily basis.
- **Maximum Residue Level Limit:** 141,594 records with a growth rate of 0.7% per year. The data is announced on a yearly basis.
- **Product Brand**: 28,081 brand records with a growth rate of 15.6% per year. The data is announced on a daily basis.
- **Sensor data:** 12,437,743 sensor readings from weather stations that are installed on the farms and temperature and humidity sensors that are installed on the processing units. The growth rate of the sensor data is 5.6% per year.

All of the above amount to 142,245,567 data records.

7 Identifying Records in the Big Data Platform

A very important aspect that you need to take into consideration when building a (Big) Data platform instance is on how to assign IDs to records in the data platform. Several practices may be applied to tackle this problem:

- Applying a hash function over crawled/scraped urls
- Some kind of internal identification process
- Attempting to identify/extract each source's unique identification method

7.1 Hash Function over Crawled Urls

This is a somewhat safe approach; urls are unique throughout the web so chances are a hash function on top can prove to be successful. It however does not come without any drawbacks. What if there are updates to the content crawled? It is not uncommon for urls of websites to be generated based on the title of the source. It is the piece of text containing the most important information on the generated content and the most SEO-friendly one. So what about updates to the titles? This can lead to updates to the url as well. So even though that is a rather straightforward choice, special care should be taken to such updates in order to avoid duplicates.

7.2 Internal Identification Process

This can be implemented either by deploying an API endpoint responsible for assigning an ID to each resource collected or a simple method/function/bash script.

The above suggested method has some very important pros, the most important of them being its blackbox way of working. Once it has been perfected, you no longer have to worry about duplicates in your platform or assigning the same ID to two different resources.

However, they have some cons as well. First and foremost, time should be spent perfecting such a mechanism. Due to the importance of ID assignment in data-related projects and platforms, one should definitely allow many hours (or story points) to such a project/task since it will be the backbone of pretty much everything you build. Another drawback we should point out is the rationale behind the identification process. Basing it uniquely on the collected content can lead to duplicates as described in the previous case.

7.3 Remote Source Identification

This approach is considered as the most challenging choice available. Although one may think of this as trivial if the data collected is in an xls or csv format where identification is rather straightforward, what if a content management system (CMS) is employed? Knowledge of it should be present if one wants to successfully assign a unique ID able to avoid duplicates. For instance, Drupal assigns a unique id to each piece of content (nid) always present in meta-tags and by default in CSS classes of article tags. However, if employed correctly one should never worry about their ID assignment or almost never. Care should be taken only when some major migration takes place on the remote source's side, a rather infrequent case.

In the Big Data platform we applied a hybrid approach. More specifically, all of the aforementioned approaches are utilized in a priority manner:

- First, we attempt to identify the ID given to the data record by the remote source.
- If this method fails, we employ our internal ID assignment that hashes a concatenation of the important and never-changing properties of the data type. For example, such a vector of properties for a food recall is the date, the reason behind the recall, and the product involved in it.
- Finally, if we are not able to safely extract these properties from the official source of information, we employ the hashing over the collected url.

Regardless of the technique utilized for our collected records, we also include our internal ID uniquely given to each of the sources we collect our data from.

8 Orchestrating the Big Data Platform

As already analyzed, the data platform collects, translates, and enriches global food safety data. A number of workflows are involved in the process. Tasks triggering one another, signifying the collection, processing, and enrichment of each of the close to 200M data points that are present in the infrastructure.

There is a very important challenge that we had to take into account in designing and implementing our Big Data platform stack. Initially we utilized cron jobs for these workflows. Every data source we track has its dedicated directory in the backend and processing servers, and within each of these directories a run.sh script is used. This is the script that manages all the action. Every single task in each workflow triggered is managed by such a script, calling other scripts created with the responsibility to handle each task. And this run.sh script is triggered by crontab.

Depending on the source, translation endpoint triggering scripts may be present. Text mining or text classification workflows may take place with their respective scripts. All initiate calls to the respective projects and endpoints.

The key points in the orchestrating process are:

1. We need to dive into the data and identify the rate at which new records are published in order to configure the extract, transform, load (ETL) workflows to be triggered only when chances of new data are present.
2. Only new data needs to be taken into account. In our stack, each of the data records collected comes with a collection timestamp and a flag signifying whether the record has been processed or not.
3. Implementing and deploying a workflow capable of executing regardless of the stress levels of a server is really challenging. A good choice at this point is splitting the workflow into atomic operations; this ensures that even though a task or a workflow may not be complete, no data loss will be observed since each new workflow triggered will always check for the previous workflows' leftovers.

