ID N. AIDR01/16915

UNIVERSITÀ CAMPUS BIO-MEDICO DI ROMA

DEPARTMENT OF ENGINEERING

UNIVERSITÀ DEGLI STUDI DELLA CAMPANIA LUIGI VANVITELLI

DEPARTMENT OF ENGINEERING

Italian National Ph.D. in Artificial Intelligence
Health and Life Sciences
XXXVII Cycle

Design and Analysis of Cloud Edge Architectures for the Development and Deployment of Distributed AI Application in Health Domain

Supervisors

Prof. Beniamino Di Martino

Prof. Antonio Esposito

Candidate

Dott. Gennaro Junior Pezzullo

10, 2024

To my family

Acknowledgements

I would like to thank...

Abstract

The work exhibited in this dissertation focuses on Cloud Continuum (CC) to analyze and define different cloud and cloud-edge architectures on which practical applications can be deployed in various areas such as healthcare, the military, cultural heritage preservation, and energy communities. The organization of the entire work consists of a series of case studies divided into three main areas: expert systems and semantics, machine and federated learning, and augmented and virtual reality. Each case study includes the design, implementation, and testing of reference architectures specific to each area. Projects covered include the development of a story composition tool and an expert system for COVID-19 diagnosis based on probabilistic and semantic rules. Other case studies cover the application of deep learning algorithms for analyzing events related to energy communities, implementing a federated supply chain in the military, and air quality monitoring using machine learning and federated learning. Pre-trained models for object recognition in augmented and virtual reality contexts are also explored, with practical applications tested at cultural institutions and technology fairs. In addition, the appendix includes a selection of published works that, while not directly central to the thesis, have contributed to shaping and supporting its development. This research is essential to developing reference architectures for the Cloud Continuum, with significant implications for the evolution of cloud-edge technologies in real-world contexts.

Contents

Chapter 1

Introduction and Objectives

In recent decades, the technological landscape has been characterized by a lot of dynamic changes. Many paradigms have entirely revolutionized the way we use technological devices. One of them is the Cloud in all its forms. We use the term Cloud Continuum (CC) [1] to refer to all cloud components from the core to the edge. This family of technologies has paved the way for numerous innovations in different sectors: from healthcare to the military, from energy communities to cultural heritage preservation. Through numerous works, many of which have been published in journals or presented at conferences over the past three years, this thesis exposes the interactions between cloud and edge components. The objective is twofold: first, to provide an in-depth analysis of cloud-edge architectures in emerging technological contexts such as artificial intelligence (AI) [2], and second, to explore practical applications through the definition, implementation, and testing of reference architectures. The analysis is based on recent studies and a review of the literature, coupled with experiments and application projects in different domains, which we will explore in detail.

1.1 Overview of the Context

In recent years, data generation, processing, utilization, and storage have undergone significant transformations. In its centralized version, the cloud has improved most implementations' functional requirements, almost eliminating scalability problems rather than maintenance or proper data storage. However, it still presents issues that are inherent to its very nature. Because cloud storage and computing sources are, by definition, remote from the user making the request, they show limitations in responding optimally to these new requirements, especially for applications that require an almost immediate response time [3] or that handle continuous data flows from distributed devices. In this scenario, edge computing, a paradigm that moves some of the processing

to the "edge" of the network, thus closer to the devices, has rapidly taken hold. In other words, data no longer needs to be transferred to a remote data center for processing; it can be processed locally, significantly reducing latency and improving the speed of operations. This decentralization of data processing has proven particularly useful in scenarios such as augmented or virtual reality applications [4], autonomous driving systems [5], or real-time monitoring of vital parameters in healthcare [6]. On the other hand, the evolution of cloud has provided a solid foundation for centralized processing, allowing access to virtually unlimited computing and storage resources via the Internet and fostering the spread of centralized models in the AI context. Integrating cloud and edge computing creates a synergy in which the two paradigms complement each other. The cloud offers large-scale processing capabilities, enabling deep analysis and long-term storage, while the edge handles operations that require immediate responses, optimizing data transfer and reducing network congestion. This combination finds application in numerous areas, including e-health, where real-time is crucial. Consider, e.g., the continuous monitoring of a patient's vital parameters and the ability to make predictions with a machine or deep learning model, where early intervention in case of abnormalities is essential. Or remote surgery, also made possible by AR/VR technologies [7], where accuracy and speed are critical to ensure successful operations. In this work, other areas will also be explored, such as the cultural and military sectors, where speed of response and operational efficiency are priorities. The analysis will focus on the different fields where cloud-edge architecture plays a relevant role from a technological perspective. Among these, artificial intelligence is prominent in all its declinations, including semantics and learning algorithms. A particular focus will be devoted to the topic of federated learning [8].

1.2 Aims and Scope

This thesis aims to explore, design, and analyze architectures that leverage cloud and edge computing integration to improve the efficiency [9] and performance of a diverse range of applications. The analysis of the architectures and their design and implementation aims to identify how the two technologies can collaborate synergistically to meet distributed and real-time data processing needs. Emerging issues in managing and analyzing large volumes of data generated in real-time by connected devices are effectively addressed. Therefore, the methodological approach adopted in this thesis is not limited to a mere theoretical discussion; instead, it includes a concrete analysis of the design and implementation of cloud-edge architectures in various application domains. The objective is to illustrate how cloud, edge, or a combination of both technologies can address specific challenges, enhancing performance and operational efficiency in sectors such as healthcare, defense, cultural heritage, and energy systems. In particular, the focus will be on the core application of artificial intelligence, which includes techniques such as machine, deep, and feder-

ated learning [10], as well as semantic [11] solutions and augmented and virtual reality (AR/VR) implementations.

Chapter 2

Background and Motivation

In this thesis, the work focuses on defining different architectures to support different technologies and case studies. In order to get as broad a view as possible, numerous topics have been explored along this path, among which a central one is the concept of Cloud Continuum. The latter represents the fulcrum on which different technologies can be implemented, including machine and deep learning, with particular reference to edge applications, such as federated learning. In addition, another topic related to artificial intelligence will also be addressed, namely expert systems, through the semantic approach. Finally the discussion will focus on virtual and augmented reality (VR/AR), concentrating on leveraging the edge cloud for real-time responses [5]. Before proceeding with the definition, implementation, and testing of the different architectures, the abovementioned technologies will be explored theoretically.

2.1 Cloud Continuum: from Core to Edge

The first two definitions of the cloud continuum concept date back to 2016. On the one hand, Harshit Gupta [12] defines the continuum as "a continuum of resources available from the network edge to the cloud/datacenter." On the other hand, Mung Chiang [13], [13],[14] offers a definition that focuses more on computational aspects, specifying where and how data processing takes place. In subsequent years, up to the present day, numerous definitions of "cloud continuum" have been proposed and published in various academic papers. Focusing on the two main approaches to defining this concept without listing them all in detail is useful. The first sees the continuum as a distribution of resources [15] among different network elements, including IoT, Fog, Edge, and even HPC (High-Performance Computing). The second approach, on the other hand, views the continuum as an extension of distributed computing capability across multiple nodes, often mentioning the possibility of running artificial intelligence (AI) [16].

2.1.1 Edge Computing

Edge computing is a broad and complex concept that has received several definitions over time. Gartner, for example, describes it as "a part of the distributed computing topology in which information processing occurs near the edge, that is, where people and devices produce or consume information" [17]. Edge computing contrasts with traditional cloud computing, where information processing occurs in large, centralized data centers, often geographically distant from end users and devices. Thus, edge computing aims to reduce latency and improve response times for end users [18]. The arrival of 5G has further reinforced the importance of edge computing. With the significantly higher data rates and more stable connections provided by 5G [19], more real-time data can now be processed directly at the network's edge. This synergy between 5G and edge computing increases operational efficiency and provides a smoother and more responsive user experience. Edge computing has two main visions in the context of 5G networks. The first sees it as "compute nodes" distributed within the network to process data as close to its origin as possible. The second interprets it as a virtualized extension of the cloud, often through technologies such as network functions virtualization (NFV) [20], enabling flexible and dynamic resource management. Both perspectives aim to improve the efficiency and speed of services within the 5G network. While the first vision emphasizes physical proximity between data and computing processes, the second focuses on virtualization and resource elasticity [21]. According to 3GPP [22], the integration of edge computing into mobile networks sytems from the need to reduce latency, improve response times for users, and reduce the load on the data network. In addition, edge computing can solve the problems of limited or unstable connectivity between the network edge and the central cloud.

2.2 Definition and History of Machine and Deep Learning

The most cited definition for machine learning is that of Tom M. Mitchell, who says, "A program is said to learn from experience E concerning some class of tasks T and with performance measurement P, if its performance on task T, as measured by P, improves with experience E" [23]. In other words, it means that we can speak of learning if and only if there is a measurable improvement in output after a task is performed. In the informatic context, machine learning is a sub-branch of artificial intelligence [24] Fig 2.1.

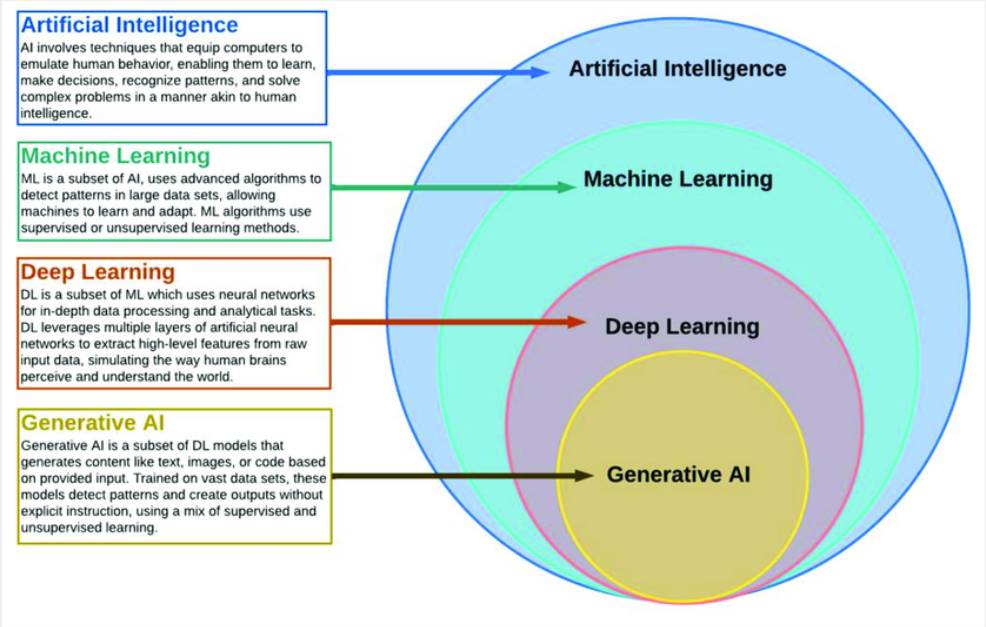


Figure 2.1: Chart that allows us to understand the hierarchy of the different strands of artificial intelligence

The first machine learning model dates back to the 1950s and can be traced to Arthur Samuel [25], an employee of IBM. He developed a program to train a computer to play checkers by analyzing future positions by the alpha-beta pruning algorithm. He used the minimax strategy to optimize moves, considering variables such as the number of pieces and the distance of the opponents. The machine estimated the probability of winning with each move made by the player. Also in the same period, specifically in 49, Donald Hebb, a Canadian psychologist, with his work "The Organization of Behavior" [26] proposed an embryonic structure of neurons based on nerve cell interactions that influenced later theories of AI algorithms. In the same years, Walter Pitts and Warren McCulloch developed the first mathematical models [27] of neural networks inspired by human cognitive processes. At the same time, Frank Rosenblatt designed the first hardware prototype, known as the "Mark I Perceptron" [28]. This device, built-in 1957 at Cornell Aeronautical Laboratory and funded by the Information Systems Branch of the U.S. Office of Naval Research and the Rome Air Development Center, was unveiled to the public on June 23, 1960. Shortly after that, the Raytheon Company developed the Cybertron [29], a machine that, with the intervention of a human operator, could be trained to recognize errors by a correction button. Also, during that period, Nilsson's work on pattern classification became a significant publication. Scientific interest in these issues remained constant over the next two decades, with studies such as Duda and Hart's 1973 study of Bayesian [30] models or the 1981 research proposing training strategies for artificial neural networks capable of recognizing 40 characters (26 letters, ten digits, and four

special symbols) from a computer terminal. Over the past decade, these technologies have grown significantly, aided by the increased availability of data and the emergence of so-called big data, which have created fertile ground for the spread of these technologies.

2.3 Types of ML: Supervised, Unsupervised and Semi-Supervised

By subdividing them by learning type, machine learning algorithms can be clearly classified [31]. According to this differentiation, we distinguish three different categories which are respectively:

- Supervised Learning: in this approach, models are trained through a series of inputs corresponding to an output labeled [32].
- Unsupervised Learning: in contrast, this type of training is done when providing the algorithm with input data without, however, specifying labeled output. In this case, it will be the algorithm itself that will identify patterns and relationships between the different data[32].
- Semi Supervised Learning in which the training set is partially labeled. In this approach, only part of the dataset is labeled. The algorithm uses both labeled and unlabeled data to improve classification by exploiting the distribution of unlabeled data to optimize predictions Fig 2.2 [33]

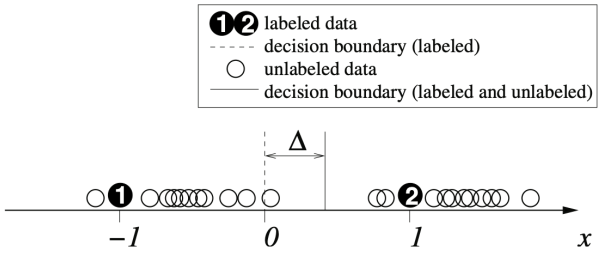


Figure 2.2: The image illustrates an example of semi-supervised data distribution, where labeled (filled circles) and unlabeled (empty circles) data are distinguished. The dashed lines represent the decision boundaries for labeled data, while the solid line indicates the decision boundary for labeled and unlabeled data.

After introducing the main categories of machine learning based on the learning process, we will examine each in more detail, examining the commonly used algorithms for each type.

2.3.1 Supervised Learning

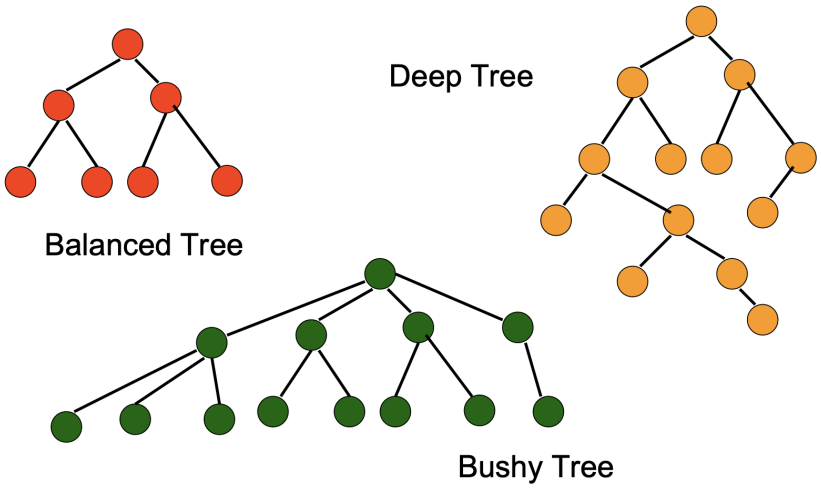
Supervised learning algorithms, in turn, can be divided into two other subcategories depending on their purpose, which are respectively: **Classification** [34], which allows approximating a mapping

function (f) from input variables (X) to discrete output variables (y). Such a type of supervised learning thus has a finite set of classes. If there are only two, we speak of binary classification, but if there are three or more, we speak of multiclass classification [35]. In addition, the classification can be linear or nonlinear depending on the boundary separating the two or more classes. **Regression** [36], which instead consists of approximating a mapping function (f) from input variables (X) to a continuous output variable (y). The most well-known algorithms that implement classification are: decision tree, support vector machine, k- nearest neighbors and the naive bayas while regarding regression we have linear and polynomial regression. In addition, it is essential to point out that some algorithms traditionally used for classification models with appropriate adjustments can be used for regression and vice versa [37, 38, 39, 40].

Classification

Classification algorithms are widely used in different fields, and some of the applications can be traced back to the healthcare field; this is the case with the classification of medical images such as X-rays [41] [42], CT scans, or MRIs [43] for the identification of tumors or other diseases [44]. Even in communications, classification algorithms are critical to filter emails from spam and inappropriate social content [45] [46]. Still, in culture and entertainment, such an approach can be used for personalization and subsequent recommendation of movies or music based on user interests [47]. These case studies represent only a small overview of the application of classification. As mentioned in 2.3.1 the algorithms are:

- **Decision Trees**: a structure used to divide a set of records recursively into smaller and smaller sets by applying simple decision rules. The structure of a decision tree has three types of nodes: **Root Node**: a node that possesses no input arcs but only output arcs, **Internal Node**: which possesses one input arc and n output arcs, and finally the **Leaf Node**: also with an input arc but no output arcs (terminal nodes). Examples of decision trees are shown in Fig 2.3.



.

Figure 2.3: Different types of decision trees are displayed conceptually. A balanced tree shows a uniform structure with the same number of nodes in each branch. On the right, the deep tree represents a configuration with multiple levels of depth, with branches extending vertically. At the bottom, the bushy tree shows a wider structure with more branches and nodes distributed horizontally, highlighting the variation in decision tree structures.

The construction of a decision tree usually takes place in two main stages: the first construction stage, in which the tree is recursively partitioned using the selected attributes; the second stage involves the removal of irrelevant branches to improve classification accuracy. Since constructing the optimal decision tree is computationally complex due to the exponential nature of the search space, efficient algorithms adopt greedy strategies to make locally optimal decisions about the attributes to partition the data. An example of such algorithms is Hunt's Algorithm, which forms the basis of approaches such as ID3 [48], C4.5 [49], and CART [50].

- **Support Vector Machine** (SVM), introduced by Vapnik in the work *Statistical Learning Theory* [51], is based on the idea of identifying the decision surfaces of a class rather than estimating the probability density. SVM began as a binary classifier and later extended to a multiclass perspective [52]. Several types of SVM are distinguished: linear and nonlinear, with separable and nonseparable surfaces. In the simplest case of a linear binary SVM with separable surfaces and a training set (TS) containing n samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_i \in \mathbb{R}^d$ represent multidimensional patterns and $y_i \in \{+1, -1$ are the labels of the classes, there exist various hyperplanes capable of separating the classes. A generic hyperplane is defined by the parameters $(\mathbf{w}, b)$.

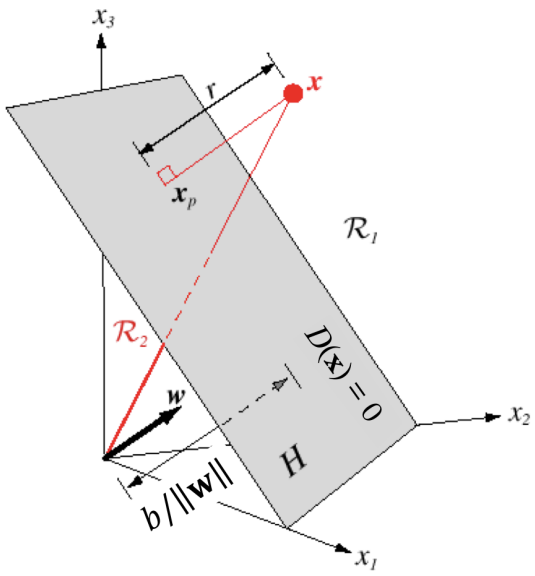


Figure 2.4: Graphical-mathematical representation of the support vector machine

The hyperplane is defined by the equation:

$$D(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b$$

where $\mathbf{w}$ is the vector normal to the hyperplane, $b/|||mathbfw$ represents the distance from the origin, and $D(\mathbf{x}) = 0$ describes the locus of the vectors on the plane. The hyperplanes separating the TS patterns, with a minimum distance $1/\|mathbfw$ on each side, satisfy the following inequalities for $i = 1 \ldots n$:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \quad \text{if} \quad y_i = +1$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \quad \text{if} \quad y_i = -1$$

or, in compact form:

$$y_i[\mathbf{w} \cdot \mathbf{x}_i + b] \geq 1 \quad \text{if} \quad i = 1 \ldots n$$

The shortest distance between the separating hyperplane and a pattern in the training set is called the margin τ.

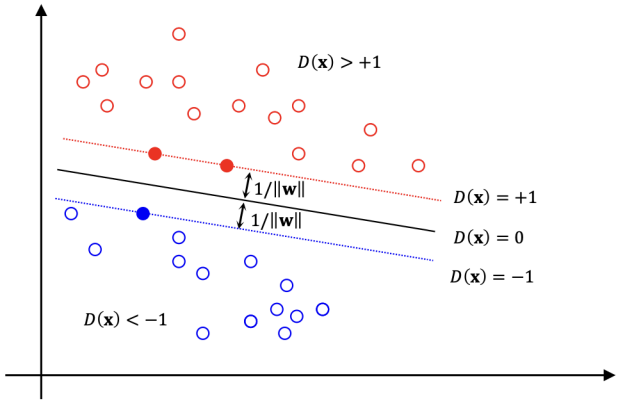


Figure 2.5: Margin between the support vectors and the separating hyperplane. The red and blue dots represent two distinct classes of data. The dashed lines indicate the decision boundaries, while the separating hyperplane is described by the black solid line

The distance of the points lying on the hyperplane $D(\mathbf{x}) = +1$ from the hyperplane of separation $D(\mathbf{x}) = 0$ è $1/\|\mathbf{w}\|$, the same applies to points on the hyperplane $D(\mathbf{x}) = -1$. Therefore, the margin turns out to be $\tau = \frac{2}{\|\mathbf{w}\|}$. The optimal hyperplane according to SVM is the one that maximizes the margin τ (or, alternatively, minimizes its inverse):

$$\text{Minimize:} \quad \|\mathbf{w}\|^2/2$$

$$\text{Constraints:} \quad y_i[\mathbf{w}\cdot\mathbf{x}_i + b] - 1 \geq 0 \quad \text{per} \quad i = 1\ldots n$$

The patterns in the training set that lie on the margin (the *support vector*, filled circles in Fig 2.5) are the ones that completely define the problem solution.

- The **Nearest Neighbor** (NN) classifier [53] assigns a pattern $\mathbf{x}$ to the same class as the closest sample $\mathbf{x}'$ in the training set (TS), according to a distance metric ($dist(\cdot)$). The basic idea is that if two points are close in multidimensional space, they are likely to belong to the same class, so it is assumed that the posterior probability is similar:

$$P(w_i|\mathbf{x}) \approx P(w_i|\mathbf{x}')$$

It can be shown that, in a training set with infinite samples, the probability of NN error is never worse than twice the minimum possible error (Bayes error P^*).

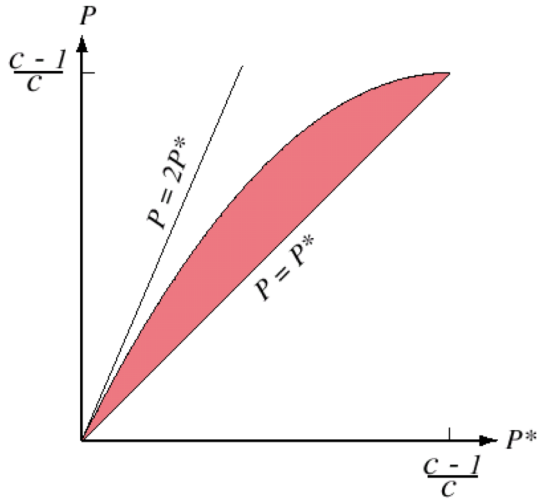


Figure 2.6: Error probability of the nearest neighbor rule compared to the minimum Bayes error

The NN classifier generates a Voronoi tessellation of space [54], where each sample in the training set creates a region (or tile) where all patterns are assigned to its same class. However, NN can be sensitive to *outliers*, as an anomalous sample can mislabel nearby patterns.

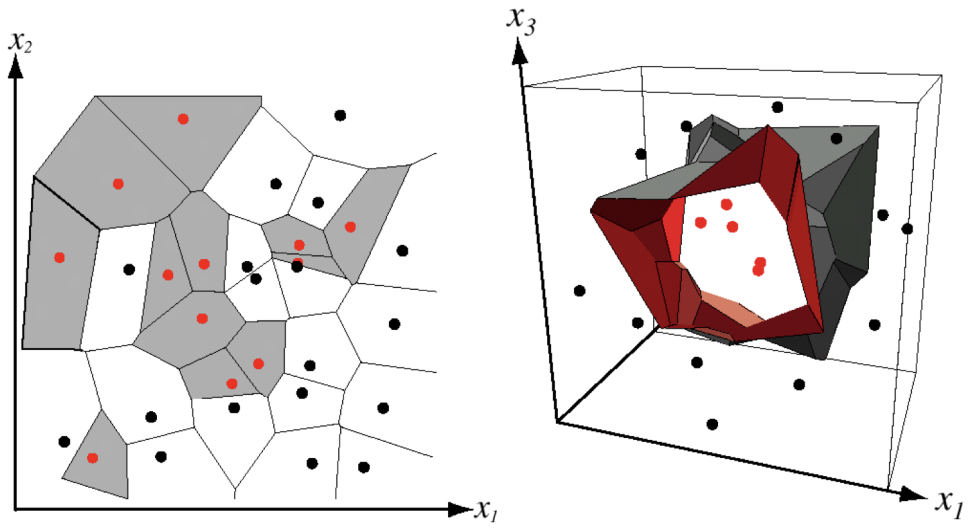


Figure 2.7: Graphical and geometrical representation of the Voronoi tessellation

To improve robustness, the *k-nearest neighbor* (k-NN) classifier is used, which classifies a pattern [55] **x** based on the k nearest neighbors. Each neighbor votes for its class, and **x** is assigned to the class that receives the most votes. From this approach, a probabilistic confidence in the classification can be extracted by calculating the proportion of votes for

each class relative to the total k votes:

$$\left[\frac{v_1}{k}, \frac{v_2}{k}, \ldots, \frac{v_s}{k}\right]$$

where $v_1, v_2, \ldots, v_s$ represent the votes obtained for each class and $\sum_{i=1}^{s} v_i = k$.

- The **Bayesian classifier** [56] [57] is based on Bayesian decision theory, which uses posterior probability to make classification decisions. Consider two states of nature (classes) ω_1 and ω_2, with prior probabilities $P(\omega_1)$ and $P(\omega_2)$, such that $P(\omega_1) + P(\omega_2) = 1$. The simple decision rule is:

  $$\text{Decide } \omega_1 \text{ if } P(\omega_1) > P(\omega_2), \text{ otherwise decide } \omega_2.$$

  Introducing an observable feature x, the conditional distributions of the classes are $p(x|\omega_1)$ and $p(x|\omega_2)$, which describe how the feature x varies in the classes ω_1 and ω_2. Bayes' rule allows calculating the posterior probability of the classes given x:

  $$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)},$$

  where $p(x) = p(x|\omega_1)P(\omega_1) + p(x|\omega_2)P(\omega_2)$. The decision is made according to:

  $$\text{Decide } \omega_1 \text{ if } P(\omega_1|x) > P(\omega_2|x), \text{ otherwise decide } \omega_2.$$

  The probability of error given an observation x is:

  $$P(\text{error}|x) = \min(P(\omega_1|x), P(\omega_2|x)).$$

  Generalizing with a loss function $\lambda(\alpha_i|\omega_j)$, which represents the cost of choosing action α_i when the state of nature is ω_j, the conditional risk is:

  $$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j)P(\omega_j|x),$$

  while the total risk is:

  $$R = \sum_{i=1}^{a} R(\alpha_i|x)P(x).$$

  To minimize risk, the action α_i is selected for which $R(\alpha_i|x)$ is minimal:

  $$\alpha_i^* = \arg\min_{\alpha_i} R(\alpha_i|x).$$

2.3.2 Regression

Regression differs from classification in that it uses values in a continuous rather than a discrete set. In more detail, regression aims to estimate a function that maps the values of the independent variables (x) to the values of the dependent variables. This technique can be used in different contexts: the healthcare environment is one in which it can be used to predict vital parameters [58] [59] in at-risk patients rather than blood glucose [60] [61] in diabetic patients. Industry, too; thanks to regression models, it is possible to predict sales or estimate potential economic losses [62]. Still, it can predict temperatures and weather conditions by exploiting historical data and variables such as pressure, humidity, and wind speed [63] [64]. These are just a few of the countless applications of regression. Going into more detail, however, there are different types concerning this family of algorithms. The most common ones are:

- **Linear Regression** [65]: a statistical technique that models the relationship between a dependent variable y and an independent variable x through a linear function. In formulas:

  $$y = \beta_1 + \beta_2 x + e$$

  Where β_1 represents the intercept, β_2 is the slope of the line, and e is the error term, representing the difference between the observed and predicted values. The parameters β_1 and β_2 are estimated using the least squares method, which minimizes the sum of the quadratic errors (e) to ensure that the line fits the observed data as closely as possible.

- **Polynomial regression** [66]: an extension of linear regression that allows more complex relationships between the dependent variable y and the independent variable x to be modeled. Instead of assuming a simple linear relationship, polynomial regression assumes that y can be expressed as a polynomial of degree M:

  $$y = A + Bx + Cx^2 + \cdots + Zx^M + e$$

  where $A, B, C, \ldots, Z$ are the coefficients of the polynomial to be estimated and e is the error term.

- **Logistic regression** [67]: this is an interesting case that allows the classification to be used as if it were a regression. So, instead of predicting continuous values, logistic regression predicts the probability that an observation belongs to one of two classes through a logistic (or sigmoid) function:

  $$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}}$$

where $\beta_0, \beta_1, \ldots, \beta_k$ are the parameters to be estimated and $x_1, \ldots, x_k$ are the independent variables.

2.3.3 Unsupervised Learning

In unsupervised learning, the goal is to detect hidden patterns without known outcomes or data labels [68]. The first difference from supervised learning is that the data are not labeled. One of the main unsupervised learning techniques is clustering 2.8.

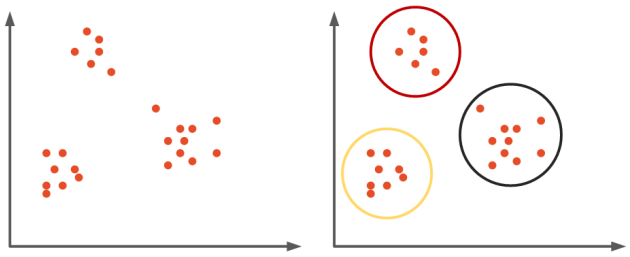


Figure 2.8: Result of a generic clustering algorithm, where three distinct classes can be isolated. On the left, the data are distributed without labeling. On the right, the three distinct classes are highlighted with colored circles, representing the algorithm's ability to group points based on internal similarities.

It has applications in many disciplines and has always received considerable interest. The main families of clustering are:

- **Hierarchical clustering**: where the goal is to aggregate patterns based on a measure of distance [69].
- **Clustering based on centroids**: clusters are identified by trying to minimize the distance of the patterns from the centroids of the clusters to which they belong through heuristic processes. Some examples of this category are: K-means, Fuzzy K-means [70], Expectation-Maximization (Gaussian Mixture) [71]
- **Density-based clustering**: in which the clusters identified are connected regions in high-density areas [72]. Of that family, one of the best-known approaches is DBSCAN [73]

In terms of application areas, however, they are different. One case study that can be mentioned is market segmentation, which uses clustering to identify the most homogeneous groups of customers and enables companies to create targeted campaigns. Using variables such as customer satisfaction or brand loyalty [74], customers with similar characteristics are clustered, or another application that can be mentioned is in the demographic field in which it is possible to classify territories based [75] on variables such as level of urbanity and population density. These are just some of the applications that demonstrate the potential of such an approach.

2.4 Definition of Deep Learning

Deep learning is a family of models characterized by multilevel neural networks [76] [77]. This technology, which attempts to mimic the human brain's decision-making process, is the basis of most artificial intelligence applications. The main difference between deep and machine learning lies in the complexity of the architecture of neural networks Fig 2.9.

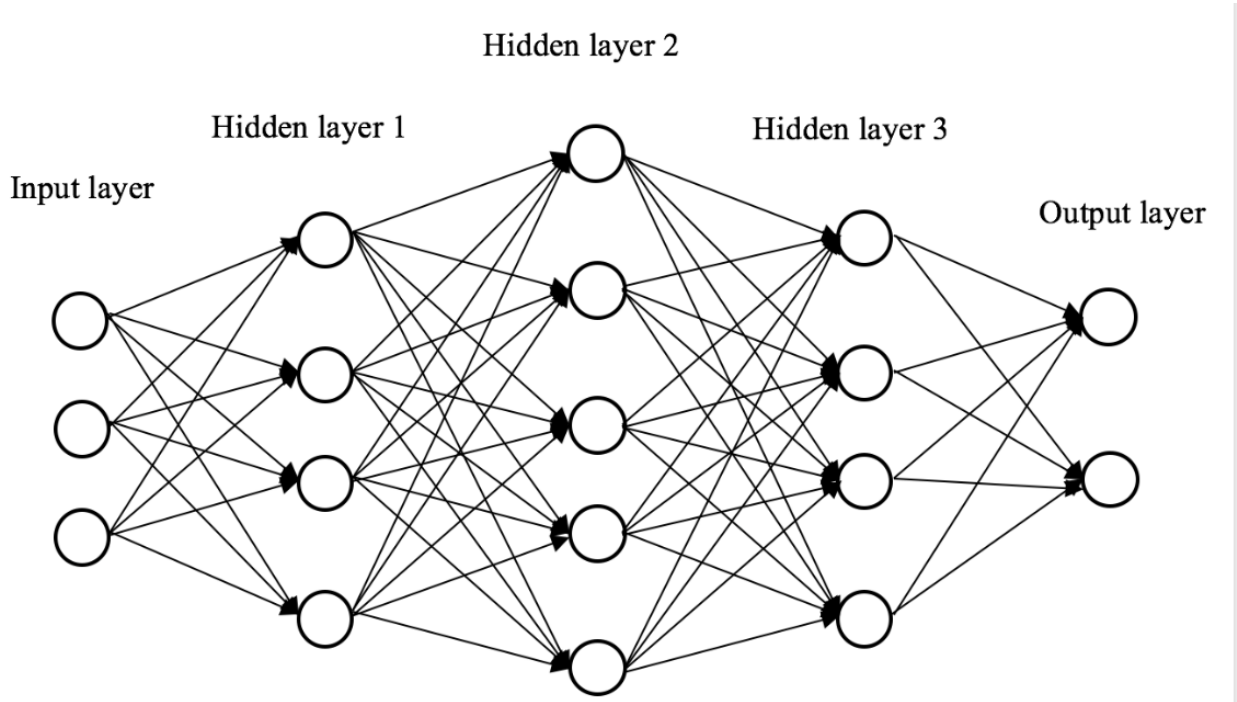


Figure 2.9: Architecture of an artificial neural network with three hidden layers. The model enables learning of complex representations through multiple layers of processing.

While the former uses one or two computational layers, the latter ranges from three and often reaches hundreds or thousands of layers. Deep learning takes advantage of unsupervised learning. This approach allows deep learning models to extract the features, functions, and relationships needed to obtain accurate results from raw, unstructured data. The applications in deep learning are endless, and different networks are defined as being more suitable or less suitable to perform specific tasks. So, in the next section 2.4.1, we will analyze the different networks and for each of them, its respective applications.

2.4.1 Typologies of Neural Networks

Deep neural networks have revolutionized the field of artificial intelligence [78], leading to significant advances in several areas. Among them, convolutional neural networks (CNNs), recurrent

neural networks (RNNs), and generative adversarial networks (GANs) are the most used.

Convolutional Neural Network

CNNs are optimized for solving problems involving data sets with spatial relationships in which rows and columns cannot be exchanged [79]. This is precisely why the most frequent use for these networks is with multimedia datasets, therefore photos, video [80], or audio [81]. Their architecture re-specifies the atomic operations they perform on data, such as essential feature extraction Fig 2.10.

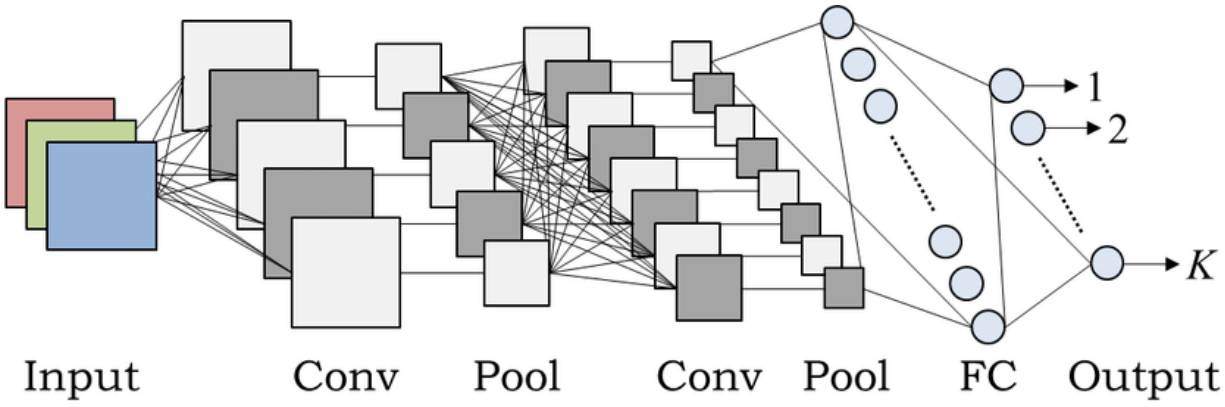


Figure 2.10: Architecture of a convolutional neural network (CNN) with convolutional, pooling, and fully connected layers for multiclass classification.

The initial, more straightforward features are progressively combined and transformed into more complex elements in the upper layers, which resemble real objects of interest, such as animals, buildings, or vehicles. Subsequently, these automatically created features are used for prediction tasks, such as object recognition in new images [82]. The uses of these networks cover all kinds of fields and are mainly used for classification. Some case studies of interest may be face recognition [83], which is used to identify and recognize faces in images or video streams. Another case study is the field of autonomous driving [84]. CNNs recognize different elements, including traffic signals, pedestrians, and other vehicles, helping vehicles make real-time decisions. Finally, such networks are also engaged in the medical field in different cases, among which one of interest is allowing the identification of tumors [85] [86] or other abnormalities in medical images [87], thus supporting the work of doctors.

Recurrent Neural Network

RNNs are a type of network used for sequential data with internal feedback mechanisms [88]. In particular, they are mainly used for time series. As for simple RNN architectures, these are little

used since they suffer from "short-term memory". The same cannot be said for more complex architectures, including long and short-term memory networks 2.11 (LSTM) [89] and gated recurrent unit (GRU) [90].

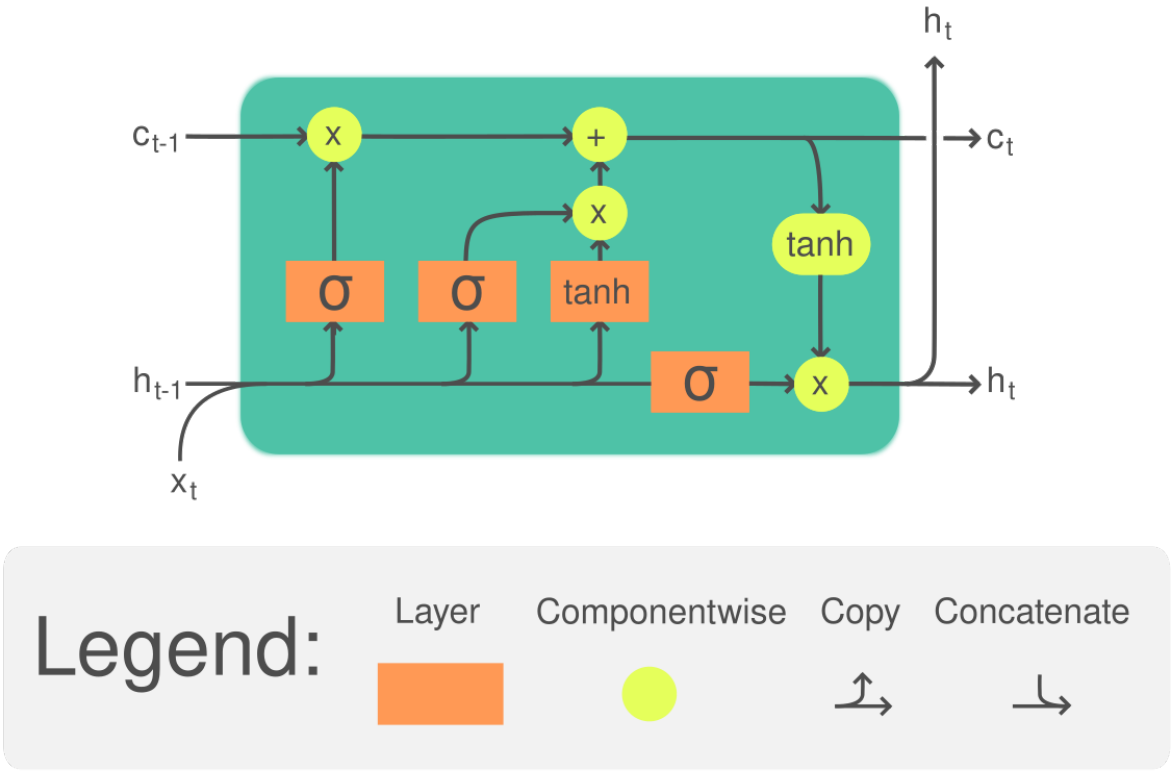


Figure 2.11: Structure of a Long Short-Term Memory (LSTM) unit with the input, output, and forget gate operations represented with the sigmoidal and hyperbolic tangent functions.

In particular, LSTM has achieved remarkable success, such as in developing OpenAI's robots that outperformed humans in the game Dota 2 or in robotics [91], where they control human-like robotic hands to manipulate physical objects with exceptional precision [92].

Generative Adversarial Network

Finally, another type of network that has found fertile ground in the last period, mainly due to the vast amount of data on the web, is generative adversarial neural networks that aim to train a network to generate new samples with some random variation [93]. Principally, GANs Fig 2.12 consist of 2 subnets. The first network is a generating network that captures the input distribution and generates new samples. The second instead is called a discriminator and aims to distinguish real examples from artificially generated ones. This type of network has given a technological jolt

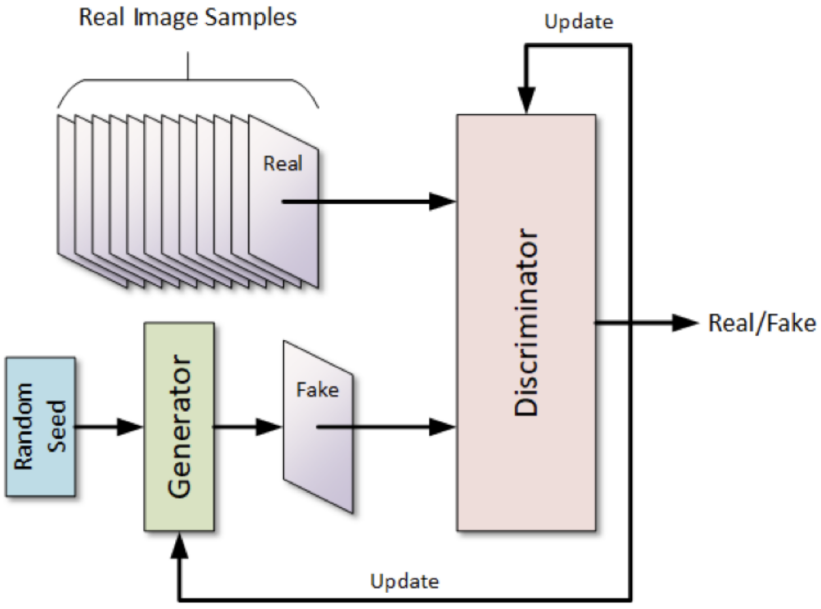


Figure 2.12: Diagram of a GAN (Generative Adversarial Network) showing the interaction between the generator, which creates false images, and the discriminator, which distinguishes between real and generated images.

in the coming years, especially with the advent of big tech and the ChatGPT (OpenAI) [94] and Gemini (Google) [95] software [96]. In this context, also connected to the work that has been done on the theme of storytelling in the 4.2 section, an analysis of the state of the art was conducted on the different tools and technologies of automated narration for cultural heritage also with use of generative AI. For completeness, the entire work, published in the conference ”Advances in Internet, Data Web Technologies (EIDWT 2024)”, is reported in appendix [97]. These networks can be used in different contexts, mainly to support activities such as image generation, image editing [98], or even code generation based on requests [99].

2.5 Federated Learning

Federated learning introduces a new paradigm for machine learning in which devices or nodes are trained locally [100],without transmitting raw data to a central server [16]. The only information that is shared is the weights of the local models that are later aggregated in the central server to refine the global model, as shown in Figure 2.13

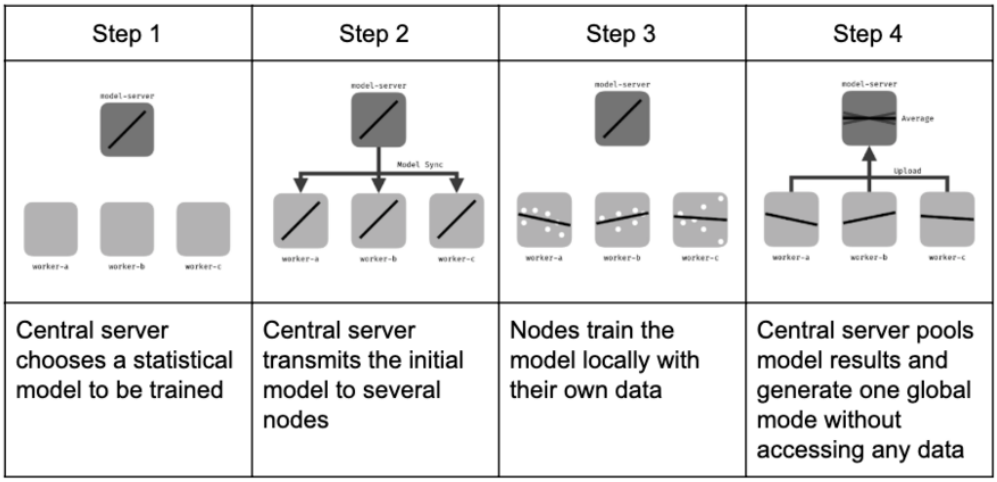


Figure 2.13: Federated learning scenario in which all data remains confined in IoT devices. In Step 1, the central server selects a statistical model to train. In Step 2, the central server broadcasts the initial model to different nodes. In Step 3, the nodes train the model locally using their data. Finally, in Step 4, the central server collects model results from nodes and generates a single global model without directly accessing local data.

This approach offers many advantages, especially in the context of edge computing. In more detail, the advantages associated with using this approach are as follows:

- **data privacy**: as mentioned above, data remains on local devices and does not have to be shared with a central server or other users [101].
- **Reduced need for data transfer**: because the data remains local, the amount of data that needs to be sent to a central server is significantly reduced. As a result, transmission costs and bandwidth impact are reduced, which within a home, especially in today's environment where IoT devices permeate every sector, bandwidth savings are essential for home network performance [102].
- **Resilience to centralized failures**: by distributing the learning process over many devices, the FL is less vulnerable to failures or problems caused by a single node.
- **Use of heterogeneous data**: FL can leverage data from multiple sources to create more robust and generalizable models, significantly improving the performance of a local model
- **Scalability and Flexibility**: unlike a global model, the computational load is distributed both in terms of the clients because they each train their data, and for the server, which, unlike the centralized approach in the federated one, only has to deal with the aggregation progress and the sending and receiving weights .

Despite its many advantages, federated learning suffers from challenges and problems inherent in the methodology. Let us first analyze in detail what they are about the context of the case study and then, in the following sections, discuss how we can mitigate them through patterns.

- **management of data heterogeneity**: the diversity and disparity of data across different nodes in quality and quantity can lead to challenges in learning a consistent and performing model.
- **Security risks**: although it improves privacy, federated learning is still subject to security problems. Mainly related to poisoning attacks or model reconstruction (thus house activity), but also to snooping or sniffing attacks and, more generally, to client IP identification (house location) [103].
- **Computational Costs**: training performed on IoT devices that, by definition, have computational resources. Training models on devices with limited resources (such as cell phones) can be costly in terms of time and energy, potentially affecting device performance.
- **Complexity of coordination and communication**: this represents one of the most significant challenges in federated learning, especially when it involves an extensive network of distributed devices. IoT components in different geographic locations may have different connections and, consequently, different execution times that could negatively impact the model's overall performance [104].

2.6 Expert Systems, Ontology and OWL

In the field of artificial intelligence, expert systems are tools that can replicate the capabilities of an expert in a given domain [105]. They can generally solve complex problems through inductive and deductive logical inference procedures. Unlike the aforementioned deep learning networks, expert systems are known for their ability to make clear all the decisions that led to that particular choice. Therefore, we no longer speak of a black box but a glass box. In general, we can classify expert systems into different categories, among which we have rule-based expert systems that use rules such as "If-Then" to arrive at new inferences. Applications of this type of expert system range from the medical field to the military. There are different case studies we can mention, such as rule-based approaches for applications of weapon systems [106] rather than for suggesting treatments based on the characteristics of a patient [107]. The second type of expert system is based on trees that classify data based on a decision tree [108]. Each node represents a possible choice, and eventually, a conclusion is reached based on the data provided. Regarding the general structure of expert systems, these are composed essentially of two parts: knowledge base, a repository that

contains within it all the information and rules needed to solve a specific problem and an inferential engine that represents the software component that is responsible for processing the information through the rules to provide a solution [109]. Finally, like all systems here, it is necessary to have a user interface to enable the individual client to interact with such a system. Ontologies in the context of expert systems and, more generally, in the semantic web are a crucial issue. These allow knowledge to be managed in a structured and formal way. An ontology, then, is a detailed description of concepts and relationships within a given domain through these tools and thus possibly Define only knowledge relevant to a given domain while ignoring all others, thus representing a way to define the contents of various sources formally. Among the languages used for creating ontologies, one of the most important is Ontology Web Language (OWL). It has been developed to extend the RDF schema by providing a richer vocabulary to describe entities such as class properties and relationships between different concepts.