In the Big Data platform, we are using Apache Airflow for the ETL pipelines. Apache Airflow is one of the best workflow management systems (WMS) that

provides data engineers with a friendly platform to automate, monitor, and maintain their complex data pipelines.

Just to give a quick overview in terms of numbers, in the current infrastructure of the Big Data platform:

- **113 ETL** workflow cron jobs are present.
- On average workflows are triggered once every **10 min**.
- **9 dedicated servers** are involved in this part of the infrastructure.
- **11** workflow jobs have been switched to Apache Airflow DAGs.
- **1 Elastic Stack** instance (involving Elasticsearch, Kibana and Metricbeat) is employed to keep track of the health of our infrastructure.

In terms of performance, the approach described in this section has proven to be a very robust one. Since Apache Airflow is deployed as a system service in the dedicated server tasked with these workflows, monitoring its uptime is a very straightforward task. Furthermore, it provides an out-of-the-box logging mechanism to easily identify possible bottlenecks and CPU/memory-hog steps in each workflow. Finally, since internally it depends on the cron daemon of the underlying infrastructure (present in all *nix distributions), we can be certain that every workflow will be triggered at the time requested.

9 Monitoring the Big Data Platform

Using the Big Data software stack you may build a very robust data platform instance, capable of handling huge amounts of data, harmonizing, linking together, and enhancing in any ML/DL or any other possible way available. However, what if:

1. A node in your Elasticsearch instance stops working?
2. Your MongoDB deployment consumes too much memory?
3. The awesome model you have designed and implemented takes up too much CPU/GPU?
4. The Apache Airflow webserver and/or scheduler stops working?

In the data platform stack, we have employed two levels of uptime monitoring: one for the uptime of the components of the stack and another for their performance. We will further analyze both in the rest of this section. The integration of these monitoring approaches has led to a downtime of under 1 h over the past 5 years.

9.1 Ensure Uptime for Every Component Deployed to the Stack

As already presented earlier in the chapter, the Big Data platform infrastructure is a microservice one. This means that every single piece of component is wrapped up

with a bunch of API endpoints, accessible through various ports in our servers. In our infrastructure, every newly deployed project, API service, and storage engine in the infrastructure come with a checkup script. This means that each and every time a new framework, tool, storage engine, and project are deployed, a script is also set up as a cron job.

This accompanying script is actually quite simple and is automatically generated with a command line tool we have implemented. All it does is that it uses *nmap* to check whether a port is open and accepting traffic or not. If it is not an automatic attempt to restart, the respective service is made and an email is sent with the last 50 lines of the respective service's logfile. To handle out-of-memory issues, the system's cache is also cleared. To ensure uptime of the stack, these scripts are added as cron jobs and are executed every 2 min.

9.2 Problems That Are Not Related to the Infrastructure

Many problems may come from the implementation and wrap-up code in a Big Data platform. One has to ensure that such cases are also monitored. Elastic has made available a tool to that end. APM is released by Elastic, fully maintained and easily integratable. In our data platform, we have connected APM to our dedicated Elasticsearch instance for monitoring purposes and the respective Kibana instance for visualization purposes. Using this Kibana instance we analyze and visualize the performance of each of the components of the stack and can go deeper into our analysis, identifying endpoints that take too long to respond and consuming too much CPU or memory, or even identifying hacking attempts leading to many UNAUTHORIZED HTTP responses and where they were made from.

10 Performance and Scalability of the Data Platform

In order to evaluate the performance and the scalability of the operational data platform, we performed a rigorous testing experimentation for three critical steps, namely, data ingestion, incident prediction, and risk assessment. We focused our experimentation on the performance of the Big Data platform stack by tracking system indicators such as (a) completion time, both on a step-by-step level and on the whole end-to-end data flow, (b) CPU (central processing unit) usage (we will track the CPU usage by each of the components as they are triggered by the flow), (c) memory usage of each of the components and technologies, and (d) network usage in terms of bytes (we employ this metric, since the whole stack is based on a microservice architecture).