2.7 Collaborative AR/VR in Cloud Continuum Environment

Augmented and virtual reality (AR/VR) are two technologies gradually transforming how people interact with the world and digital [110]. In recent years, major corporate players have brought increasingly high-performance headsets to the market with increasingly faithful reconstructions of reality.

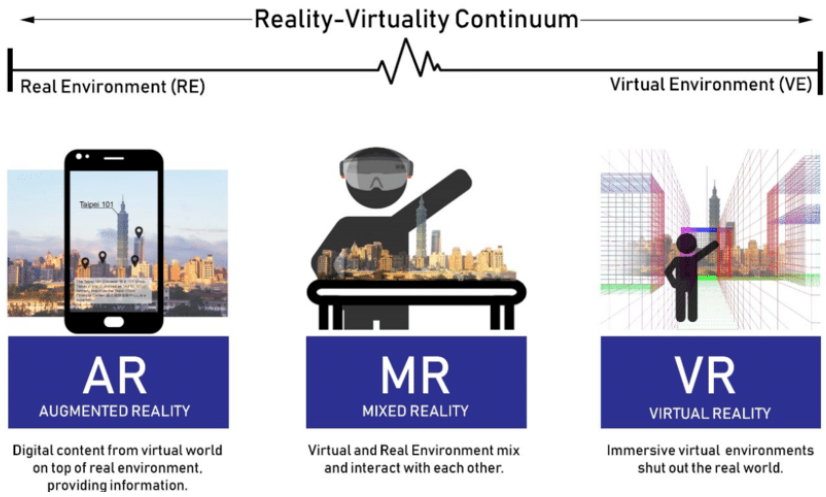


Figure 2.14: The reality-virtuality continuum showing the relationship between Augmented Reality (AR), Mixed Reality (MR) and Virtual Reality (VR) based on the degree of interaction between the real and virtual environment.

The difference between the two technologies is that while VR creates fully immersive experiences isolated from the real world, AR overlays digital content on top of the real world, enhancing

interaction with the surrounding environment [111][112]2.14. These technologies are widely used in education, entertainment, and healthcare rather than the military. However, with the increasing data complexity of digital content, high computational resources are required, and in these real-time contexts, latency is a critical factor [113]. A very interesting topic in this context is collaborative augmented and virtual reality. In other words, the possibility of interaction by multiple users on virtual objects or shared environments. For computed real-time only, the cloud is insufficient for current systems. It uses a combination of cloud and edge computing to manage the synchronization of environments and virtual objects between different users. Numerous works have dealt with defining different methodologies and approaches for the synchronization of different users [114].

2.8 Reference Architectures

Since the objectives of this thesis work include defining different reference architectures, before designing and implementing them, it was necessary to review the state of the art and the different reference architectures to understand theoretically and generally how to design the architectures and what the guidelines are. This analysis of the different architectures culminated with a conference paper published at Advanced Information Networking and Applications (AINA 2024) in which a review of some of the cloud continuum architectures in contexts of 5G usage. Well within this section, we will therefore go on to report the most exciting architectures analyzed during the state of the art exploration. The architectures are presented below taking an extract from the work already written titled ”Survey on Reference Architecture for Cloud Continuum and Multi-access Edge Computing (MEC) in 5G Networks” [115]:

- The National Institute of Standards and Technology (NIST) has defined a reference architecture for the cloud continuum, which identifies the major players, their activities, and functions within cloud computing (see Figure 2.15) [116, 117].

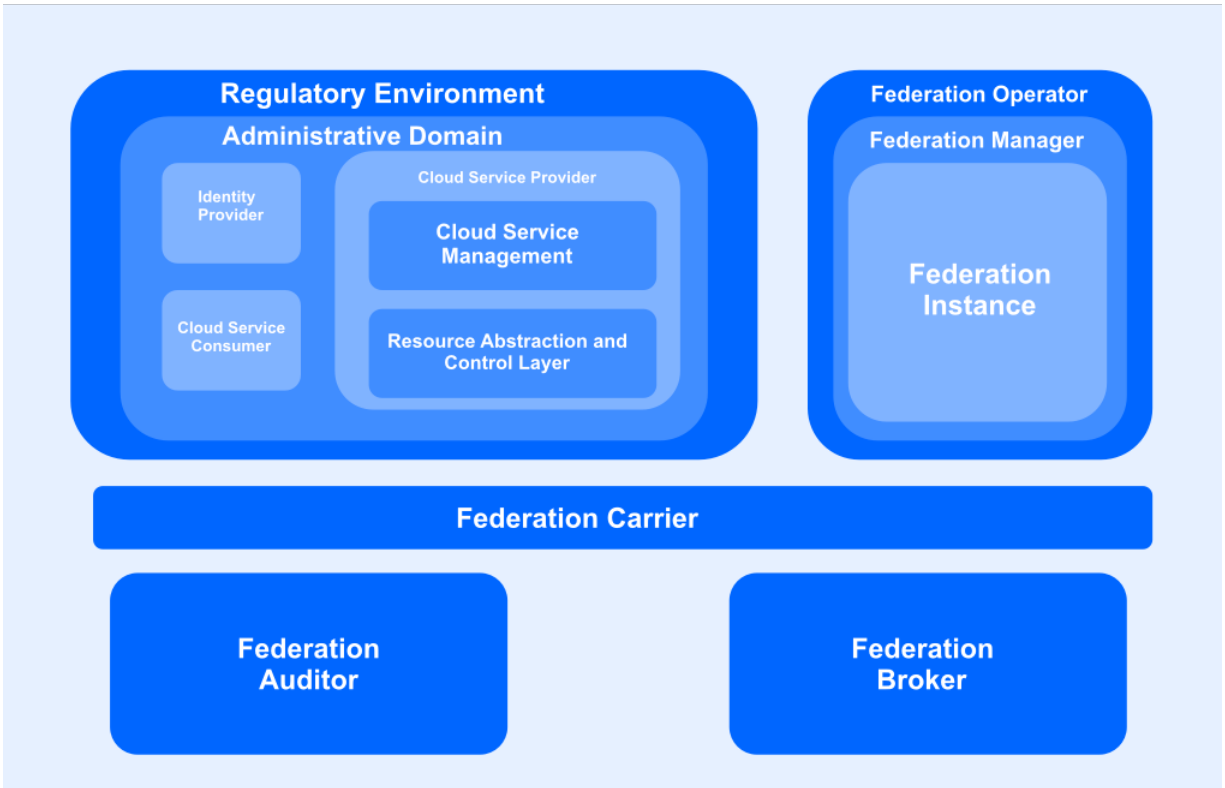


Figure 2.15: Overview of the NIST Reference Architecture for Cloud Federations. The diagram illustrates the regulatory environment's main components, including the administrative domain with cloud service providers, cloud service management, and cloud service consumers. Also depicted are the roles of the federation manager, federation operator, federation broker, federation carrier, and federation auditor, who work together to ensure that the federation operates and is governed in compliance with applicable regulations.

Before going into details, it is necessary to analyze the concepts of Administrative Domains (AD) and Normative Environments (RE), as they are fundamental in this cloud federation model. Generally, the basic authentication and authorization process takes place within an Administrative Domain, consisting of an Identity Provider (IdP), a Cloud Service Provider (SP), and a Cloud Service Consumer (CSC) or user. The IdP issues identity credentials to the consumer, and when the consumer requests a service, the SP verifies the credentials with the IdP and then decides on access. All administrative domains operate within a regulatory environment. Other relevant actors in the NIST architecture include:

1. Federation Manager, responsible for managing the federation;
2. Federation Operator, who facilitates the operations of the Federation Manager;
3. Federation Auditor, the independent third party who assesses policy compliance;

4. Federation Broker, an intermediary between federations that provides functionality such as brokerage and service aggregation;
5. Federation Carrier provides connectivity and cloud service transport between consumers and providers.

This architecture ensures enhanced interoperability for shared operations, such as military operations, and security is ensured through strict authentication and authorization standards .

- Fl Edge is an open-source system for Industrial IoT based on an extensible microservice architecture. Its reference architecture, shown in Figure 2.16, is divided into: **Plugins** (components with blue background) are modules that extend Fledge's functionality and are written in Python or C++. **Microservices** (components with blue background), which can be deployed in single or multiple environments.

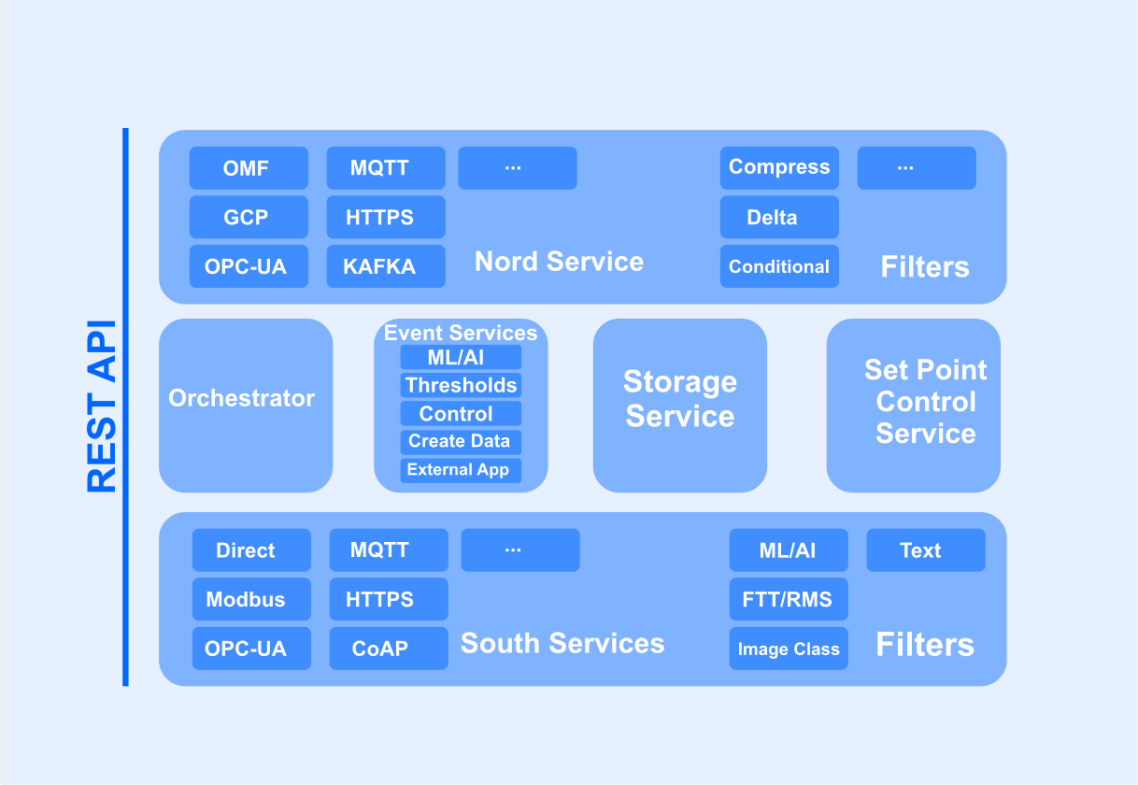


Figure 2.16: The architecture of the FI Edge. The diagram shows the main components of the edge infrastructure, including the North and South services, which manage the integration of various protocols and communication with devices through services, respectively. The orchestrator coordinates the interaction between the different services. All these components are accessible through a REST API interface for centralized management.

Key components of the architecture include:

1. the Fledge Core, which coordinates key functions such as configuration management and scheduling;
2. the Storage Layer, which provides flexibility and scalability;
3. the South and North Microservices, which connect Edge devices and larger systems by facilitating data transmission;
4. the Filters and Pipelines, which allow customization of data streams;
5. the Event Service, which responds to predetermined rules for monitoring.

Fl Edge is a modular platform with REST API and GUI, offering advanced data management and processing solutions, effectively connecting edge devices and more complex systems.

- The architecture of KubeEdge, shown in Figure 2.17, is divided into three layers: cloud, edge, and device. At the center of the cloud is the Kubernetes master (left) and the CloudCore (right), which includes the EdgeController and DeviceController. This process information from the control center is transmitted via the Cloud Hub to the EdgeHub in the edge section.

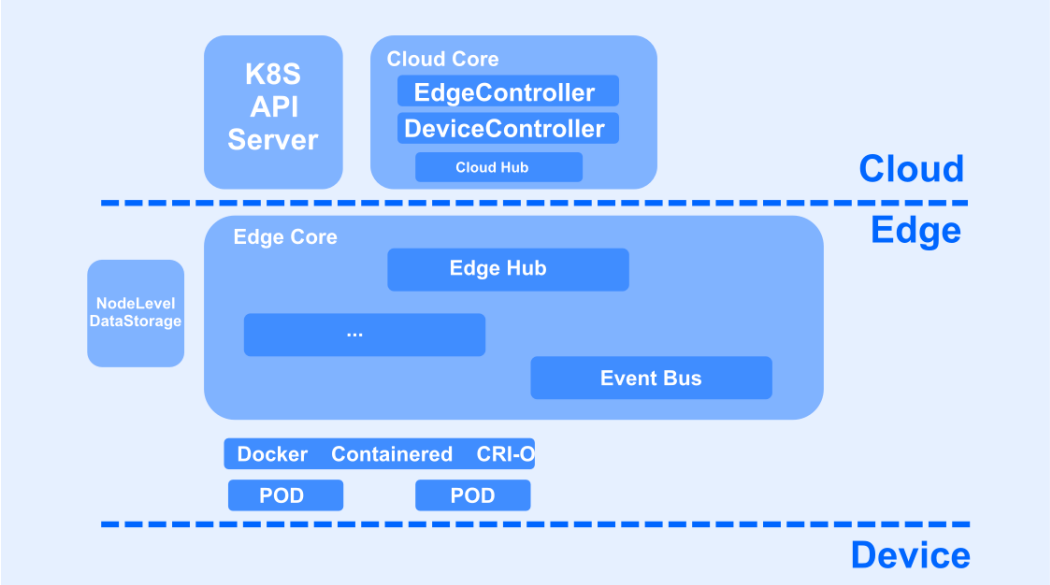


Figure 2.17: KubeEdge architecture diagram showing the split between the cloud, edge, and devices in the KubeEdge platform.

The edge layer manages applications and devices, and the EdgeHub receives stored and forwarded information to the device. This allows Edged to access metadata and ensure continuity of service even if the edge node is disconnected.

- The reference architecture for the Cloud Federation, proposed by the Future Cloud Cluster (a working group of the European Commission), is divided into three levels (see Figure 2.18):

  1. The Edge Computing layer provides low-latency computational and storage capabilities, enabling near-real-time data processing.
  2. The Cloud layer provides interoperability among cloud components.
  3. The Federation Management layer deals with the orchestration of resources and services between cloud and edge.

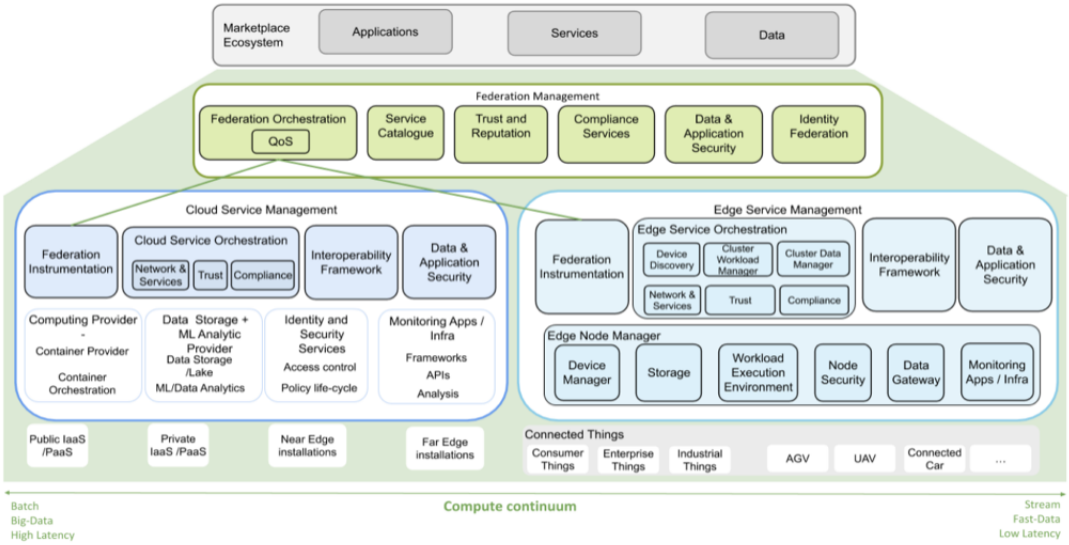


Figure 2.18: Future Cloud Federation Reference Architecture of the Future Cloud Cluster

The architecture enables the creation of clusters of edge devices, including nested ones, and provides a framework for management, device discovery, workload allocation, data management, networking, compliance, interoperability, and security. This architecture emphasizes the importance of holistic federation management.

Chapter 3

Methodology

The methodology applied to achieve the objective above defined in the section 1.2 of the chapter 1 will be to delineate different case studies in various domains and with different technologies among those mentioned within the chapter 2. After defining the goal for each case, different methodologies supported by reference architectures will be determined depending on the case study. Moreover, for each of them, we will not only define the architecture from a theoretical point of view but also follow an implementation and testing phase to validate its real use. The case studies have been preliminarily divided by technologies for greater reader clarity. In particular:

- **Semantics and Expert Systems**: two main case studies will be discussed in this section, the first one aiming at building a tool for automatic story composition. The tool has been realized for the European project RASTA. After defining a methodology, a supporting architecture was realized and validated through implementation and testing. In addition, based on this first work, a methodology to automate the creation of the stories based on the texts will also be proposed. The ultimate goal of the second work is to implement an expert system to calculate the probability that a generic patient is affected by COVID-19 [118] and, in this case, suggest a treatment based on the patient's characteristics. Therefore, after mentioning the methodology composed of probabilistic parts through Baesyan rules and semantic approaches to recommend treatment, a supporting architecture will also be realized here, which will then be implemented and validated through a testing phase.

- **Machine, Deep, and Federated Learning**: This section will discuss different case studies walked through and undertaken throughout the PhD. The first case of our interest that we will share is a work done in collaboration with ENEA to create a tool capable of scraping within the web and through Deep Learning algorithms, including NLP [119] or sentiment analysis [120] could search for events related to the energy community and give them weight. Such work is supported by an architecture that has been implemented and tested. The second

work, on the other hand, addresses the military theme by defining a supporting architecture for a federated supply chain. The third work was done at the National Institute of Artificial Intelligence in Barcelona during a personal research period abroad. This work carried out a methodology followed by the implementation of machine and federated learning algorithms to support a case study on air quality where it was necessary to perform clustering techniques for pattern recognition. Once again, this work was supported by an architecture for deploying such an algorithm. Finally, the last work aims to define different scenarios and algorithms for predicting and monitoring diabetic patients. Here, after carrying out a study concerning the choice of algorithm and its calibration, different centralized, federated, and local scenarios were defined for each architecture on which the code has been deployed and tested.

- **Augmented Virtual and Hybrid Reality**: This last topic was addressed in the last year of the PhD program and culminated in the definition of three different reference architectures to support different augmented and collaborative reality fruition scenarios. Different case studies were then deployed to validate the use of each architecture. Different solutions will be proposed to support case studies in detail, such as object recognition through pre-trained learning models or even virtual and augmented reality collaborative fruition scenarios to collaborate within a shared space. The work will not be limited to the mere denizens of different architectures, but it will also extend to their testing and validation in the real world through different collaborations that have allowed us to do different live demos, in particular, the former at the Central State Archives in Rome and the latter at the Bologna fair WeMakeFuture at a MiC stand [121].

Chapter 4

Semantic Technologies and Applications

This chapter will focus on defining two architectures for distinct purposes in the semantic field. In the first case, the goal is to outline the main features of an expert system designed to estimate the probability that a patient has contracted COVID-19 and provide personalized treatment recommendations based on the patient's specific characteristics. After a detailed description of the features and requirements, an architecture will be presented, implemented, and tested. In the second case, the goal is to define a methodology for graphical representation of narrative texts. To do this, the types of nodes will first be identified based on what has emerged from the existing literature, and then a system will be designed to leverage the realized graphs to map them into a single integrated ontology. To this end, a tool will be developed to manually create stories and convert all node connections into ontological relations. This will be supported by an architecture divided into different components, each implemented using specific technologies. After design, such a system will be implemented and tested. Both works have been submitted for publication in different scientific journals (respectively for Embedded Systems and Software: Practice and Experience)and are currently being evaluated. In addition, a methodology published at the International Conference on Emerging Internet, Data and Web Technologies [122] will be proposed to automate the generation of graphs from texts using natural language processing (NLP) techniques. Below, the work will be exhibited, and text excerpts from the three papers will also be taken.

4.1 Covid-19 Expert System

SARS-CoV-2 has deeply scarred our lives forever, generating numerous physical, psychological [123] and economic problems [124] . The ease with which it was possible to become infected and the silent behavior of the virus in many individuals who had contracted the disease is what made it such a fearsome disease. Containment measures were tried to reduce the infection however

these only partially mitigated its effect. Another issue not to be underestimated has been that of treatment. This was because many of the drugs and therapies used normally for the same symptoms were not effective. Building on the above, this work aims to create an expert system that can help citizens recognize and combat this disease. To do this it will then be necessary to define and implement an architecture capable of supporting this expert system. The work is therefore organized as follows: after defining objectives and methodology a system architecture will be delineate followed by implementation and validation of proper functioning. Finally, in the last section, the results obtained will be discussed.

4.1.1 Objective

The main objective of this work is to create an expert system that can support citizens in recognizing and combating SARS-CoV-2 disease and a hypothetical other highly contagious virus. Going into more detail, this project aims to suggest to a given patient the drug treatment to take if he or she had a certain probability (calculated using Bayesian rules) of having contracted SARS-CoV-2 disease to prevent serious harm caused by this disease. To do this, it is necessary to first define a clear methodology and then implement it to verify that it works properly.

4.1.2 Methodology

To achieve the goal of this project, several Ontologies and OpenData were considered, which formed the basis for the final result. The process was divided into several steps, which are outlined below. Initially, a conceptual model was created to schematically and organize the path to be followed to achieve the goal. The project was structured in two parts:

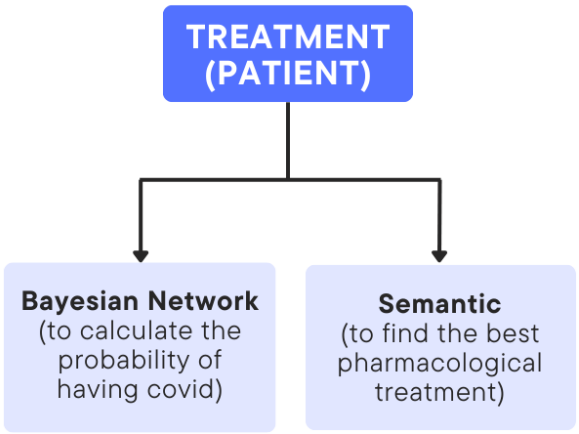


Figure 4.1: logical diagram of the problem methodology

1. The first part is based on a probabilistic approach, which involves creating a mathematical model based on Bayesian rules to calculate the probability that a person is affected by COVID-19.

2. The second part, which is more semantic, aims to identify the treatment to be administered to the patient if the probability of infection is high.

As shown in figure 4.1, the Bayesian network returns a probability of contracting COVID-19, while the semantic aspect suggests the most appropriate drug treatment. The "Treatment(Patient)" element provides the specific treatment for the patient if the probability calculated by the Bayesian network exceeds a certain threshold.

4.1.3 Data Collection

The first step was to search for ontologies to be used as the basis for the objective set. For this, we started with work carried out by Biswanath Dutta and Michael DeBellis [125], who provided an ontology for analysing and collecting data on COVID-19. The ontology itself also refers to other ontologies, which will be discussed in more detail in the next chapter . After collecting the ontologies, the next step was to search for the data to be inserted to populate them. Again, thanks

to the work mentioned above, it was possible to identify data to be inserted that would fit with the ontologies used. Data were then sought for the medical aspect.

4.1.4 Bayesian Network

As mentioned above, a mathematical model based on Bayesian rules [126] is used to calculate a person's probability of having COVID. In this chapter, we will see how these Bayesian rules are used. To calculate the percentage that a patient has COVID, four events are taken into account: A→ event related to having COVID, B→ event related to symptoms, C→ event related to close contacts the patient has had with positive people, D→ event related to (not close) contacts the patient has had with positive people. Having said this, we can now analyse the rules:

$$P(A|B,C,D) = \frac{P(A) \cdot P(B,C,D|A)}{P(B,C,D)} \tag{4.1}$$

(1) represents the probability of a person having COVID, taking into account the three events B, C, and D. To obtain our conditional probability. However, further calculations are needed because not all probabilities are available. Thanks to our data in the ontology, we have the prior probabilities of the four events available: P(A) → is a prior probability of having COVID-19, determined by an outbreak factor of the patient's location. In particular, therefore, if the patient resides in a location where there is an outbreak or has recently traveled to a location where there is an outbreak, the a prior probability of A will be higher; P(B) → a prior probability of having symptoms, again determined by the location where the patient resides and thus related to the influenza factor, i.e., whether influenza is circulating in that location; P(C) → a prior probability of the patient having had close contact with positive persons, determined by the patient's social activities; P(D) → a prior probability of the patient having had contact with positive persons determined based on the latter's social activities. Given these known probabilities, it will be possible to calculate :

$$P(B,C,D|A) = P(B|A) \cdot P(C|A) \cdot P(D|A) \tag{4.2}$$

$$P(B,C,D) = P(B) \cdot P(C) \cdot P(D) \tag{4.3}$$

For the calculation of (2), we need the following probabilities

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)} \tag{4.4}$$

$$P(C|A) = \frac{P(A|C) \cdot P(C)}{P(A)} \tag{4.5}$$

$$P(D|A) = \frac{P(A|D) \cdot P(D)}{P(A)} \tag{4.6}$$

In particular, (4) represents the conditional probability of having B symptoms if one has COVID. We can estimate P(A|B) with uncertainties, e.g. having B=1 symptom brings a probability of having COVID equal to X, if B=2 symptoms then the probability of having COVID is equal to Y, with Y>X and so on. Similarly, the same is done for (5) which represents the probability of having been in close contact with C positive persons if one has COVID and with (6) which instead indicates the probability of having been in contact with D positive persons given COVID. We can summarize what we said with this logical scheme 4.2.

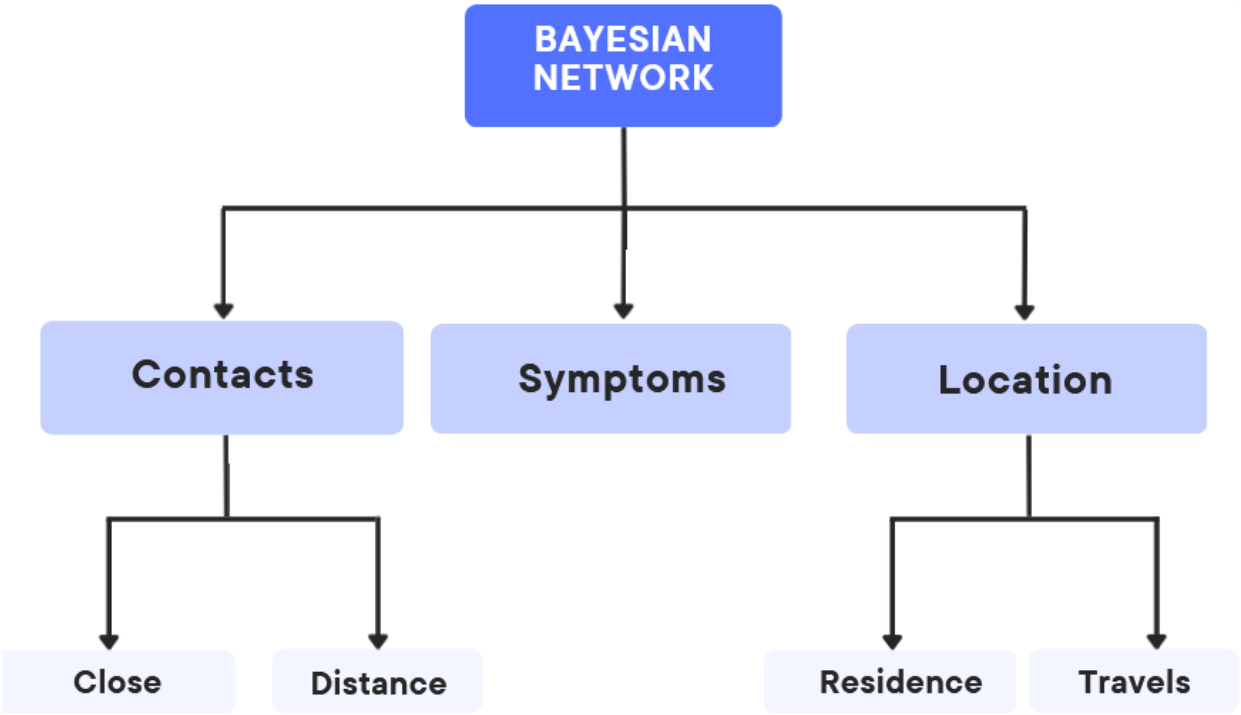


Figure 4.2: Logical scheme of the Bayesian network indicating the events taken into account

As we see, the Bayesian Network considers the patient's symptoms, contacts had (trying to distinguish between close and distant contacts) and the location where the patient resides or has travelled.

4.1.5 Semantic

In this subsection we will broadly analyse the more semantic part related to finding a cure. Once a patient's percentage of having COVID is established, if it exceeds a certain threshold, the system

goes on alert, sending the patient a cure, which must respect several factors, such as whether the patient is allergic to a particular drug, whether a certain drug is not contraindicated (he or she may have trauma from previous illnesses that do not go well with a specific drug), whether he or she is a patient at risk (non-vaccinated patients, the elderly and patients with various previous traumas are therefore considered in this category) and therefore needs particular types of drugs. BMI is also considered when estimating the appropriate dosage of medication in milligrams(mg) to administer. Thanks to the population of our ontology, we know everything about the patients, so both biographical and clinical data. This allows a reasonably detailed analysis of the treatment to be suggested. What is done is to search our ontology for a drug such that:

1. the patient must not be allergic to that medicine
2. the patient must not have contraindications caused by previous illnesses for that medicine:
3. if the patient already has respiratory problems, a more suitable medicine is sought for that;
4. based on a risk factor (e.g. the patient's advanced age, the number of previous pathologies, whether or not he/she has had a correct vaccination), one determines whether this medicine should be 'heavy' or not, to assign heavier drugs only to patients who need them, while also trying to optimise a possible distribution.

Once the most suitable drug has been found, the dosage in mg is calculated, and how many days to administer it is determined. This information is taken from the data sheets for each drug. It is essential to point out that several drugs may be recommended to the patient, in which case only one is chosen based on a priority factor. In addition, the dosage in mg will consider factors such as BMI and age. In the figure below 4.3, we have summarized what we have described before. This is only a logical scheme that will be better analyzed in the next chapter. As we can see, we get our medicine checking on different aspects, such as a check for the allergies that the Patient can have (to avoid those medicines that can contain elements dangerous for the person), a check for the Risks and the comorbidities, so that at the end we get only the best Medicine for our Patient. Finally, we determine the appropriate dosage.

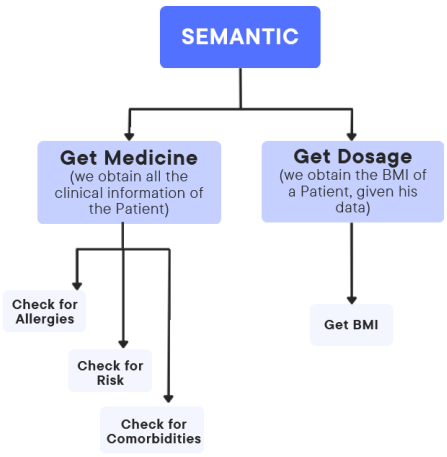


Figure 4.3: Logical scheme of the Semantic part

4.1.6 Implementation

In the previous chapter, the methodology for carrying out this work was described, analysing the different steps in a non-implementative manner. In this chapter, we analyse the procedures performed more technically, attempting to provide more low-level details of the work done. Inference for semantics and probabilistic calculation will be performed using the Prolog language. It is important to say that in order to do inference with Prolog, we first transformed the ontology with its imported data into Prolog facts via the Thea library [127]. The Prolog facts were then loaded into the Prolog file, where we created our rules with a consult. In both sections, i.e. for the probabilistic calculation and for the search for the cure, rules defined in Prolog were used. These rules were first represented using a Call Graph to get an idea of how to use and compose them, and then written in the Prolog language. More generally, the expert system's functioning for managing and treating COVID-19 develops through different interconnected phases. Each of them plays a crucial role in the overall process of diagnosis and therapy. Let's see the workflow in detail 4.5, 4.4:

- **Initial Analysis**: The system uses a Bayesian network to calculate the probability that a patient is infected with COVID-19 (*Bayesian Network Analyzer*). This operation is carried out for each individual patient registered within the system (*Authenticator*).
- **Notification to the Patient**: If the probability of infection exceeds a predetermined threshold, the system sends a notification to the patient (*DiagnosticUI*), informing him/her of his/her probability of being infected and suggests going to the doctor as soon as possible for a check-up.
- **Treatment Search**: After sending the notification to the patient, the system proceeds to the next phase, in which the most suitable drug for the treatment (*BestCure Engine*) is searched

for the specific patient (e.g., age, previous illnesses, chronic illnesses)

- **Dosage Calculation**: After selecting the drugs, the system calculates the optimal dosage for the individual drug (*DoseCalc Module*), always based on the patient's characteristics
- **Communication to the Doctor**: Once the appropriate drug and dosage have been identified, the system generates a message that is sent to the patient's doctor (*DiagnosticUI*). This message contains all care tips.

Finally, in both schemes, you can see that two different databases are used. This choice is driven by the need to separate sensitive information on individual patients (Info Database) from non-sensitive information on drug names and dosages (Medicine Database).

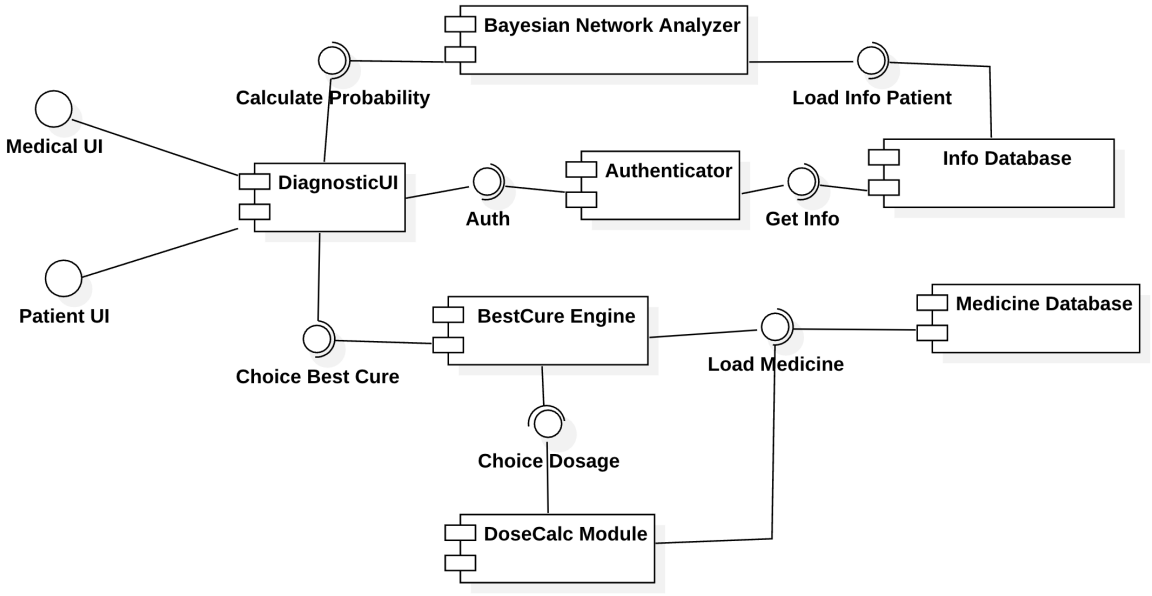


Figure 4.4: Diagram of the components of the expert system for individualized detection and treatment of COVID-19 patients, illustrating the interactions between the various modules such as the diagnostic interface, authenticator, optimal care engine, dose calculation module, and medicine and information databases, supported by a Bayesian network analyzer for diagnostic probability.

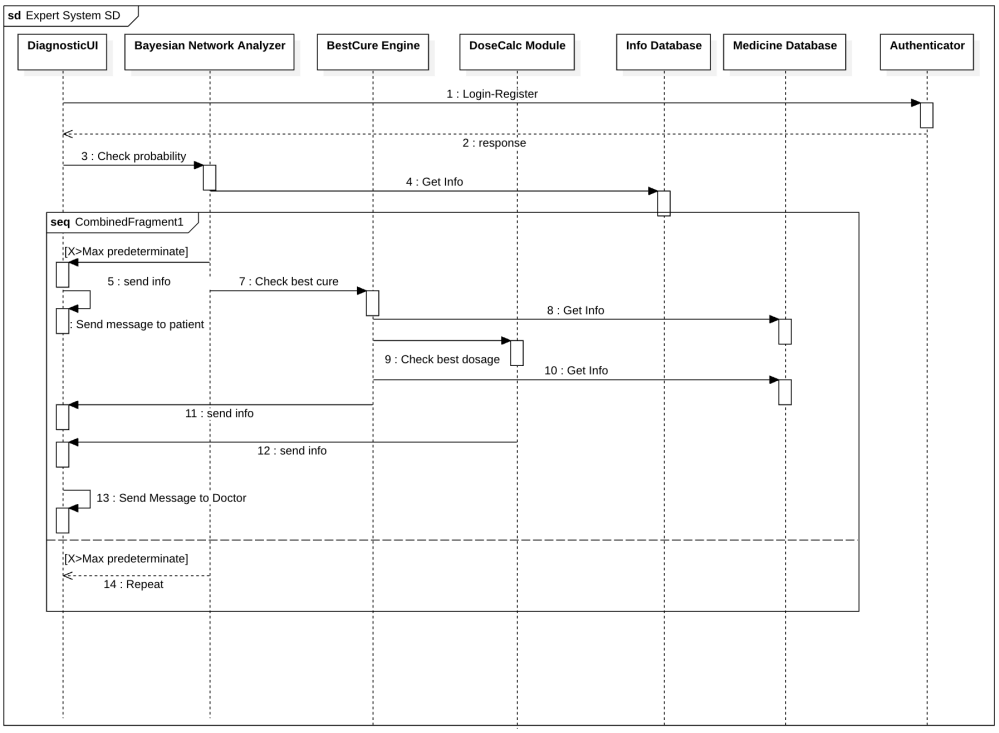


Figure 4.5: Sequence Diagram of the expert system for individualized detection and treatment of COVID-19 patients showing the flow of communication between different components of the system.

4.1.7 Ontologies

OWL language and Protégé software represented and visualized the ontologies discussed above. We started with the CODO ontology [125], which already imported the ontologies FOAF [128], for describing persons and their relationships; Schema.org, for describing location and gender; Snomed CT; OBO [129] for symptoms and clinical findings. To these, the DOID ontology (Human Disease Ontology) [130] was added for rating a comprehensive classification of human diseases organised by etiology [131], manually imported into CODO to retrieve everything from a single .owl file. Other Classes, Object properties and Data properties were then added to the starting ones to have more data to exploit and realise the objective. This task was done following the methodology of our previous work [132]. For example, the Medicines Class was added, in which we found several medicines to be given to patients. These were associated with Object properties such as isAllergic (to define that a patient is allergic to that medicine) and data properties to define whether a medicine is heavy (i.e., whether to suggest it to patients at risk or not). Figure 4.6 shows an example of classes.

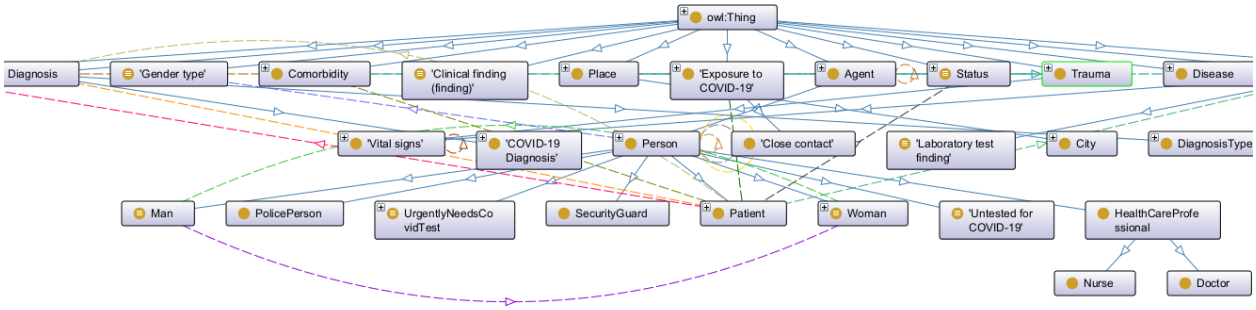


Figure 4.6: OntoGraf to represent classes and their relations in the ontology

4.1.8 OpenData

In this section, we describe how we populated the ontology with data from the website of the Indian Ministry of Health [133] and Family Welfare. Data has already been imported in the ontology from Biswanath Duttaand and Michael DeBellis [134], who used the spreadsheets by Siva Athreya and other researchers at the Indian Statistical Institute [135]. The data were imported by them into Protégé [136] using the Cellfie Protégé Plugin [137] and in their paper [125] is explained how it is structured. Furthermore, we manually imported data from AIFA [138] since there was no structured format to import. We can notice it in the figure above, looking at the instances of the class Medicines. To reach our goal, we had to add more properties to our Patients so we could have more accurate inferences. We can see all the object properties and data properties in the following table, referred to a particular class. If we want to discuss these properties, we can organise it in this way: each Patient has some relationships (some of them can be close ones), has some symptoms, certain drug allergies, comorbidity, a location, some places visited recently. The Patient also has important data properties such as an ID, height, friendly(this let us know whether it is more likely that there was a lot of contact or not), vaccinated (boolean), weight, age. For the class Medicine instead: for each drug, we know if it is not indicated with certain comorbidities, which helps avoid those drugs that the patient cannot take. In this class we also find important data properties such as lowBMI (a boolean value, that tells us if it is for those ones who have a low BMI i.e kids), if it is useful for lungs (since COVID can be very dangerous for them, it is important to know if this boolean value is true so that the system can choose it in case the Patient has some respiratory trauma or any trauma in his lungs), the dosage in Mg and for how many Days, the strength (if true, the system uses it only if the risk is high) and a priority. Then, there is the class Trauma, which contains all the possible comorbidities that a person can have, and an important data property for each one is whether that one is related to a respiratory trauma or not. In the end, the class City, of course, has a Country, but what the system focuses on most is whether it is dangerous (i.e., there

Table 4.1: Object Property and Data Property of the main Classes

Class	Object Property	Data Property
Patient	hasRelationship	Case ID
Patient	hasCloseRelationship	Height
Patient	hasSymptom	Friendly
Patient	isAllergicTo	Vaccinated
Patient	hasComorbidity	Weight
Patient	location	Age
Patient	travelledFrom	NA
Medicine	notIndicatedWith	lowBMI
Medicine	NA	usefulForLungs
Medicine	NA	DosageMg
Medicine	NA	DosageDays
Medicine	NA	Strong
Medicine	NA	Priority
Trauma	NA	RespiratoryProblem
City	hasCountry	isDangerous
City	NA	isInfective

is an outbreak) or whether it is infectious (if some flu is circulating). This last data property is important because the Patient could have symptoms, but they may not be caused by COVID.

4.1.9 Bayesian Network Implementation

In this section, we are going to examine the implementations of the Bayesian Network in Prolog. As we have already discussed in the previous chapter, to find the probability of a Patient having COVID, we used Bayesian rules based on data that we had in our Prolog's facts. As we see in the figure 4.7 , a first example of an implemented Call Graph could be this one(Note that this is not the probability that gives us the final result, but is the one that together with the probability obtained in Figure 7 and the prior probability P(A) actually calculates the final P, as seen in Chapter 4.2 formula (1)). This gives us the conditional probability P(B,C,D|A) discussed in the previous chapter given a Patient. To be calculated, other probabilities need to be calculated, which are calculated by searching for data in the ontology. In more detail, we can say that starting with our highest level rule, we slowly descend in observing other rules that are called upon by those above them. At the second level, in fact (following a top-down approach), we obtain the conditional and unconditional probabilities, which are useful for the final calculation of the conditional probability P(B,C,D|A). In fact, for the calculation of the latter we need (as seen in Chapter 4.2), the conditional probability of having had B symptoms given the certainty of having COVID-19 (P(B|A)), the conditional

probability of having had C close contacts given the certainty of having COVID ((P(C|A)), and the conditional probability of having had D non-close contacts given the certainty of having COVID ((P(D|A)). The prior probability of having COVID P(A) is then also calculated and passed on for the calculation of the final rule that gives us the probability of having COVID P. Each of these rules invokes other, lower rules (as can be seen from the graph), almost all of which are rules that make inferences with the ontology, i.e. they take data from the ontology, allowing the necessary calculations to be made. For example, to obtain the prior probability of Get PA, we need the location where the patient resides to do a whole series of analyses, as explained in the previous chapter. We also see that to obtain P(B|A), we need not only the patient's symptoms but also the prior probabilities P(A) and P(B), which are obtained by recalling the appropriate rules. For P(C|A), on the other hand, we call up the rule to obtain the prior probability P(C), a list of positive close contacts with whom the patient has had close contacts, and again the P(A). Finally, almost the same is done with P(D|A), with the difference that here we call the rule to obtain P(D) and check the list of positive contacts with whom the patient has had contact.

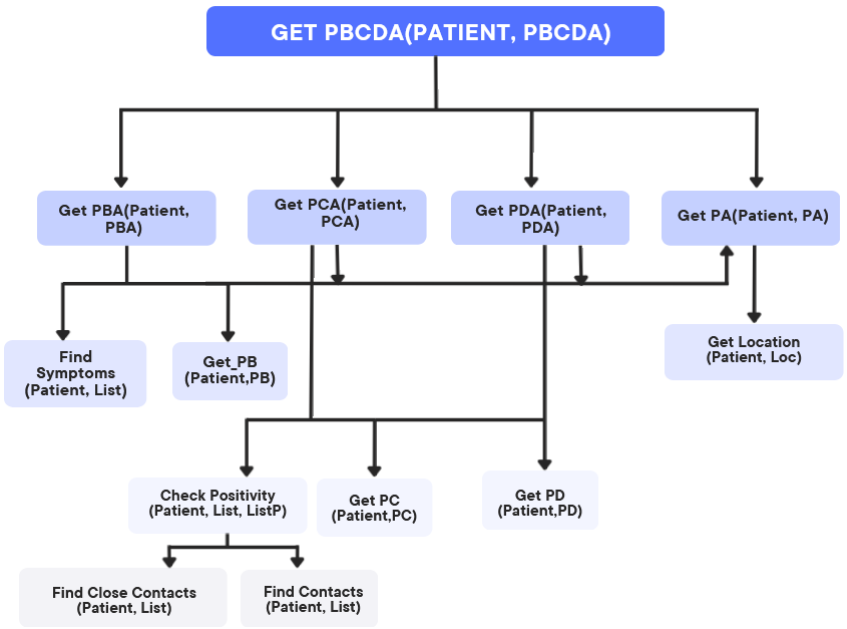


Figure 4.7: Example of Call Graph for the Bayesian Network, which gets the conditional probability of the events B, C, D that given A

Algorithm 1 Get PBA

Patient ← input
List ← findall on hasSymptoms, return a List of symptoms
PA ← GetPA(*Patient*, *PA*)
PB ← GetPB(*Patient*, *PB*)
PAB ← estimate PAB Given ListOfSymptoms
PBA ← calculate **return** *PBA*

In the following algorithm we simply researched all the symptoms of a Patient using the findall of Prolog, based on the Object Properties hasSymptoms. Then, we get P(A) with the specific rule, that will check if the location where the Patient lives is a place where COVID is circulating. We also need P(B), which is linked to the probability of the Patient having B symptoms. This can be estimated based on the location where he lives. For example, if in that place a lot of people are having fever or other symptoms, is more likely that the Patient will have those symptoms. Given the List of symptoms, we can estimate P(A|B), so for example, B=1. Symptoms give us a certain probability of having COVID . In the end, we can calculate P(B|A) with all the data we collected. The algorithm we have just seen and the Call Graph are just a part of the full project, used for giving an idea of how the implementation has been made. Another part of our Call Graph, it is shown in 4.8. Here, we are obtaining the joint probability of the three events: B, C, and D . The calculation of this, call up the individual rules for obtaining B, C, and D, which are taken by inferring with the ontology and taking the data that populate it (such as the patient's location or social activities and thus contacts). As we did before, we can see an example of one of these rules.

Algorithm 2 Get PB

Patient ← input
Loc ← GetLocation(Patient,Loc)
Valuete the Location
PB ← SetPBBasedOnTheLocation
return *PB*

Let us try to understand this simple algorithm. First of all, we give the Patient in input. After that, we found his location with GetLocation. Here we have to analyse the location, it means we have to understand if in that place is circulating influence, because if so, the PB will be higher. We know that from our ontology since we can use our Object Properties to see if a specific place is at high flu risk or not.