More specifically, we performed the experimentation using 3 real scenarios for the dataset upload, 20 different use cases for recall prediction experiment, and risk assessment experiments to better showcase the scalability potential of the deployed

Table 1 Different data types processed by the Big Data platform and their growth rate

Data type	Number of records (as of March 2021)	Annual growth (%)
Food safety incidents	412,308	6.3
Inspections	227,603	1.46
Laboratory testing results	102,187,114	14.68
Suppliers	702,578	1.46
Prices	415,670	14.9
Country indicator	38,061,648	5.14
News items	68,971	6.57
Maximum residue level limit	141,594	0.7
Product brands	28,081	15.6
Sensor data	12,437,743	5.6

Big Data stack. Since all of the data platform has been deployed in a microservice architecture, our step-by-step and end-to-end experimentation was done over the API endpoints provided by the platform. To showcase the potential of the deployed stack in terms of scalability, we performed the experimentation using three usage scenarios for each step of the process by gradually increasing the requests made toward the platform. To have more accurate results in the case of the incident prediction, 20 different scenarios are tested due to the small data size. First, we identified and abstractly described the provided datasets of Table 1. Then we moved on with evaluating them against the Vs of Big Data, and we identified the data flow each dataset will follow in the stack, denoting the steps of this flow. Finally, using a Python script that simulated bursts of this data flow throughout the data platform stack, we monitored and reported our chosen metrics for this benchmark in real time. We employed Metricbeat, i.e., the proposed tool for monitoring the scalability of Big Data platform, over our Elastic Stack to monitor and report the chosen metrics. The methodology and the results of the platform's evaluation are presented in detail in a previous study of our team [15].

According to the results of the performance evaluation, in terms of the dataset ingestion step, the data platform had a good performance with respect to the completion time as well as the CPU, network, and memory usage. It is a step that can be easily made with a high degree of concurrency without seriously affecting the rest of the stack. The incident prediction demonstrated a very good performance with respect to the completion time as well as the CPU, network, and memory usage. Performance increased by lowering the volumes of data handled. Increasing data needs more time, more CPU, and memory to be trained. Similar behavior is observed in risk assessment step.

11 Discussion

Building a Big Data platform that can collect and process all the available global food safety data requires an architecture that is scalable and can accommodate the dynamic requirements in functionalities, data, and speed of analysis. As presented in this chapter the starting point of the design is a robust internal data schema that can scale to many different data types and can support data standards, i.e., standard properties and semantic vocabularies. Putting significant effort to define the internal data model will give important advantages in the long run.

As presented in Sect. 10, an important aspect that needs to be carefully tested is the scalability of a Big Data platform both in terms of volume and velocity of data. To that direction we have conducted several experiments for realistic scenarios with real and synthetic datasets [15]. The parts of the Big Data platform that are more computational intensive are the data processing and enrichment processes. Increased data volume and velocity can be supported by expanding the infrastructure with more computational and memory resources but still keeping the platform very efficient.

It is critical to have an open architecture that can easily support new data sources and data types but also new processes. Using a data platform that is based on microservices enables such an open and scalable architecture. We have validated this in the case of the food safety sector through many new requirements for new data types and data sources in order to provide descriptive and predictive analytics that will help the experts to take data-informed decisions.

In terms of future work, the Big Data platform will be expanded by mechanisms that will assign global identifiers to entities such as companies, batch numbers of the products, hazards, ingredients, and raw materials. Relying on such global identifiers will enhance the traceability information that will be managed and processed by the platform.

Furthermore, as the scope of the Big Data platform expands from managing only open data to also managing and processing private data, it is important to look at the data security aspects that will enable the setup of public-private data trusts [17]. We are already working in the context of the Food Safety Market H2020 project (https://foodsafetymarket.eu/) on the deployment of authentication and authorization methods as well of data anonymization techniques. We start by deploying secure data exchange services for very critical processes like food certification. Combining AI, data, and security technologies, we aim at creating a data space for the food safety sector.

Finally, the development of marketplace services that will allow data sharing, data discovery, and data monetization is a next step that will open up new possibilities in the food safety sector [18].

12 Conclusions

This chapter presented a Big Data platform that efficiently collects, processes, links, and mines global food safety data to enable the generation of insights and predictions that can help food safety experts to make critical decisions and make our food safer. Using data standards and deploying mature and state-of-the-art Big Data technologies, we managed to develop a data platform that is open and scalable. Best practices on the deployment of data platforms such as identification of records, orchestrating pipelines, automating the aggregation workflow, securing exchange, and monitoring of a Big Data platform were shared.

Acknowledgment This work is funded with the support by the European Commission and more specifically project TheFSM "The Food Safety Market: an SME-powered industrial data platform to boost the competitiveness of European food certification" (grant no. 871703) (https://foodsafetymarket.eu/), which is funded by the schema "Innovation Actions (IA)" under the work program topic "H2020-EU.2.1.1.—INDUSTRIAL LEADERSHIP—Leadership in enabling and industrial technologies—Information and Communication Technologies (ICT)." This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use, which may be made of the information contained therein.