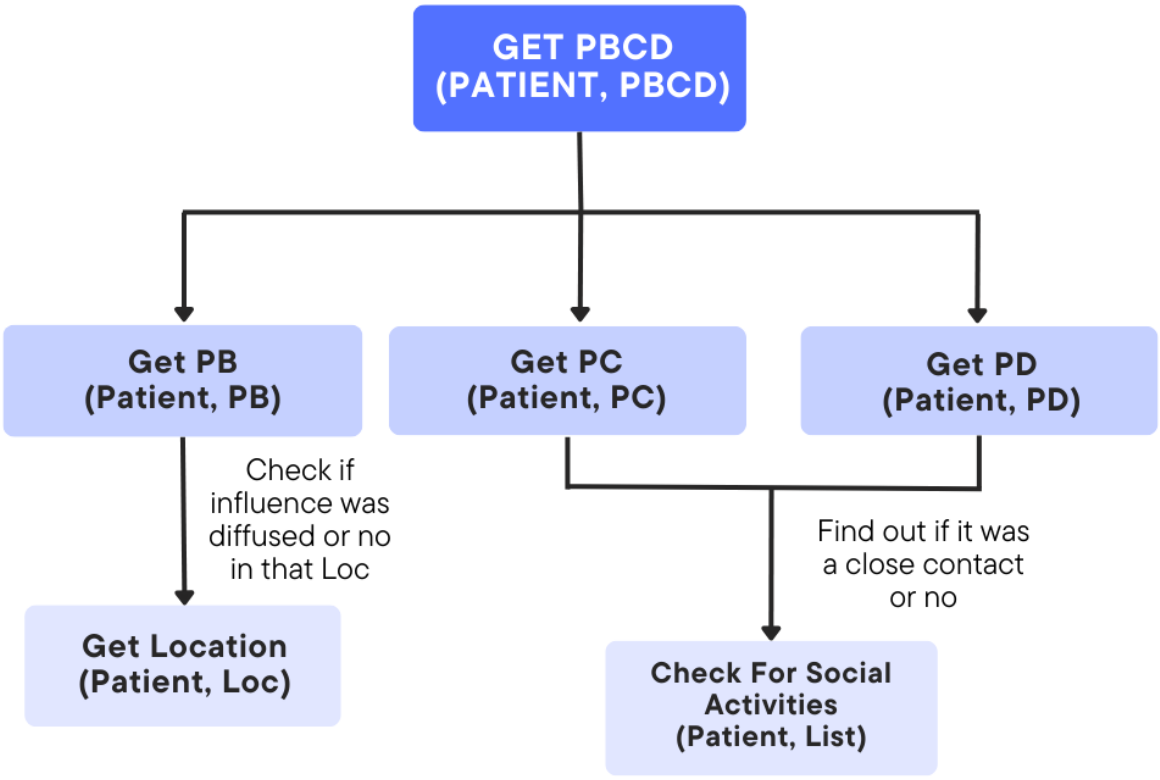


Figure 4.8: Example of Call Graph for the Bayesian Network, which gives us the joint probability of the three events B, C, D

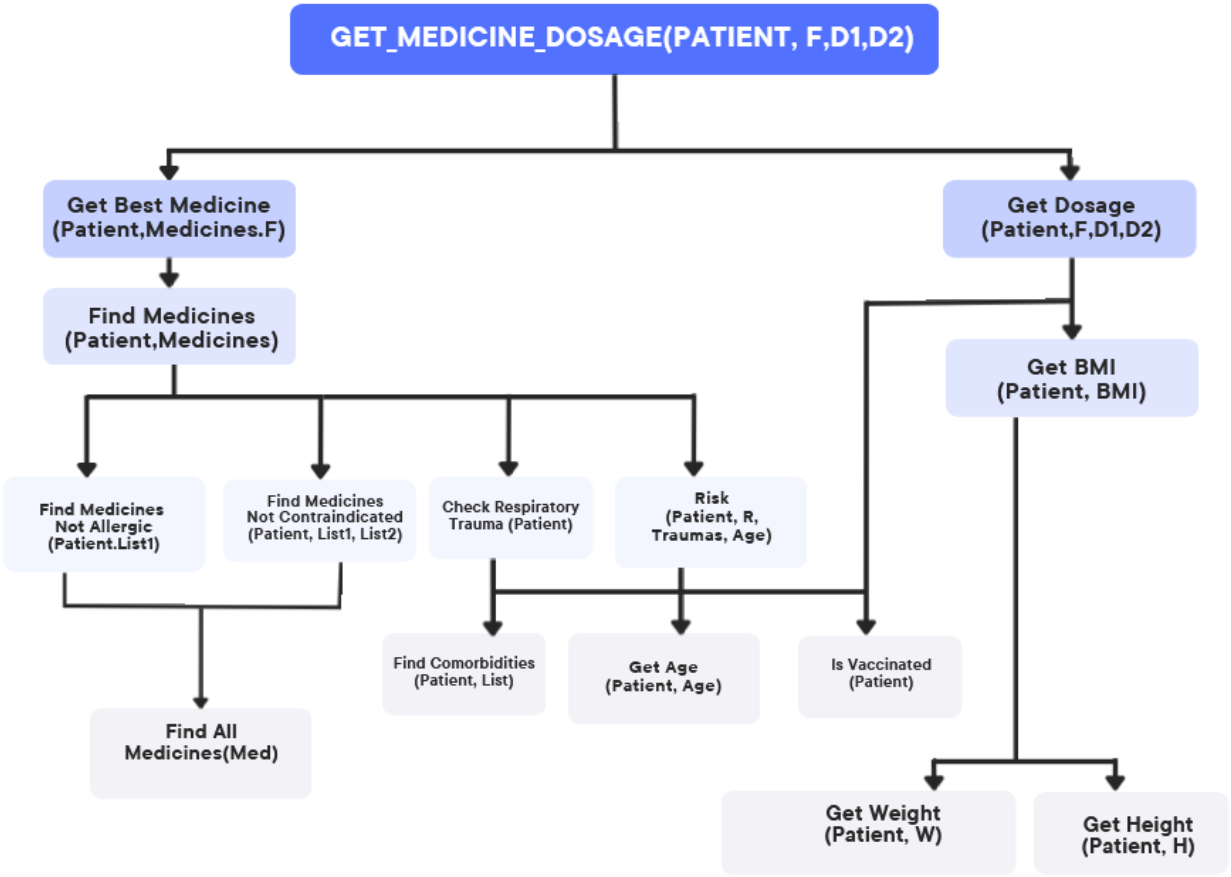


Figure 4.9: Example of Call Graph for Semantic, which gives us the best medicine to take and the dosage for it

4.1.10 Semantic Implementation

We now discuss the Implementation of Semantic part, which is always realised in Prolog. Before discussing algorithms, we first look at a scheme Fig. 4.9, as we did for the Bayesian Network Implementation. So we find the best medicine, F, and its dosages D1 and D2 (dosage in mg and dosage about the days) following the call graph. We see how we first find the best F, choosing among all the Medicines that comply with the requirements mentioned in Chapter 4.3. This choice is taken based on some indicators of priorities (which are Data properties of the Medicines in our ontology), for example, we can know that a medicine X is more efficient than a medicine Y, so we will choose X. After we find F, we can estimate its dosages based on the BMI of the Patient but also on the age. As seen above, following a top-down approach, we see that the lower rules make inferences with the ontology, giving the higher rules data to work with. In order to find the best medicine, in fact, one must first find all the medicines the patient can take (Find Medicines at level 3 of the graph) and then choose the most efficient one (a choice that the patient's doctor

may also condition). The search for these medicines, therefore, invokes lower rules that bring back data lists from the ontology, such as all those medicines that the Patient cannot take because they are contraindicated or because of some allergies. Now let us try to see two examples of

Algorithm 3 Get Medicine Dosage

1: $Patient \leftarrow$ input;
2: $F \leftarrow$ GetBestMedicine$(Patient)$;
3: $D1, D2 \leftarrow$ GetDosage$(Patient, F, D1, D2)$;
return $F, D1, D2$;

algorithms.Algorithm 3 finds the best medicine, F, and of this, it finds the dosage in mg D1 and the number of days the patient has to take it, D2. Now we are trying to see a more complicated algorithm and understand how we applied our semantic logic. The following algorithm is only

Algorithm 4 Find Medicines

$Patient \leftarrow$ input
$List1 \leftarrow$ FindMedsNotAllergic$(Patient, List1)$
$List2 \leftarrow$ FindMedsNotContraindicated$(Patient, L1, L2)$
$R \leftarrow$ Risk(Patient,R,Trauma,Age)

$ifRiskishigh \rightarrow CheckRespiratoryTrauma$
$ifThereIsTrauma \rightarrow FindStrongMedicinesForIt$
$ifNoTrauma \rightarrow FindStrMedForItNotLinkedWithResp$

$ifRiskisNothigh \rightarrow FindForMedicinesLessStrong$
$ifIsAKid \rightarrow FindForMedicinesForKids$

return $Medicines$

a part of the complete code made in Prolog. It is focused on showing how we tried to find the best medicines based on the Patient, checking all the possible combinations. So, once we have the Patient, we first find all the medicines he can take, because he is not allergic $\rightarrow List1$. Then, from our List1, we discard other medicines that he cannot take because they are contraindicated, so we create List2 where there are all the medicines he can take (not allergic and not contraindicated). We also check the Risk of the Patient(i.e based on his age, his trauma), so we decide whether to choose a heavy medicine or not, so that we can also optimise a possible distribution. A check is also made if the patient has respiratory problems, as in this case, it would be more effective to suggest a suitable drug. Finally, there is also a check on the patient's age, because more suitable medicines will be suggested if a child.

4.1.11 Results and Discussions

In this chapter, we will explain how the project was tested at the end of the work in order to verify its functioning. As shown above, before writing code in Prolog, we schematised everything through call graphs (it should be emphasised that only a part of the call graph has been reported here). We therefore started by testing the leaf rules of the graph, and then went up as we went along, thus using a bottom-up approach.To test the system's functioning, before looking at what the rules in Prolog are called, it is good to talk about the test set used. As mentioned in the previous chapters, the ontology was populated with data taken from various sources. Then later these data were enriched and modified to suit the objective of our problem. Thus, real data were used mainly taken from the Indian Ministry of Health [135] and modified to suit our purpose (e.g. by adding additional relations or properties). Given the vastness of the rules used, we will only report a few here to summarise the whole, which are significant and can show how we then went back up to test more complex rules from the leaf rules. Let us start, for example, by analysing the operation of the get_best_medicine rule, via the SWI-Prolog call stack.

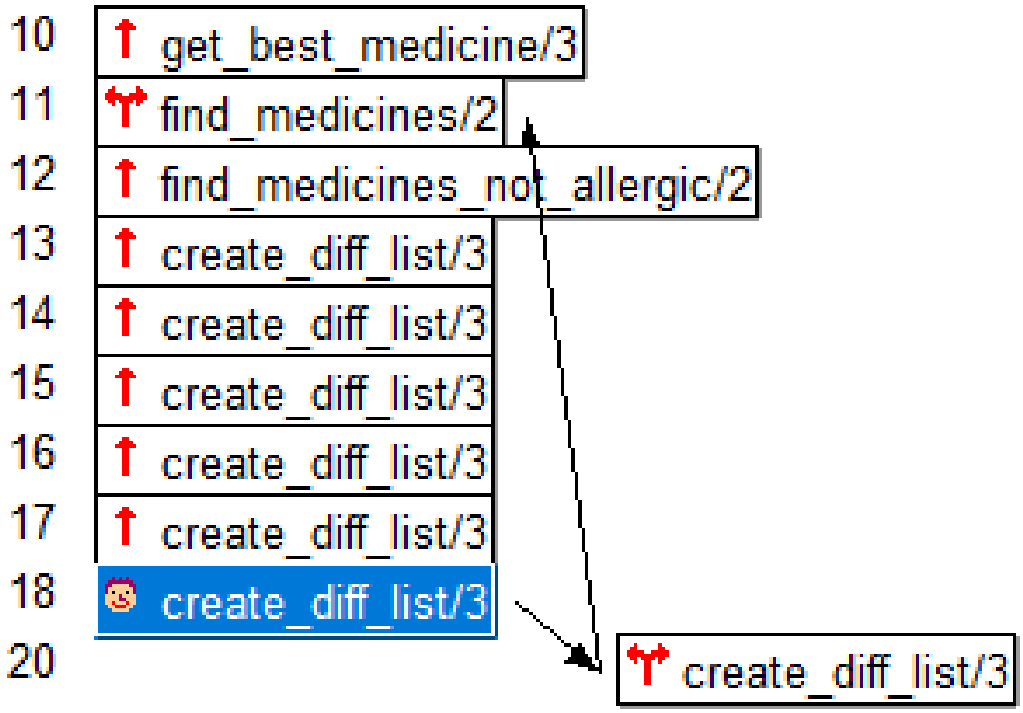


Figure 4.10: Call Stack SWI-Prolog for getting the best medicine after searching the medicines available and the ones the patient is not allergic to, making a final list thanks to create diff list which contains the medicines he can take

In Fig. 4.10, we can see the calls made to find all the medicines to which the patient is not al-

lergic. Two lists are created, those of the medicines and the medicines he is allergic to, and through create_diff_list a list is created with only the ones he is not allergic to. As we see, create_diff_list is called several times because it gradually removes from the list of medicines that the Patient cannot take because he or she is allergic. Let us now move on and see what other rules are called. In 4.11 and 4.12 , we see that the same thing has been done for the medicines not contraindicated for the Patient, with the difference that another 'find_medicines_to_discard' rule is used to discard from the list of drugs to which the patient is not allergic, even those that the patient cannot take because of previous illnesses (comorbidity). First, the rules for obtaining all the medicines that he cannot take due to contraindications, e.g. due to previous illnesses or other diseases, are recalled

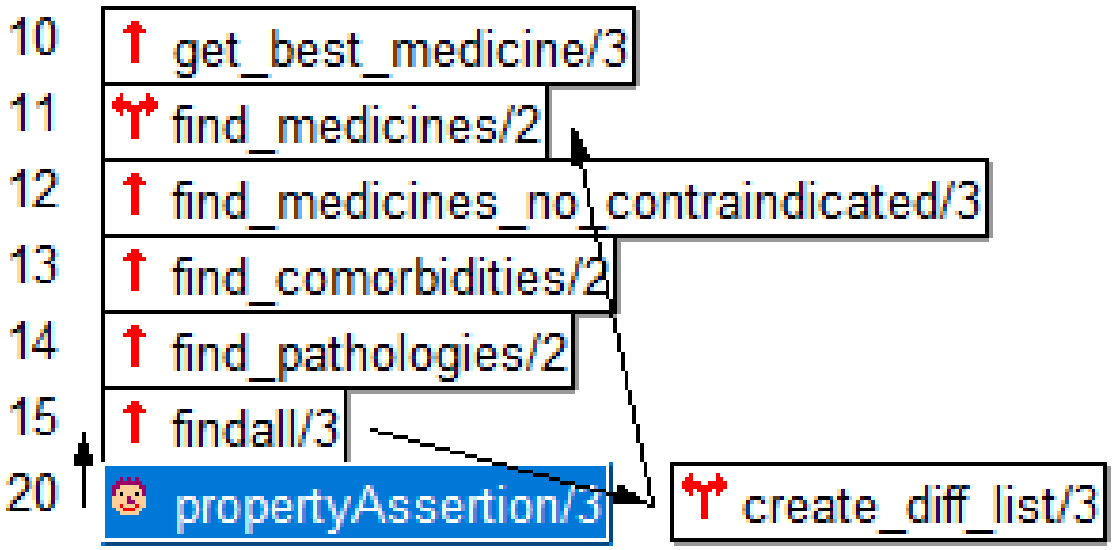


Figure 4.11: Call Stack SWI-Prolog for finding the medicines not contraindicated for the patient, so not contraindicated for comorbidity or a pathology the patient has I

Next, one tries to eliminate from the list of possible medicines those that are on the list of contraindicated medicines, thus obtaining only the medicines that are not contraindicated (with at least one element in common I check if I have someone on the list of suggested medicines that is not indicated, so that I can eliminate it with find medicines to discard). So, to summarise, in finding medicines, I will only have the medicines to which the patient is not allergic and to which he has no contraindications

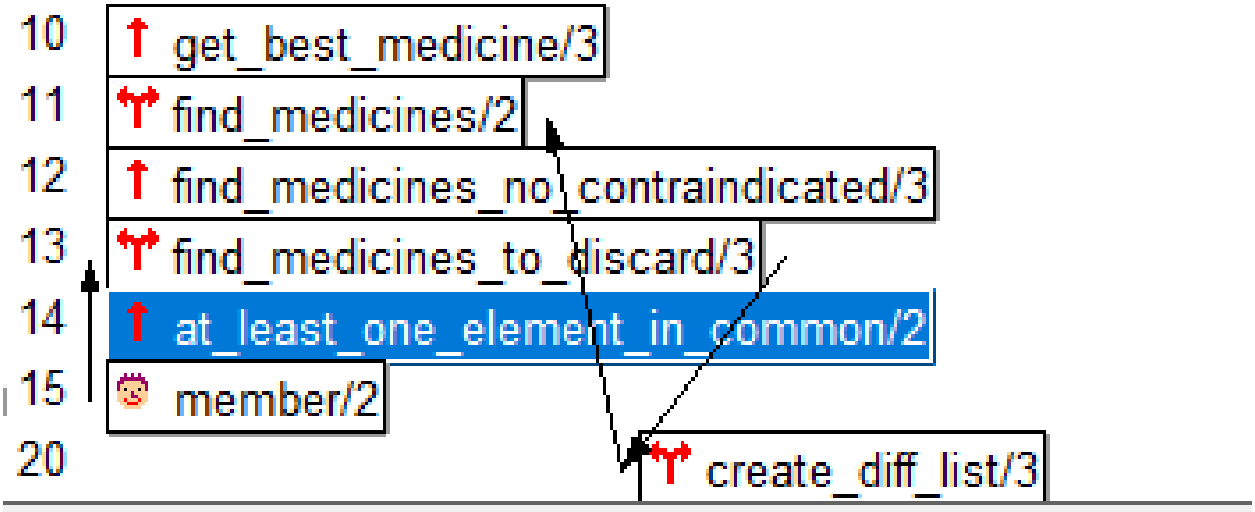


Figure 4.12: Call Stack SWI-Prolog for finding the medicines not contraindicated for the patient, discarding all the medicines he cannot take because of a comorbidity II

Now we have to analyse the patient's risk to determine what kind of medication to give him (i.e., whether heavy or not). We start by checking the patient's trauma 4.13; the more trauma he has, the more he will be at risk. Let us, therefore, make a trauma count according to the prior illnesses that the patient has.

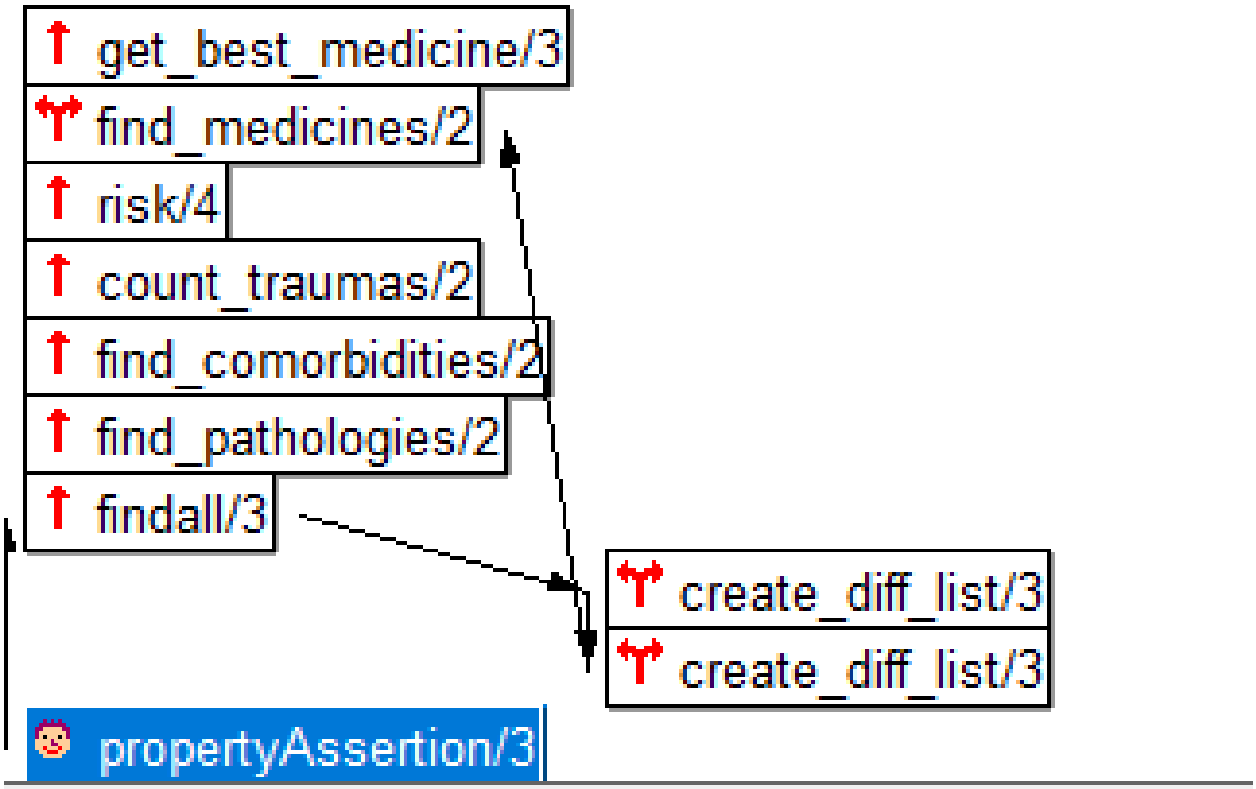


Figure 4.13: Call Stack SWI-Prolog which analyses the patient's risk, so if he needs a strong medicine. We see the risk analysing the trauma or all the comorbidity he has

Then we check the age 4.14, an older patient will be more at risk than a young one. Furthermore, from the age we can also see if we are dealing with a child and therefore give him drugs suitable for a child.

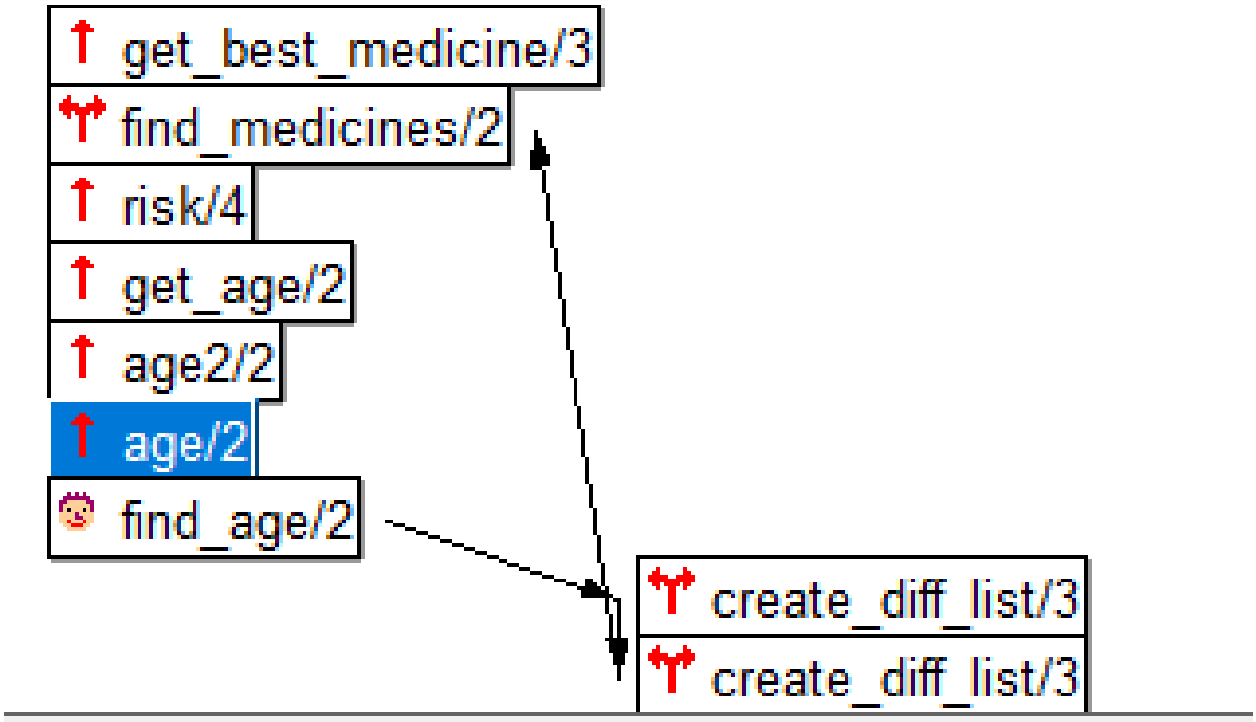


Figure 4.14: Call Stack SWI-Prolog for analysing the risk of the Patient, checking his age

We also check if the Patient is Vaccinated 4.15. Having the vaccine doses recommends a lower probability of risk in case you have COVID.

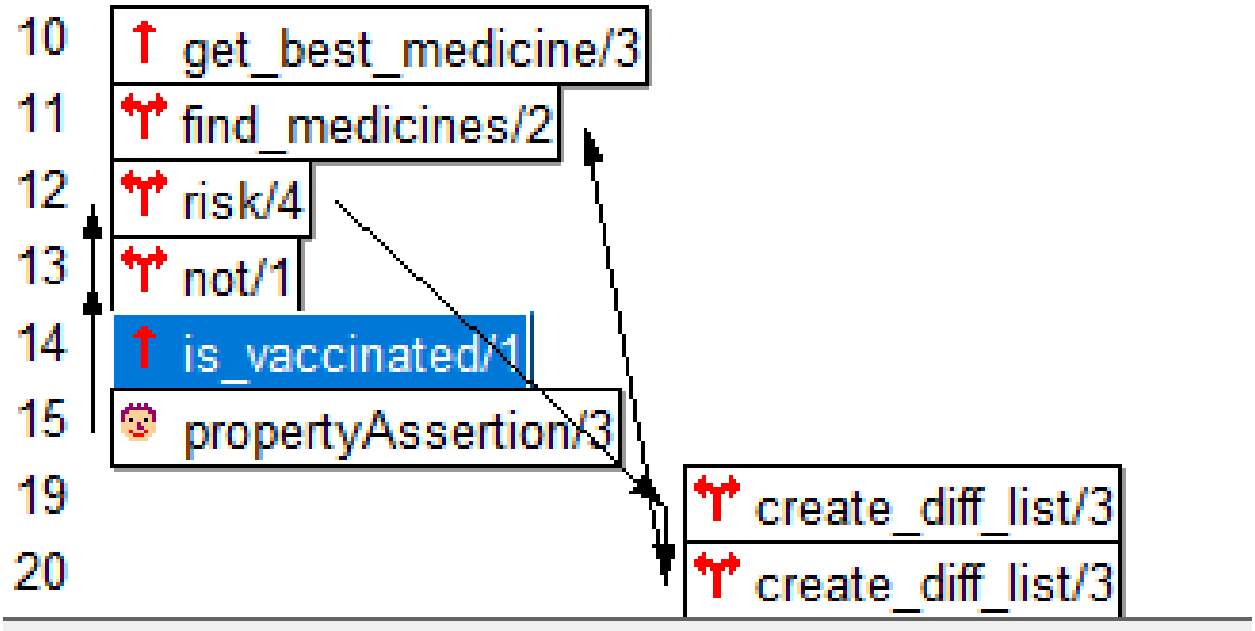


Figure 4.15: Call Stack SWI-Prolog for analysing the risk of the Patient, checking if he is vaccinated or not

At the end, we check if the Patient has some kind of respiratory trauma 4.16. We do this by checking for references to respiratory trauma among previous illnesses

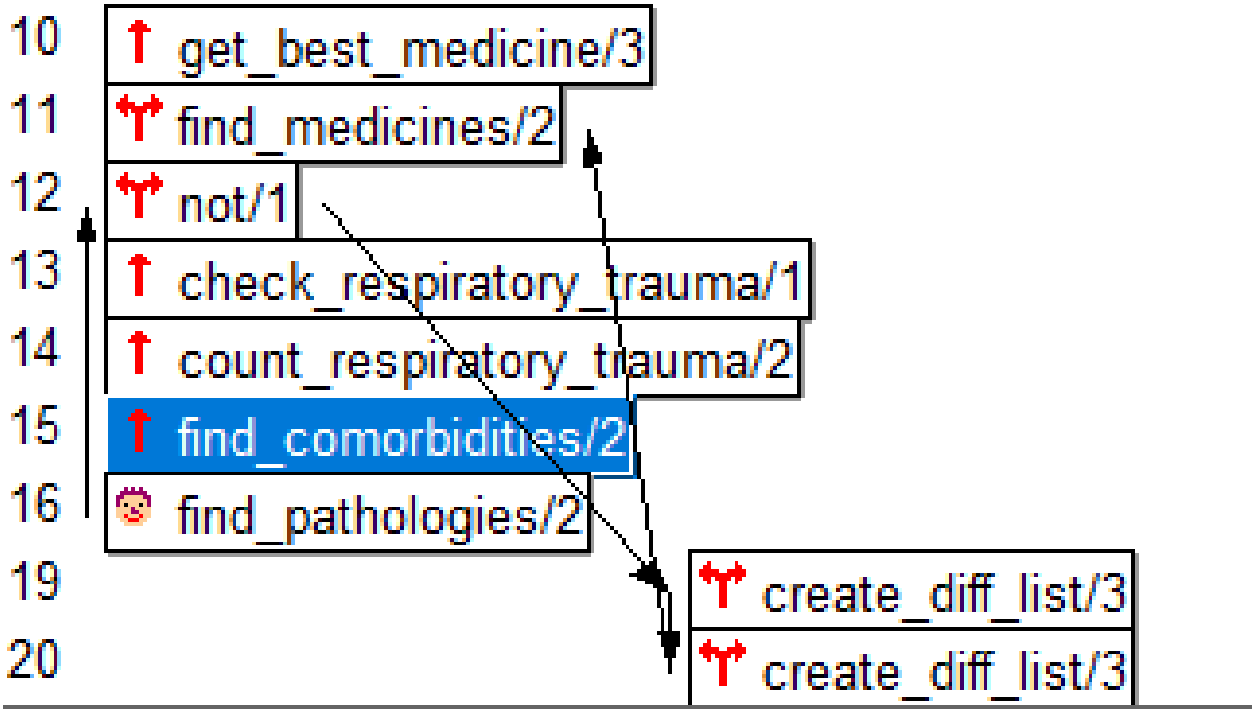


Figure 4.16: Call Stack SWI-Prolog for analysing the risk of the Patient, checking if he has some respiratory traumas or not

Now that we have all this information, we can get our best medicine from the available ones. For this, we just need to get the one with the max priority, so the one is more efficient 4.17. It should be emphasised that the choice of the best medicine is something the referring doctor will also decide on, since he knows the patient best

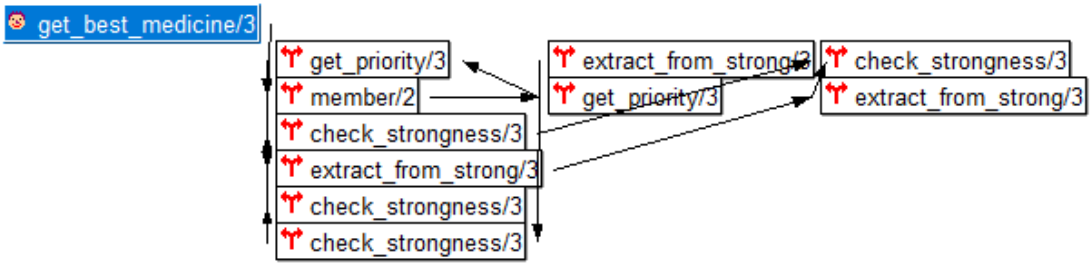


Figure 4.17: Call Stack SWI-Prolog for getting the medicine with the max priority from all the ones the Patient can get

This was an example of what the rules were called for, a well-articulated rule like the one we have just seen. The same can be done with the other rules; this is how we tested them as we went along. We would like to point out that the tests were carried out on a well-known sample of data for which we had all the necessary information, as described in the previous chapters. There is

nothing to prevent us from adding more patients and trying to test the system created. In the next chapter, we will give a hint as to how the following project can be further elaborated.

4.1.12 Conclusion

In this work, an expert system for the diagnosis and personalized treatment of SARS-CoV-2 has been projected, developed and validated. Probabilistic models guided by semantic techniques have been integrated within it. The final validation of the system, detailed in the previous chapter, demonstrates its effectiveness as a tool in the fight against COVID-19 and also a solid base against any type of highly contagious disease. Among the future developments, we can project to expand the database of ontologies and open data to include more different health information. On the other hand, using machine learning algorithms [92] could further improve the probabilistic assessments and treatment recommendations. This could make the system adaptable to new data and emerging medical knowledge. A crucial point to consider is the rapid change of the virus [139] [140]. This can lead to changes in symptoms, the speed of transmission, the danger and mortality of the virus and other relevant factors in general. For this reason, it would be fascinating to develop an expert system that considers this dynamic part, thus ensuring a more effective and timely response to emerging challenges. A preliminary work, published at the IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC) conference, was done to handle this type of casuistry [132]. In particular, in this work, an analysis of different ontologies was carried out; it was followed by an integration of classes, individuals, and relations, which can be used for the abovementioned purposes. For ease of reading, we will rewrite internally within the appendix, specifically in the section A. This expert system is a step forward in the use of artificial intelligence in healthcare to improve patient outcomes and the management of public health crises.

4.2 CR8

In the digital age, interactive storytelling has gained popularity and is used not only for entertainment but also for marketing, training, and education. However, creating engaging and complex stories can be a challenge that requires a wide range of artistic and technical skills. This paper presents a new methodology and prototype tool for graphically composing stories based on an ontological model in which semantics shapes all structural elements of the narrative. This tool not only facilitates the graphical creation of narratives, but also enables the creation of visual queries to examine the story and highlight the relationships and connections between the elements of the narrative structure, such as scenes, events, and actions, as well as the agents involved, such as actors and props. The main advantage of this tool is its ability to simplify the process of creat-

ing and analyzing stories, making advanced storytelling techniques accessible even to those who are not technical experts. Users can visualize and manipulate narrative components through an intuitive graphical interface, interactively exploring the interactions and hidden meanings between various elements. This visual and semantic approach enables better understanding and management of complex stories, facilitating collaboration among teams of creators and improving the effectiveness of narrative communication. This tool has multiple possible applications: it can be used in writing screenplays for films, creating interactive educational content, designing engaging advertising campaigns, and in narratology and social science research. The ability to graphically interrogate the structure of a story and explore the relationships among its elements offers new perspectives and tools for analyzing and understanding narratives, paving the way for changes in the way stories are conceived, developed, and shared.

4.2.1 The Proposed Methodology

The proposed methodology underlying the tool is structured as a pipeline of five key stages, each of which contributes to the creation and enjoyment of storytelling. Figure 4.18 shows the pipeline.

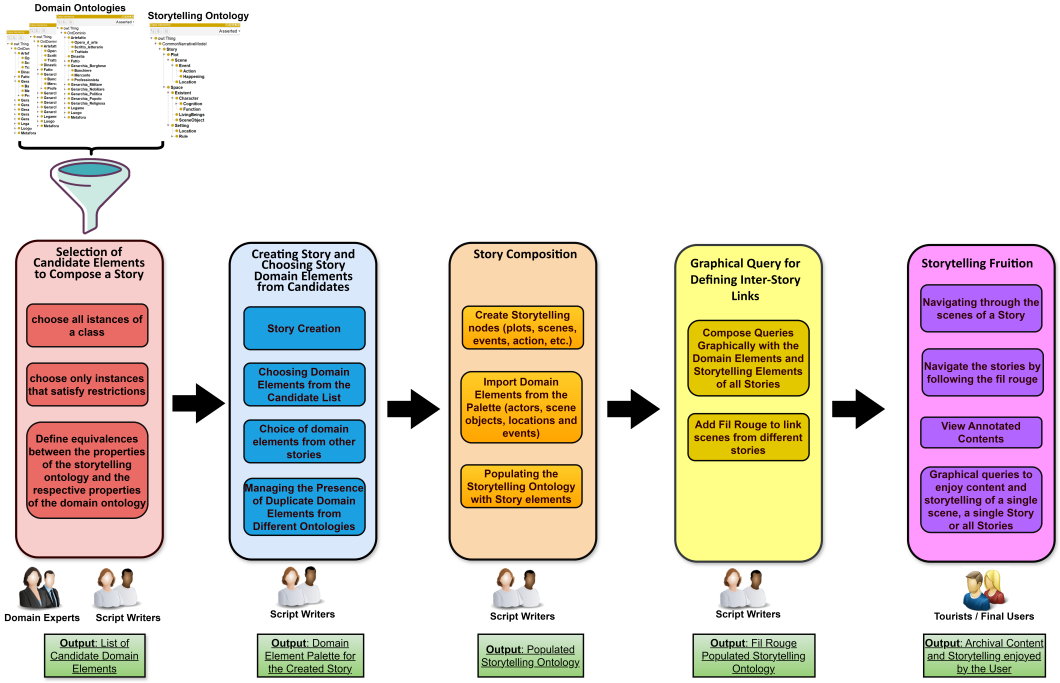


Figure 4.18: Methodology Pipeline.

The phases illustrated in the methodology in Figure 4.18 are explained below:

1. **Selection of candidate elements to compose a story**: is the phase in which the elements (actors, scene objects and locations) to be used for story creation are chosen from one or

more domain ontologies. The selection of candidate elements is the main operation, which uses a storytelling ontology and one or more domain ontologies as input and produces a list of candidate domain elements as output.

2. **Creation of the story and choice of domain elements among the candidate elements**: is the phase in which the user creates a new story, assigning it a name and selecting the domain elements to be used from the candidates chosen in the previous step and the elements present in other stories. The output of this phase consists of a domain element palette for the created story.

3. **Story composition**: is the phase in which the user uses the elements chosen in the previous phase to construct a coherent and detailed narrative. The story is structured by organising the actors, events, scene objects and locations selected within scenes and actions that make up the plot. The output of this phase consists of the storytelling ontology populated with all story elements, both structural (plots, scenes, events and actions) and domain elements (actors, locations and scene objects).

4. **Graphical queries for defining inter-story links**: is the phase that allows domain experts to create and manage links between different stories using a graphical tool. This activity allows composing queries to visualize specific portions of stories, navigating through annotated content, and creating ”inter-story links” also named ”*fil rouge*”, that connect elements of different stories. This links can be saved in a new ontology which can constitute the knowledge base for a complex storytelling system.

5. **Storytelling fruition**: is the phase that allows the end user to explore the story starting from its specific components, such as events, places or characters, allowing to navigate through the scenes of the story, follow the various ”fil rouge” and view the annotated contents, such as references to archival sources. The user can choose where to start the narration, making the experience highly personalized and interactive, adaptable to their interests.

In the following Subsections, the various phases of the proposed methodology are explored in more detail, accompanying the description of each phase with a use-case diagram.

4.2.2 Selection of Candidate Elements to Compose a Story

Figure 4.19 reports the UML use cases related to this phase.

The system must be pre-loaded with the storytelling ontology and the set of domain ontologies to be used. The domain expert can create a new domain ontology using an ontology editing tool, or can choose to use an existing domain ontology. All domain ontologies used will be automatically

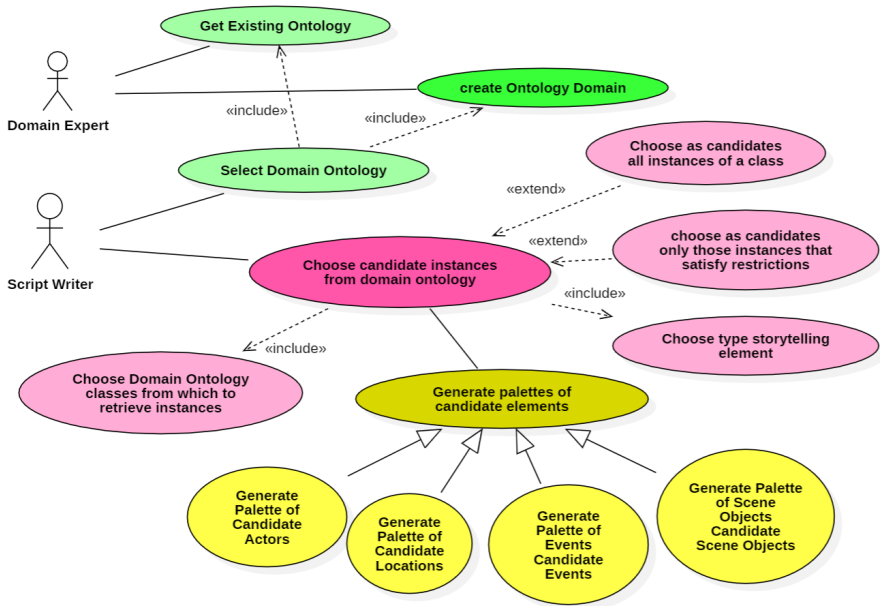


Figure 4.19: Use Case of Phase ”Selection of Candidate Elements to Compose a Story”.

imported into the final populated storytelling ontology produced in the following steps. After that, the script writer can choose a class from the selected domain ontology, from which he can select the desired domain instances. The script writer has two possibilities:

- It can select all individuals of the selected class as domain instances;
- It can select as domain instances only the individuals of the selected class;
- It can select as domain instances only those individuals of the selected class that satisfy defined constraints.

With the last option, the script writer has the possibility of applying as many constraints as he wishes by selecting certain object properties defined in the domain ontology that have the selected class as their domain, and then choosing one or more desired values for that object (once the object property is selected, all the possible values that this property can assume in the ontology are displayed). For example, assuming an ontology modelling all sovereigns in history had been chosen, the script writer could select the class ”*Kings*” and then restrict the selection of kings by applying the object properties ”*reigned in*” to the value ”*Kingdom of Naples*” and ”*has dynasty*” to the value ”*Asburgo*”. In this way, all and only the kings of the dynasty ”*Asburgo*” who reigned in the ”*Kingdom of Naples*” are filtered out of all the individuals of the class ”*Kings*”. All domain instances selected from a class become candidate elements of a storytelling element type (actors, scene objects, events, and locations). When the script writer finishes selecting domain instances from the classes of one or more domain ontologies, a palette of candidate elements for each storytelling

element type is generated. For each candidate domain instance chosen, the following information is stored: i) IRI of the relevant domain ontology; ii) class of the relevant domain ontology; iii) corresponding storytelling element.

4.2.3 Creation of the Story and Choice of Domain Elements among the Candidate Elements

Figure 4.20 shows the UML use case related to this phase.

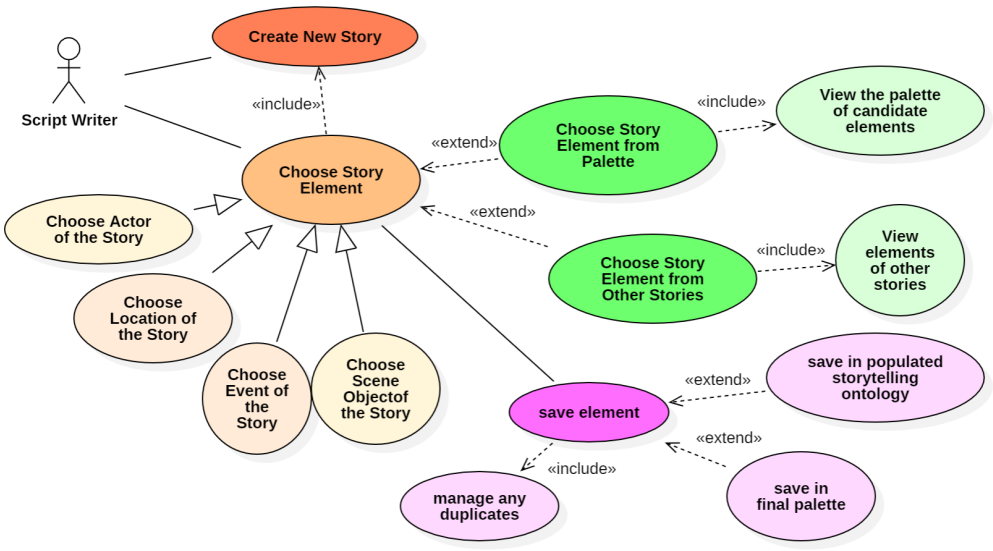


Figure 4.20: Use Case of Phase ”Identifying the elements of a story from the candidates”.

In this phase, the script writer creates a new story, assigning it a name, and selecting the domain elements to use among the candidates chosen in the previous phase and the elements present in other stories that he/she has modelled with the tool in different work sessions. At the moment the story is created, an individual of the class ”*Story*” is added, renamed with the name assigned by the script writer. The elements chosen at this stage are automatically inserted into the **Populated Storytelling Ontology** as individuals of the classes ”*LivingBeings*”, ”*Location*”, ”*SceneObject*” and ”*Event*”. In addition, all the individuals chosen by the script writer are automatically included in the graphic palette of actors, scene objects, events, and locations that he or she will display in the next step during storytelling modeling. A duplicate handling mechanism is also provided at this stage: Since the script writer has the opportunity to work with multiple domain ontologies, the same domain instance could be defined in different ontologies. For example, the domain instance ”*Julius Caesar*” could be modeled in the ontology of the sovereigns of history, but also in an ontology that models the characters of the Roman empire. In that case, the script writer has the possibility of selecting both and specifying that the two candidate instances are equivalent. This

information is also stored in the populated storytelling ontology: a new individual is created that is set ”equivalent” to the other two individuals through an equivalence property, such as ”*same Individual As*” in the Ontology Web Language (OWL). This phase ensures an accurate and organized selection of domain elements, laying a solid foundation for the story composition in the next phase.

4.2.4 Story Composition

Figure 4.21 shows the UML use case related to this phase.

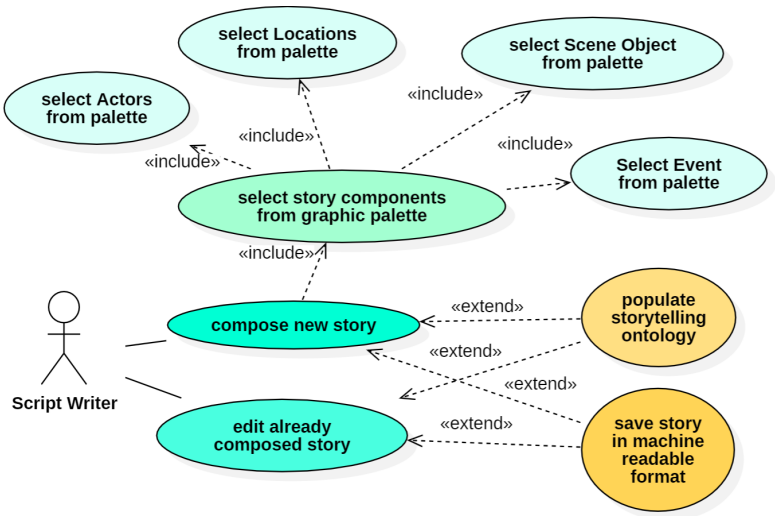


Figure 4.21: Use Case of the Phase ”Story Composition”.

In this phase, a comprehensive and organized story can be created using the domain elements selected in the previous phase. Using a special ”Story Board”, the script writer can define the structure of a story by defining the nodes ”scene”, ”event”, ”plot”, and ”action”. Then, to these nodes you can associate the domain elements; for example, a scene node can be associated with the corresponding location, actors involved, scene objects used and the event that influences the scene. The domain elements can be chosen from a special graphic palette built from the story elements selected in the previous step. One particular case is events, which can be either selected from the palette as domain instances or defined as a new node in the Story Board. The script writer can create multiple scenes and organize them in a logical and temporal order of events in the story. Once the story composition is complete, the script writer can save the work populating the storytelling ontology with all elements about the story. In particular, Individuals of the classes ”*Plot*”, ”*Scene*”, ”*Event*” and ”*Action*” are created, and by object properties the narrative flow between these elements is modeled. Finally, all domain instances used are linked to the structural elements of the story by means of special object properties. For example, all actors involved in a scene are linked to the specific individual of the ”*Scene*” class by the object property ”*hasScene*”.

It's important to highlight that the domain elements will belong to domain ontologies, while the structural elements inserted through the Story Composer become individuals of the ontology of storytelling: this approach ensures greater flexibility, as it's possible to modify the information of a domain instance without changing anything in the history, since the ontologies in which these elements are saved are different. The story is structured and coherent, ready to be presented and consumed by the end user thanks to the use of ontologies and relationships defined between the elements.

4.2.5 Graphical queries for Defining Inter-Story Links

Figure 4.22 shows the UML use case related to this phase.

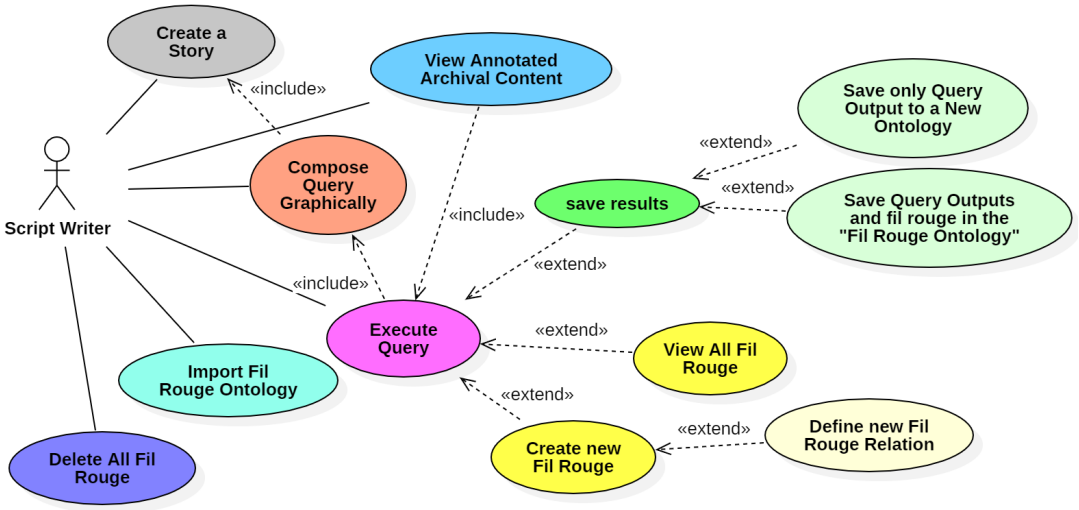


Figure 4.22: Use Case of the phase "Querying of Stories".