References

1. Sklare, S., Kontogiannis, T., & Stoitsis, G. (2020). The case of chocolate: Lessons for food authenticity and big data. In *Building the future of food safety technology: Blockchain and beyond*.
2. Cavin, C., Cottenet, G., Cooper, K. M., & Zbinden, P. (2018). Meat vulnerabilities to economic food adulteration require new analytical solutions. *Chimia, 72*(10), 697–703. https://doi.org/10.2533/chimia.2018.697
3. Spink, J., Elliott, C., Dean, M., & Speier-Pero, C. (2019). Food fraud data collection needs survey. *NPJ Science of Food, 3*, 8. https://doi.org/10.1038/s41538-019-0036-x
4. RASFF – Rapid Alert System for Food and Feed. (1979). https://ec.europa.eu/food/safety/rasff-en
5. HorizonScan. (2015). *FERA*. https://horizon-scan.fera.co.uk/
6. gComply. (2015). *Decernis*. https://decernis.com/solutions/gcomply/
7. DigiComply. (2017). *SGS*. https://www.digicomply.com/
8. Zillner, S., Curry, E., Metzger, A., Auer, S., & Seidl, R. (2017). *European big data value strategic research & innovation agenda*. Big Data Value Association.
9. BDVA, CLAIRE, ELLIS, EurAI and euRobotics, *Joint Strategic Research Innovation and Deployment Agenda (SRIDA) for the AI, Data and Robotics Partnership*, September 2020.
10. Jin, C., Bouzembrak, Y., Zhou, J., Liang, Q., van den Bulk, L. M., Gavai, A., Liu, N., van den Heuvel, L. J., Hoenderdaal, W., & Marvin, H. J. P. (2020). Big Data in food safety - A review. *Current Opinion in Food Science, 36*, 24–32. ISSN 2214-7993
11. Indrasiri, K., & Siriwardena, P. (2018). *Microservices for the enterprise - Apress, Berkeley*. Springer.
12. EFSA. (2015). *The food classification and description system FoodEx2* (revision 2). EFSA supporting publication EN-804. 90 p.
13. FAO. (1994). *Commodity categories*. http://www.fao.org/fao-who-codexalimentarius/codex-texts/dbs/pestres/commodities/en/

14. GS1. (2016). *EPC Information Services (EPCIS) Standard*. https://www.gs1.org/sites/default/files/docs/epc/EPCIS-Standard-1.2-r-2016-09-29.pdf
15. Polychronou, I., Stoitsis, G., Papakonstantinou, M., & Manouselis, N. (2022). Stress-testing big data platform to extract smart and interoperable food safety analytics. *International Journal of Metadata Semantics and Ontologies*.
16. Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician, 72*(1), 37–45.
17. U.S Food and Drug Administration. (2020). *New era of smarter food safety, FDA's Blueprint for the Future*.
18. Otto, B., & Jarke, M. (2019). Designing a multi-sided data platform: Findings from the International Data Spaces case. *Electron Markets, 29*, 561–580. https://doi.org/10.1007/s12525-019-00362-x
19. Gaona-García, P. A., Stoitsis, G., Sánchez-Alonso, S., & Biniari, K. (2016). An exploratory study of user perception in visual search interfaces based on SKOS. *KO Knowledge Organization, 43*(4), 217–238.
20. K.G. Saur Verlag. (1998). *Functional requirements for bibliographic records: Final report*. KG Saur.
21. Lianou, A., Papakonstantinou, M., Nychas, G.-J. E., & Stoitsis, J. (2021). *Fraud in meat and poultry products, Food Fraud Book*. Academic Press.
22. Polychronou, I., Katsivelis, P., Papakonstantinou, M., Stoitsis, G., & Manouselis, N. (2020) Machine learning algorithms for food intelligence: Towards a method for more accurate predictions. In: Athanasiadis I., Frysinger S., Schimak G., Knibbe W. (eds) Environmental software systems. Data science in action. ISESS 2020. *IFIP Advances in Information and Communication Technology*, vol. 554. Springer. doi:https://doi.org/10.1007/978-3-030-39815-6_16
23. The Food Safety Market: an SME-powered industrial data platform to boost the competitiveness of European food certification. (2020). https://foodsafetymarket.eu/.