The script writer, after creating one or more stories, can compose queries using a graphical tool to interrogate the system, asking it to display only portions of stories of interest. From the query result, it's also possible to "navigate" the stories by viewing, one node at a time, semantically annotated archival contents associated with that node. Once the query is executed, the script writer can create some fil rouge to connect elements belonging to different stories through a "fil rouge relations," which can be acquired from a predefined set defined by other domain experts, or a new one can be defined. An example of a "fil rouge relationship" could be "*is_Same_Scene*", which defines that there is a strong similarity between two scenes belonging to different stories. This situation is very common, as the same "fact" is often narrated in different movies/books. Performing a query is the fastest way to identify these situations. For example, the script writer could ask the system to extract all and only scenes taken in a specific time range, in a specific location and involving certain actors; two scenes of different stories that satisfy these constraints,

have a strong chance to tell the same thing in different stories. All the fil rouge relationships entered by the script writer are saved in "**Fil Rouge Ontology**"; the latter one imports the storytelling ontology and defines the inter-story between the different storytelling elements of different stories, which in the ontology are modelled as object properties. This ontology constitutes the input of the next phase of fruition, because it can allow the end user to enjoy the contents of a story and switch to another story whenever an inter-link is identified. It also allows for the modelling of interwoven stories.

4.2.6 Storytelling Fruition

Thanks to the ontology of the storytelling populated and the Fil Rouge ontology defined in the previous phases, it is possible to build a knowledge base of a user system that will allow any user to enjoy the content of the stories modelled with the tool. The completion phase allows the end user to explore the story starting with its specific components, such as events, locations, or characters. The final user can choose where to start the narration, making the experience highly personalized and interactive. This mode allows the user to flow freely through the story, offering an engaging experience that can be tailored to their interests. It is possible to define many scenarios of fruition.

4.2.7 Tool development

This Section describes the architecture of the tool that implements the methodology proposed in Section 4.2.1. The architecture of the instrument is designed in modules, each of which performs a unique function. This choice was dictated in order to have a clear organization and, consequently, efficient maintenance of the system. In this Section we will therefore analyze in detail the various modules of the system, detailing the functions for each of them and how they interact with each other. The architecture of the system is shown in Figure 4.23.

The architecture shown in Figure 4.23 is macroscopically divided into 3 layers: the first layer contains all the components relating to the selection of candidate elements (**Candidate Elements Selection**), the filtering of individual elements (**Filtering Elements**) for individual stories and all the components relating to creation (**Story Creation**), fruition of the story (**Story Fruitions**) and all the management of constraints to carry out individual atomic operations (**Constraint Checker**). This first layer is deployed on a local component, specifically any personal computer. The second layer instead contains all the information and data relating to the use of digital contents (**Visualization Digital Elements**). Since these contents are not public, this part of the components will be deployed on a private server. Finally, the third layer is relative to all the storage components of the content produced including element candidate files, stories, ontological relationships (**Storage File**, **Database info Stories and Users**), access credentials (**Authentication**),

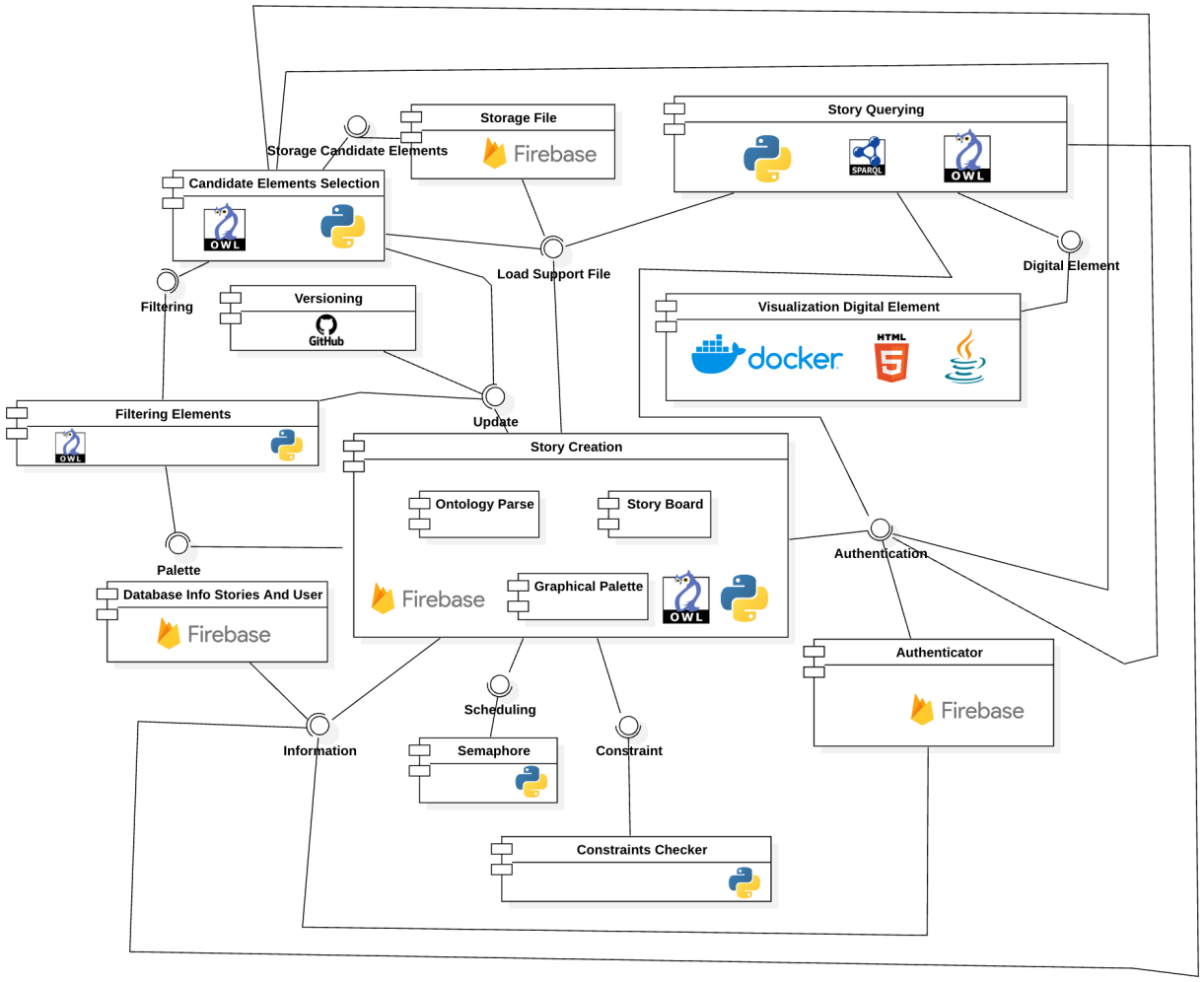


Figure 4.23: Component diagram of the Entire Architecture of the Story Composer Tool.

access management via semaphore mechanism (**Semaphore**) and versioning mechanism to release new software updates (**Versioning**). These components are deployed in a public cloud server. In the following, all the individual components of the system are described and then, also through the use of sequence diagrams, it is specified how communication takes place.

- **Authentication**: the first component deals with the authentication mechanism necessary to guarantee correct access within a public cloud shared by multiple users. Recording as shown in the component diagram in Figure 4.23 is not permitted for security reasons. To register a user by design choice, an administrator must be contacted. This component was implemented in **Python** and uses **Firebase** for user management.

- **Versioning**: the versioning mechanism is necessary to be able to send software updates relating to local components. Updates by design choice are not mandatory. In fact, there is a function within the component that requires confirmation from the individual client before sending the update. The reason why updates are made optional is to give individual users the opportunity to choose when to download and to also be able to work with an older version. This component was implemented in **GitHub**.

- **Semaphore**: as a multi-user tool it was necessary to introduce a semaphore mechanism to avoid overwriting within the individual stories or the file relating to the candidate elements. From this perspective, therefore, this mechanism, through the use of triggers based on information saved within a public database, makes it possible to avoid having multiple users work on the same files. This component was implemented in **Python** and uses **Firebase**.

- **Storage File**: the component in question allow saving any type of operation carried out through the tool, from creation to modification to deletion of individual elements within files or the entire file. It also contains the global ontology that has all the relationships between the elements of the individual stories and between all the stories. Precisely for this reason it contains the necessary information for the ”fil rouge” through which it is possible to view the connections between multiple stories. Finally, it contains all the stories using files that map the components of the ontology in JSON format This component was implemented using **Firebase**.

- **Database info Stories and Users**: in this component, all the information relating to users and files produced by the tool is saved, therefore unique identifiers for the pointers to the files within the storage, useful information for the semaphore mechanism for being able to manage inputs and references to names to be used as information for the user (To name one: there is an implementation part within the tool that allows all users to view who is working on a specific story). This component uses **Firebase**.

- **Visualization Digital Elements**: this component positioned within a private server has the task of providing access and visualization during use to digital contents that can be viewed through web interfaces. The visualization occurs through unique pointers that are inserted during the creation of the story. This component was realized in **HTML** and **JavaScript** and subsequently containerized and deployed thanks to the use of **Docker**.

- **Story Creation**: the component in question represents the core of the entire work as this allows the creation and modification of individual stories through the help of all the other components. From here it is possible to request the database to upload or download stories, modify them through the use of 3 elements: i) a **Story Board** to compose a story by inserting structural components (plots, scenes, events and actions) and link them together; ii) an **Ontology Parser** to populate the storytelling ontology with all elements of the story modelled with the Story Board; iii) a **Graphical Palette** filled with domain instances (actors, locations and scene objects) that can be associated with the structural components of a story via the Story Board. The properties of domain instances chosen from the palette are already set automatically thanks to the information retrieved from the domain ontology, whereas the properties of structural elements (e.g. scene description) that can be entered from the Story Board must be set manually by the user. This component implements all functionalities defined in Section 4.2.4. This component was realized in **Python** by extending an already existing module available online[1]. The **OWL** language was chosen to model the domain ontologies, and Python's **OWLReady2**[2], **SPARQLWrapper**[3] and **RDFLib**[4] libraries were used to retrieve information from the ontologies.

- **Constraint Checker**: since the connections of individual objects on the Story Board follow the ontological storytelling model, this component was added with the purpose of preventing the user from entering unauthorized connections or unauthorized nodes by performing real-time checks. The component also inserts a visualisation with different colours for the various types of relations and nodes, ensuring greater user-friendly experience. In addition, this component checks whether unauthorized elements have been inserted when the element is created or modified. If these problems occur, it formats the values correctly; if it fails to do so, it issues an error message (e.g. incorrect formatting of the date). This component was implemented in **Python**.

- **Candidate Elements Selection**: by design choice, as already mentioned before, only actors,

[1] https://github.com/tristanpoland/SkiffUI?tab=readme-ov-file
[2] https://owlready2.readthedocs.io/en/latest/
[3] https://sparqlwrapper.readthedocs.io/en/latest/
[4] https://rdflib.readthedocs.io/en/stable/

locations, events and scene objects can be selected from the graphical palette. To insert these elements into the palette it is necessary to first collect them from domain ontologies. This was done to ensure that all the selected elements were taken from authoritative sources. This component implements all functionalities defined in Section 4.2.2. This component was realized in **Python** and uses the Python's **OWLReady2**, **SPARQLWrapper** and **RDFLib** libraries.

- **Filtering Elements**: once the candidate elements have been selected it is necessary to create a specific palette for the single story. For this purpose, this component was created that allows one to choose the domain instances to be used in the history from the list of all candidate instances constructed in the first stage of the Methodology described in Section 4.2.1. In particular, this component implements all the functionalities defined in Section 4.2.3, including the choice of elements from the list of candidate domain instances or from other stories, and the management of duplicate domain instances. This component was implemented in **Python** and uses the Python's **OWLReady2**.

- **Story Querying**: this component uses the populated storytelling ontology produced by the "Story Creation" component as a knowledge base to make queries to identify possible connections between narrative elements of different stories. This component implements all functionalities defined in Section 4.2.5. The querying language used is **SPARQL**; the component was developed in **Python**, and uses the **SPARQLWrapper** and **RDFLib** libraries to make SPARQL queries, and uses the **OWLReady2** library to define new object properties to create inter-story links.

After having described the individual components of the architecture, the implementation of the use cases defined in Section 4.2.1 is now detailed using some UML Sequence Diagrams. Figure 4.24 shows the UML Sequence Diagram that explains the interaction between the architecture components during the execution of Phase "*Selection of Candidate Elements to Compose a Story*" and "*Creation of the Story and Choice of Domain Elements among the Candidate Elements*" illustrated in Sections 4.2.2 and 4.2.3. As can be seen in Figure 4.24, the first interaction takes place by the user who must authenticate to perform any type of operation. If the authentication is successful, it can lead to loading the global file containing all candidate elements. At this point, the semaphore component is contacted, which has the task of checking whether someone is currently working on the file. If this is the case, it returns a message in the frontend with the name of the person who is currently working thanks to the help of the Database Info Stories And User component, otherwise it downloads the file. Once downloaded, the user can add elements via the frontend by taking them from domain ontologies. In particular, the user sees the taxonomy of the selected domain ontology, from which a specific class can be chosen. From each class, the user

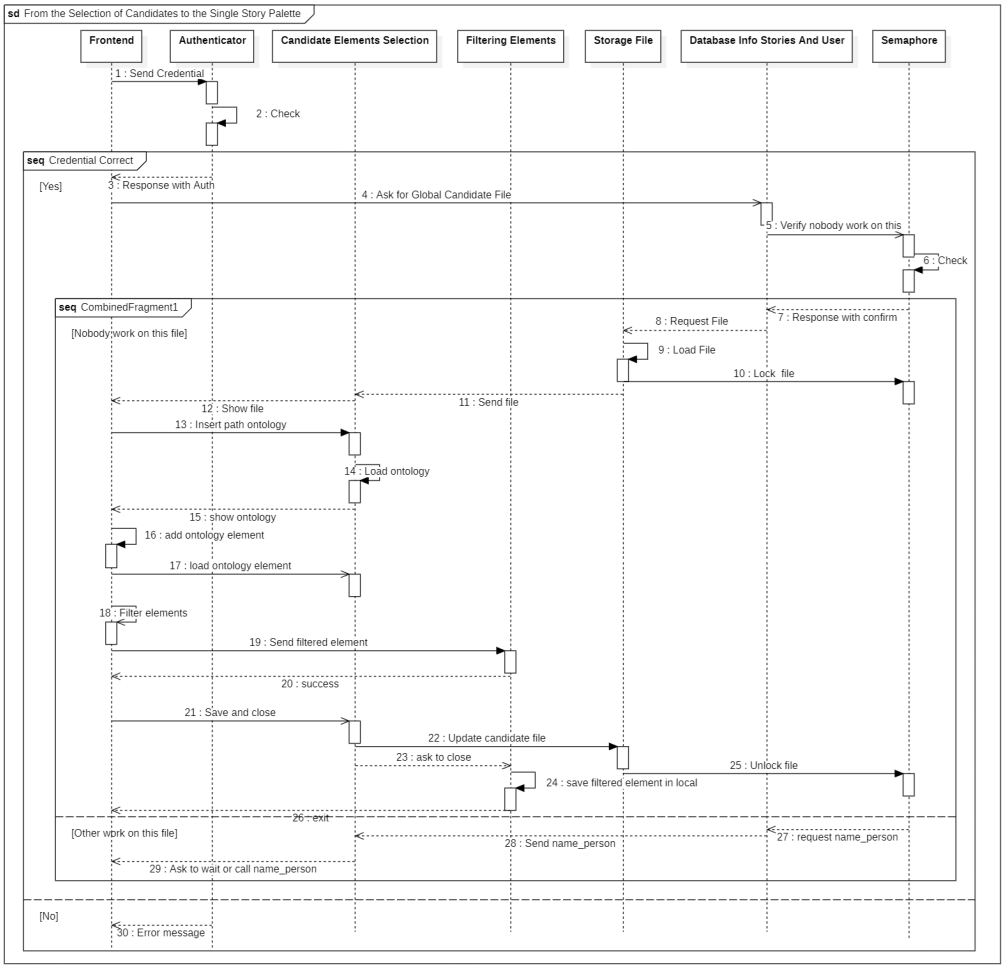


Figure 4.24: UML Sequence Diagram of the first two phases of the Methodology.

may choose all the domain instances of his or her interest and associate the corresponding storytelling element with them, producing a list of candidate elements as domain instances to be used to compose stories. Once completed in the next step, using the Filtering Elements component, the user can select the elements for the specific story to be created. Once this is done, the user can close and save the new palette created specifically for the construction of the single story. During the closing process, the file containing the candidate elements is updated and only after is the block released by the semaphore for the other users. Figure 4.25 shows the UML Sequence Diagram that explains the interaction between the architecture components during the execution of Phase "*Story Composition*" illustrated in Section 4.2.4. As it is possible to see from Figure 4.25, before creating a story, an authentication phase is required. Subsequently it's possible to either load a story from the database and in this case it is always checked with the semaphore mechanism whether someone else is working on it (if so the user's request is rejected) or it's possible create a story from scratch. Subsequently, the palette is loaded into the tool and the user can proceed to create the story via the Story Board and the Graphical Palette. The user can compose the story by inserting nodes and edges from the Story Board or by retrieving domain instances from the Graphical Palette. In this part there is an interaction with the Constraints Checker component that monitors the correct design of the story. When the story is finished, the user can save the story. Saving is done in two operations: i) a JSON file is created in the Storage File which can be used in future sessions to view the story in the Story Board; ii) the Storytelling ontology is retrieved from the Storage File, and it is populated with all the story elements modelled with the tool and then saved back in the Storage File.

4.2.8 Implementation of Dynamic Query

As reported in Section 4.2.5, once the script writer has modeled several stories with the tool, by means of a querying system, he can search all modeled stories for all narrative elements that satisfy certain constraints. The tool offers the user the possibility of graphically composing the query using the following mechanism:

- Query variables can be acquired by adding "blank nodes" from the Story Board, i.e. nodes of a specific type (e.g. scene nodes or action nodes) whose names and fields are not filled;
- The constants used in the query can be acquired from the Graphics Palette, which is populated with all the domain instances chosen as candidate elements in the first phase of the methodology (actors, locations and scene objects), or with all the narrative elements included in the stories modelled by the script writer (plot, scenes, actions and events).

The query is executed on a knowledge base consisting of the storytelling ontology populated

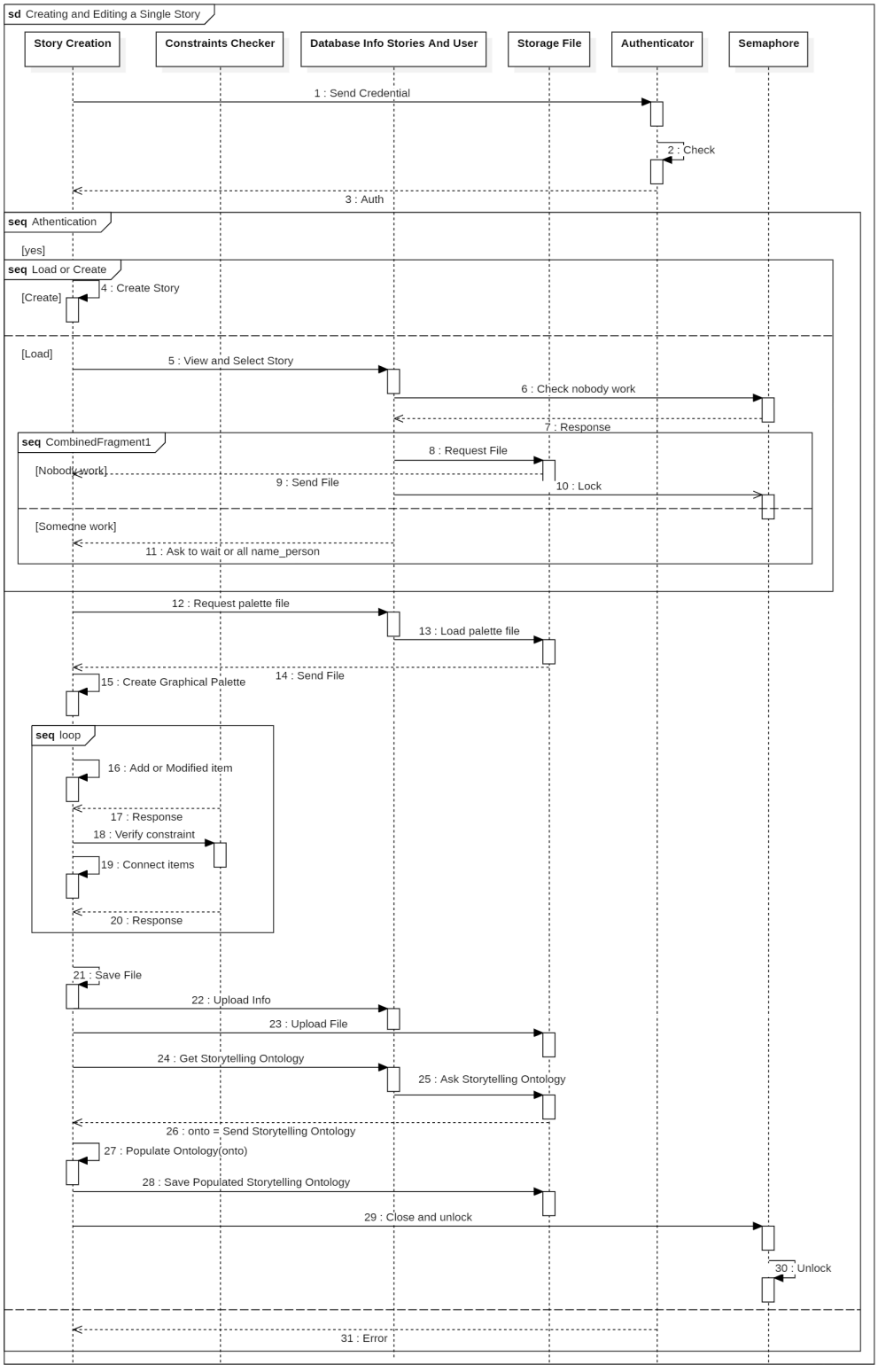


Figure 4.25: Sequence diagram of Phase ”Story Composition”.

with all story elements, which also imports all the domain ontologies used. The knowledge base was constructed in **OWL**, therefore, the language chosen to execute queries is **SPARQL**. The elements to perform the query are built graphically using the Story Board and the Graphics Palette, however, a module was implemented in Python that dynamically builds the SPARQL code of the query and then executes it using the *SPARQLWrapper* library. In Snippet 5 is the pseudocode that allows the query to be constructed.

Algorithm 5 Dynamic Query Generation

```
Require: DomainOntologiesList, ConstantNodesList, VariableNodesList
Ensure: querySPARQL
  querySPARQL = ""
                                                                                    ▷ add prefixes
  querySPARQL = querySPARQL + "PREFIX storytelling : <http :// www.w3.org/ storytelling #>"
  for all ont in DomainOntologiesList do
      querySPARQL = querySPARQL + "PREFIX " + ont.Prefix + ": <" + ont.iri + ">"
  end for
  querySPARQL = querySPARQL + "SELECT "
                                                                    ▷ add variables at SELECT
  for all node in VariableNodesList do
      querySPARQL = querySPARQL + "? + node.type + " "
  end for
  querySPARQL = querySPARQL + "WHERE { "
                                                                    ▷ add storytelling constraints
  for all node in ConstantNodesList do
      querySPARQL = querySPARQL + "? + node.name + " a " + node.type + " . "
      querySPARQL = buldStorytellingStructure(node.type)
  end for
                                                                    ▷ add domain constraints
  for all node in VariableNodesList do
      if node.type in ["Actor", "SceneObject", "Location"] then
          for all constr in node.constraints do
              if constr.typo == "typeOf" then
                  querySPARQL = querySPARQL + "? + node.name + " a " + getPrefix(node) + ": " + constr.class + " . "
              else if constr.type == "object property" then
                  querySPARQL = querySPARQL + "? + node.name + " " + constr.obj + " " + getPrefix(node) + ": " + constr.value + " . "
              end if
          end for
      else if node.type == "Scene" and node.date exists then
          if node.date is constant then
              querySPARQL = querySPARQL + "? + node.name + " < storytelling :hasTime> \"" + node.date + "\" . "
          else
              querySPARQL = querySPARQL + "? + node.name + " < storytelling :hasDate> ? + node.name + _date . "
          end if
      end if
  end for
  querySPARQL = querySPARQL + "} "
                                                                    ▷ return query return querySPARQL
```

As shown in Snippet 5, the algorithm must take as input a list of used domain ontologies, a list of constant nodes, and a list of variable nodes. For each domain ontology used, a prefix must be generated in order to use the assertions defined in that ontology in the query. The prefix of the storytelling ontology containing the semantic representation of all stories modeled with the tool is also defined. In the query, the variable nodes will correspond to the elements in the "SELECT" clause. The presence of one or more constant nodes, on the other hand, adds triple patterns in the "WHERE" clause to the query; these triple patterns correspond to storytelling constraints that must

be satisfied. For example, the presence of a constant node of type ”Actor” linked to a variable node of type ”Scene” implies that one must search for all scenes in all stories in which that specific actor appears. These storytelling constraints will be entered by the *buldStorytellingStructure()* function, which takes as input the node type (e.g., actor, scene object, or location), and retrieves all the action structure in which that node appears: for example, given a constant node of type ”Actor”, the function retrieves all the structure of actions in which that actor appears, scenes in which those actions are present, plots belonging to those scenes and history defining those plots.

For each variable node, two cases must be treated separately: i) if the node is of type ”scene”, ”event”, ”plot” or ”action” the node doesn't add constraints to be inserted in the ”WHERE” clause of the query; ii) if the node is of type ”actor”, ”scene object” or ”location”, the list of constraints inserted in the node must be retrieved. Constraints can be of two types: 1) Constraints of type ”*typeOf*”, which add a triple patterns to search in a domain ontology for individuals belonging to the class indicated as the range of typeOf property (for example all individuals of the class ”King”); 2) Constraints of type ”object properties”, which add a triple patterns to search in a domain ontology for individuals that satisfy specific properties (for example all actors who were born in ”Vienna” and have reigned in ”Madrid”). A particular case is given by the scene node, which even if defined as variable node, can have a constraint in the date. The date can be a variable or a constant. If the date is a constant, it's necessary to add an additional triple pattern in the ”WHERE” clause of the query, because it means that the aim is to search all the events that occurred on a given date. To insert this last triple pattern must be used the object property ”*hasTime*” defined in the Storytelling Ontology.

4.2.9 Case Study

In this section, an example of the use of the StoryCR8 toolkit is presented. A case study dealing with historical narratives from the period of the construction of the Royal Palace of Caserta[5] and the Bourbons' domination in Naples[6], which spans the period between 1700 and 1800 in the Kingdom of Naples[7].

The first step in using the tool is to select the domain ontologies to be used. In this case study,

[5]The Royal Palace of Caserta is a royal residence, historically belonging to the Bourbons of the Two Sicilies, located in Caserta. Commissioned by Charles of Bourbon, the first stone was laid on 20 January 1752 to start construction work, based on a design by Luigi Vanvitelli: this was followed by his son Charles and other architects. The palace was completed in 1845. Together with the Carolino aqueduct and the San Leucio belvedere, it was included in the UNESCO World Heritage list in 1997.

[6]The Bourbons are one of the most important and oldest royal families in Europe. Of French origin, the family is a cadet branch of the ancient Capetian dynasty, which, following the extinction of the other branches, inherited the throne of France in 1589.

[7]The Kingdom of Naples was a state that ruled the part of the Italian Peninsula south of the Papal States between 1282 and 1816.

an OWL-based simple ontology was produced that models historical characters from the Bourbon period. As far as the locations are concerned, the "**Locations Ontology**"[8] was used as the domain ontology, populated with open data[9] from the list of world nations and Italian cities, regions, and municipalities. A view of the populated Location Ontology is shown in Figure 4.26.

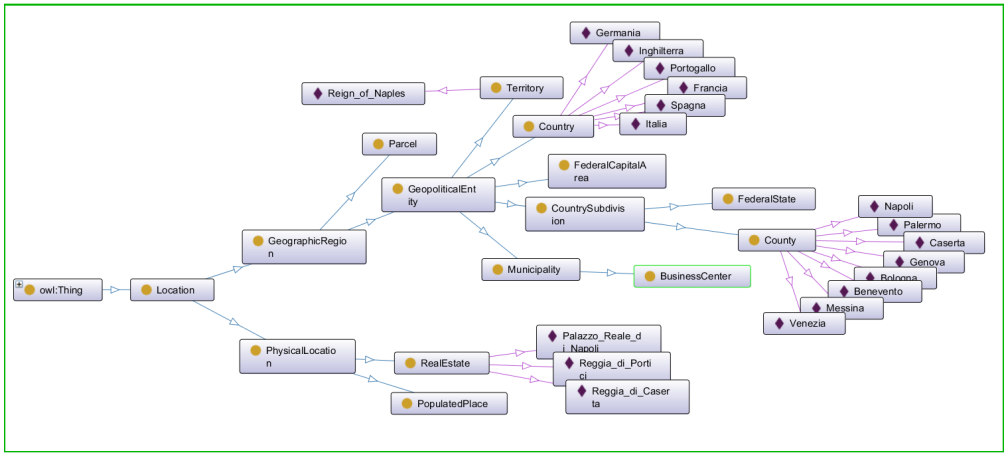


Figure 4.26: View of the Location Ontology Populated with Open Data of Italian Cities and Municipalities.

As the Figure 4.26 shows, the Location Ontology provides a very high level definition of geographic region and geopolitical entity related concepts, including, but not limited to, countries, sub-country regions such as states and provinces, municipalities, etc. Populating the ontology, individuals were added regarding nations, cities and and historic buildings.

After choosing the domain ontologies to be used, it is possible to use the tool to perform the "Selection of Candidate Elements to Compose a Story" activity described in Section 4.2.2. An example of candidate selection from the domain ontology modeling historical characters from the Bourbon period is shown in Figure 4.27.

As can be seen from Figure 4.27, the graphical interface of the candidate selection tool consists of several boxes, detailed below:

- *Class Ontology Box*: in this box, the taxonomy of classes of the selected domain ontology can be explored. Only one domain ontology can be selected at a time. The chosen domain ontology is very simple, containing a few concepts relating to historical characters, e.g. "*King*", "*Queen*", "*Architect*", "*Minister*", "*Dynasty*", etc.

- *Class Instances Box*: in this box all instances of the class selected from the Class Ontology Box appear. in the example shown in Figure 4.27, the class "*King*" was selected and all

[8] https://spec.edmcouncil.org/fibo/ontology/FND/Places/Locations/
[9] https://www.gardainformatica.it/database-comuni-italiani

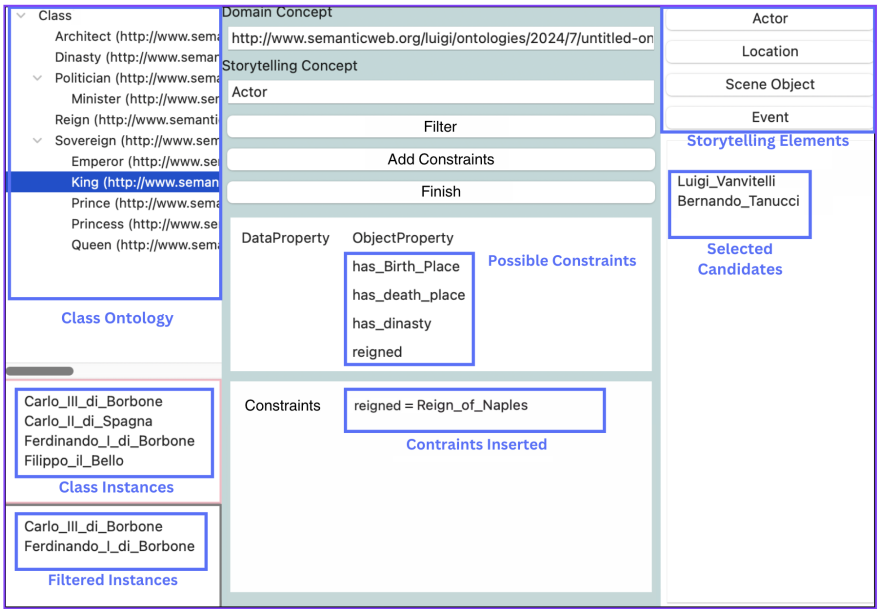


Figure 4.27: Use of the Candidate Element Selection Tool from Domain Ontologies

individuals of that class in the ontology appear, e.g. "*Charles III of Bourbon*", "*Ferdinand I of Bourbon*" and "*Charles II of Spain*".

- *Storytelling Element*: from this box, the narrative element to be associated with all the domain instances chosen by the selected class must be selected. For the "*King*" class, the "*Actor*" element was chosen, since the aim is to use some individuals of the "*King*" class as characters in the story.

- *Possible Constraints*: for sufficiently complex ontologies a filtering mechanism has also been implemented, based on object and data properties of domain ontologies, to improve and speed up the search. If the user chooses to enter a constraint, all object properties and data properties that have the chosen class (or one of its superclasses) as a domain will be shown in this box. Once a property has been chosen, from a combobox it's possible to choose one of the values that property has been given within the domain ontology. For example, if the object property "*reigned*" is chosen, from the combobox the user will be able to choose from all the kingdoms over which the sovereigns modeled in the ontology have reigned.

- *Constraints Inserted*: all the restrictions that have been chosen appear in this box. For example, in Figure 4.27 it's possible to see that only one constraint was chosen, namely to select all and only the kings who ruled over the "*Kingdom of Naples*".

- *Filtered Instances*: in this box appear all individuals of the class "*King*" that satisfy the constraints defined in the box "Constraints Inserted". From the example in Figure 4.27 it can

be seen that the kings who reigned in "*Kingdom of Naples*", among all those in the "Class Instances" box, are "*Charles III of Bourbon*" and "*Ferdinand I of Bourbon*".

- *Selected Candidates*: in this box appear all individuals that have been chosen as candidate domain instances from the "Class Instances" box or the "Filtered Instances" box. For each candidate in the list, some information is stored such as the domain ontology it belongs to, the class it belongs to, and the storytelling element that has been associated with it.

This mechanism makes it easy to select different elements from different ontologies. Once all candidate elements have been chosen from the two domain ontologies, the story can be created and the character can choose which of these candidate elements become domain instances to be used in the story, creating the appropriate palette of domain instances of actors, locations, and scene objects to be used to compose the story with the Story Board. In this case study, two different stories were modeled: i) a story about the construction of the Royal Palace of Caserta; a story about the life of the famous architect Luigi Vanvitelli[10].

A view of the story of the "construction of the Royal Palace of Caserta" modeled using the Story Board of the tool is shown in Figure 4.28. As can be seen from Figure 4.28, at the bottom

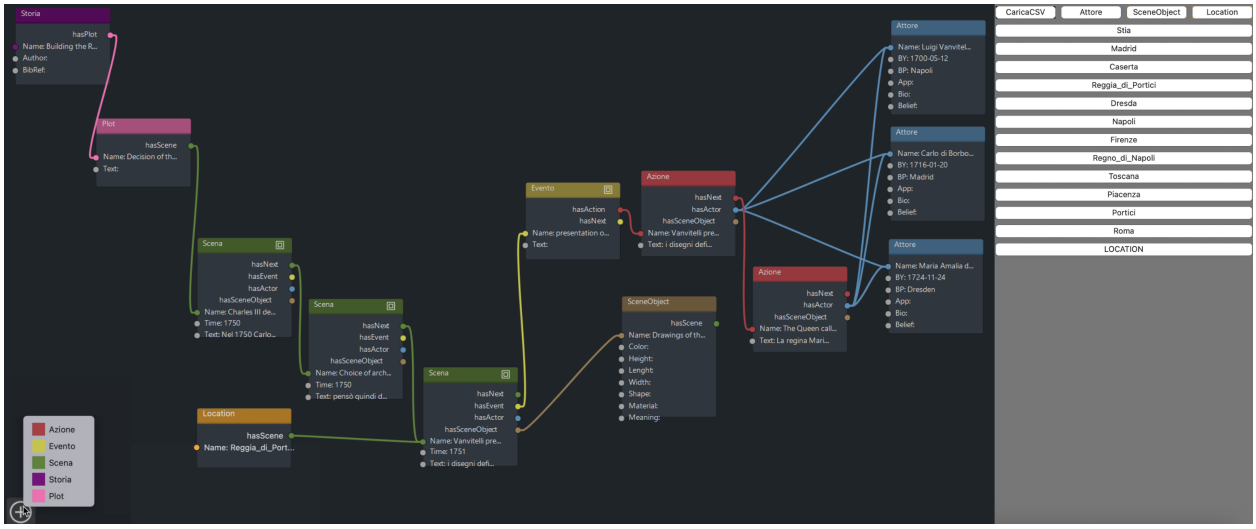


Figure 4.28: Creating the Story "Construction of the Royal Palace of Caserta" with StoryCR8.

left of Story Board are the nodes that it possible create from scratch. Clicking on one of these elements opens a form that can be filled in with the storytelling characteristics (e.g., the name, the text, the time of a scene, etc.) of the respective element clicked on. To the left of the Story Board, on the other hand, is the Graphical Palette containing the domain instances selected in the previous

[10]Luigi Vanvitelli was an Italian architect and painter. The most prominent 18th-century architect of Italy, he practised a sober classicising academic Late Baroque style that made an easy transition to Neoclassicism.

step grouped by type (Actors, Locations and Scene Objects). Finally, in the centre, the Story Board is presented in which the user can insert nodes and connect them appropriately according to what are the constraints defined by the ”Constraint Checker” component. In Figure 4.28, only the modeling of the first plot in the story of the ”Construction of the Royal Palace of Caserta,” which tells of the decision to build a new royal palace in Caserta, has been shown. The plot consists of three scenes carried out in the following order:

1. *Scene 1*: tells of King ”*Charles III of Bourbon*”'s decision to build a new royal residence. The scene takes place in 1750 in the Royal Palace of Portici, Italy. In Figure the events, actions and characters in the scene were not shown.

2. *Scene 2*: tells of the choice of architect to design the Royal Palace of Caserta. King ”*Charles III of Bourbon*” analyzes several designs before choosing ”*Luigi Vanvitelli*”'s. The scene takes place in 1750 in the Royal Palace of Portici, Italy. In Figure the events, actions and characters in the scene were not shown.

3. *Scene 3*: tells of architect ”*Luigi Vanvitelli*”'s presentation of the design of the Royal Palace of Caserta in the Royal Palace of Portici in 1751. The scene consists of two consecutive actions. In the first action ”*Luigi Vanvitelli*” presents the designs of the Royal Palace to the sovereigns ”*Charles III of Bourbon*” and his wife ”*Maria Amalia of Saxony*”. In the second action, queen ”*Maria Amalia of Saxony*” asks the architect ”*Luigi Vanvitelli*” to also draw up an initial master plan for the city of Caserta. In the first action all three characters participate, while in the second action only ”*Luigi Vanvitelli*” and ”*Maria Amalia*” participate. The scene object ”*Drawings of the Royal Palace of Caserta*” is also used in the scene.

The stories modelled with the tool are inputted into the Parser ontology, which populates the Storytelling Ontology with all the components of the created stories. An example of a storytelling ontology populated with the two stories created can be seen in Figure 4.29.

As it is possible to see from Figure 4.29, all the scene nodes modelled with the tool have become individuals of the class ”*Scenes*”, and all the edges that connect the scene nodes have become object properties of type ”*hasNext*” (orange edges in Figure). All the elements inserted in the Story Board by the Graphical Palette in the storytelling ontology become individuals of the classes ”*LivingBeings*”, ”*Location*” or ”*SceneObject*”, which through a property of type ”*same individual as*” are linked to the respective domain instance defined in one of the domain ontologies used (all domain ontologies are imported into the populated storytelling ontology).

So, all the components that have been graphically modeled have been used to populate the storytelling ontology, which now becomes a rich knowledge base on which it is possible to run SPARQL queries. The tool, in fact, provides a special Story Board to graphically perform a

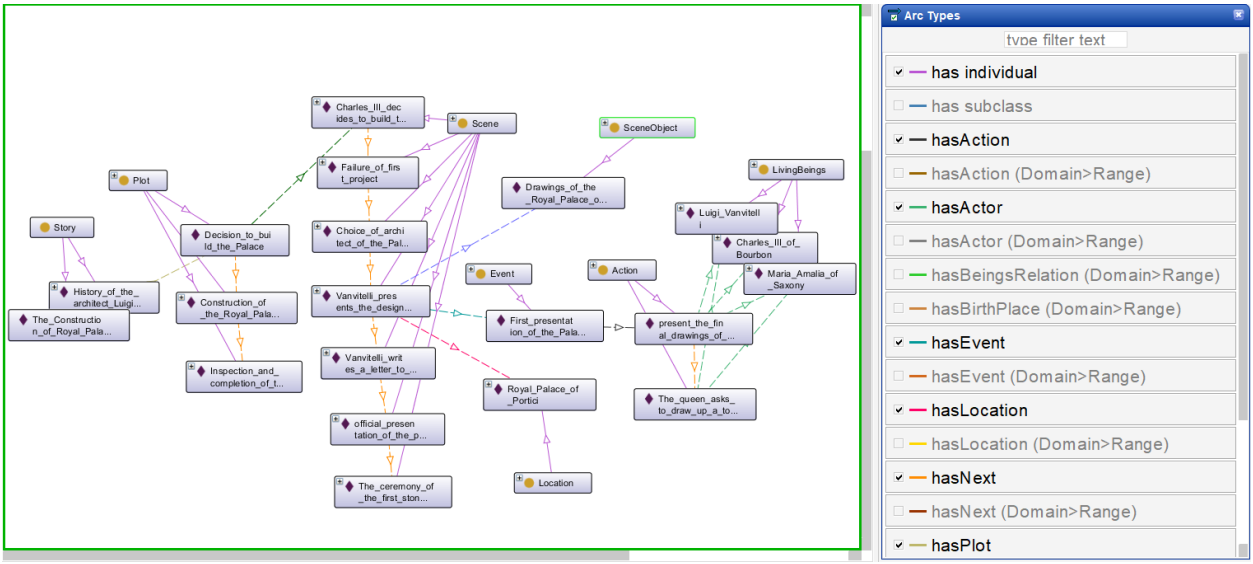


Figure 4.29: Storytelling Ontology Populated with the two stories modeled with StoryCR8.

SPARQL query to search all the narrative components in all the modelled stories that satisfy certain constraints executing the algorithm defined in Section 4.2.8. Figure 4.30 shows an example of a query graphically constructed by the tool.

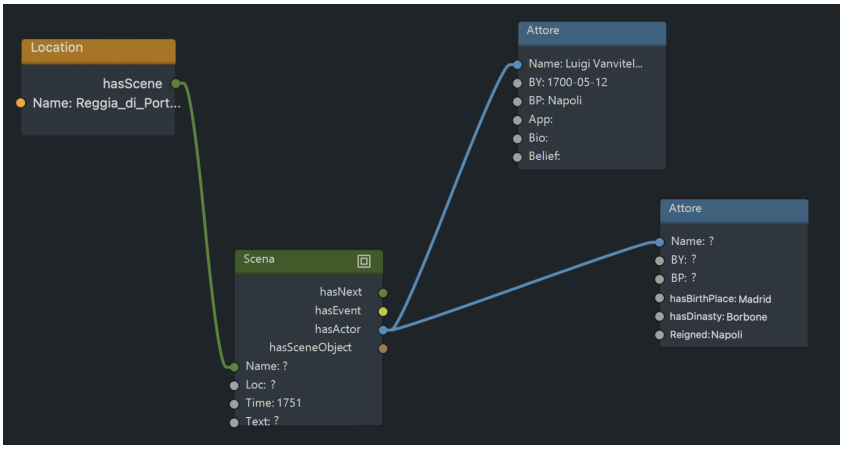


Figure 4.30: Example of Graphical Construction of the Query.

As can be seen in Figure 4.30, two variables were inserted in the query: i) the first variable is of type "*Scene*" and has the field "*Time*" unified to the value "*1751*". It indicates that the aim is to search all the scenes of all the stories stored in the Storytelling Ontology that happened in 1751 and that satisfy the constraints imposed by the constants. ii) the second variable is of type "*actor*" and has three domain constraints to respect. It indicates that in the scene must appear at least one character born in "*Madrid*", of the "*Bourbon*" dynasty and who reigned over the "*Kingdom of Naples*". In the query there are also two constants: the location "*Royal Palace of Portici*" and

the actor "*Luigi Vanvitelli*". Applying the algorithm described in Section 4.2.8, the dynamically generated SPARQL query is illustrated in the snippet 4.1.

Listing 4.1: Dynamically Generated SPARQL Query.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX storytelling: <http://www.w3.org/storytelling#>
PREFIX dom1: <https://spec.edmcouncil.org/fibo/ontology/FND/Places/Locations/#>
PREFIX dom2: <http://www.semanticweb.org/customHistoryOntology#>

SELECT ?Action ?Event ?Scene ?Plot ?Story ?DomainActor
    WHERE{
        {?Action a storytelling:Action}.
        {?Event storytelling:hasAction ?Action}.
        {?Scene storytelling:hasEvent ?Event}.
        {?Plot storytelling:hasScene ?Scene}.
        {?Story storytelling:hasPlot ?Plot}.
        {?Action storytelling:hasActor storytelling:LuigiVanvitelli}.
        {?Action storytelling:hasLocation storytelling:RoyalPalaceOfPortici}.
        {?Action storytelling:hasActor ?Actor}.
        {?Action storytelling:hasActor ?Actor}.
        {?DomainActor owl:sameAs ?Actor}.
        {?DomainActor dom2:hasBirthPlace dom1:Madrid}.
        {?DomainActor dom2:hasDinasty dom2:Borbone}.
        {?DomainActor dom2:reigned dom2:Napoli}.
      }
```

The result of the SPARQL query shown in the Snippet 4.1 is shown in Figure 4.31.

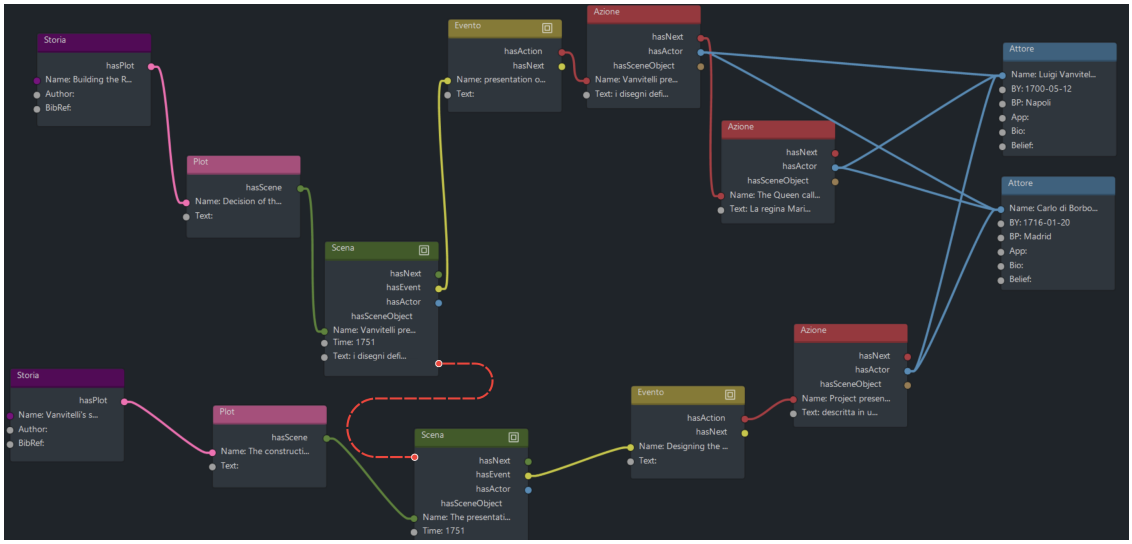


Figure 4.31: Result of SPARQL Query.

As it's possible to see from Figure 4.31, the query returned all the scenes of all the modelled stories that happened in "*1751*" at the "*Royal Palace of Portici*" which included the participation of "*Luigi Vanvitelli*" and at least one actor born in "*Madrid*", of the "*Bourbon*" dynasty and who reigned over the "*Kingdom of Naples*". The only character who respects these domain's constraint

is "*Charles III of Bourbon*", which is the only actor that appears in the result except "*Luigi Vanvitelli*". Furthermore, it is possible to see from Figure 4.31 that the entire structure of storytelling has been returned, showing not only the scenes but also the actions of the scenes, events, plots and associated stories. The user can use the result of a query to identify narrative components of different stories that have a "fil rouge" relationship according to the methodology described in Section 4.2.5. For example, in Figure 4.31 it's present a fil rouge (a red curved edge connecting two scenes): from the result of the query it emerged that the same scene, inherent to the first presentation of the project of the Royal Palace of Caserta, was narrated in two different stories. The tool offers the possibility to insert the link between the two scenes, which in the Populated Storytelling Ontology is modeled as an object property that connects two narrative elements of different stories. Figure 4.32 shows the semantic representation of fil rouge that connects the two different scenes.

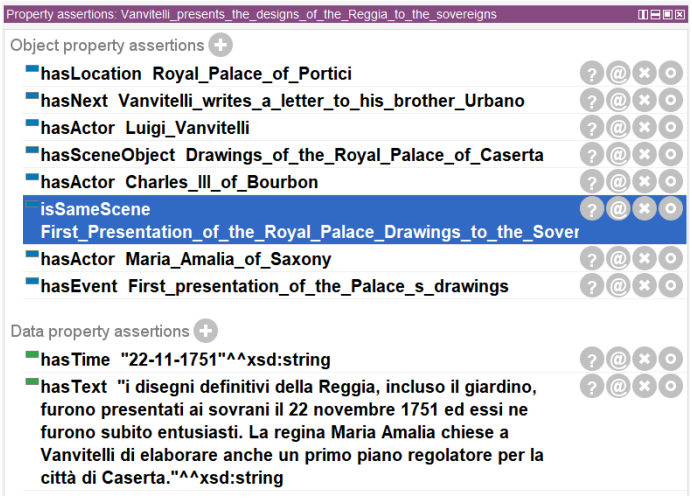


Figure 4.32: Semantic Representation of Fil Rouge.

Figure 4.32 shows the semantic representation of the "*Vanvitelli presents the designs of the Reggia to the sovereigns*" scene in the ontology of storytelling. The fil rouge has been modeled as an object property called "*isSameScene*" that links the individual "*Vanvitelli presents the designs of the Reggia to the sovereigns*" of the "*Scene*" class with the individual "*First Presentation of the Royal Palace Drawings to the Sovereigns*" of the "*Scene*" class. The tool offers the user the possibility to define all the relations of common thread of his interest. Finally, the tool also offers the possibility of associating a digital content to a node. In this case study it was chosen to associate some narrative components of the two stories modelled an IIIF annotation. Using the IIIF format it was possible to create a first prototype of a system for the fruition of digital content associated with history. Figure 4.33 shows an example of digital content in IIIF format associated with the scene object "*Drawings of the Royal Palace of Caserta*".

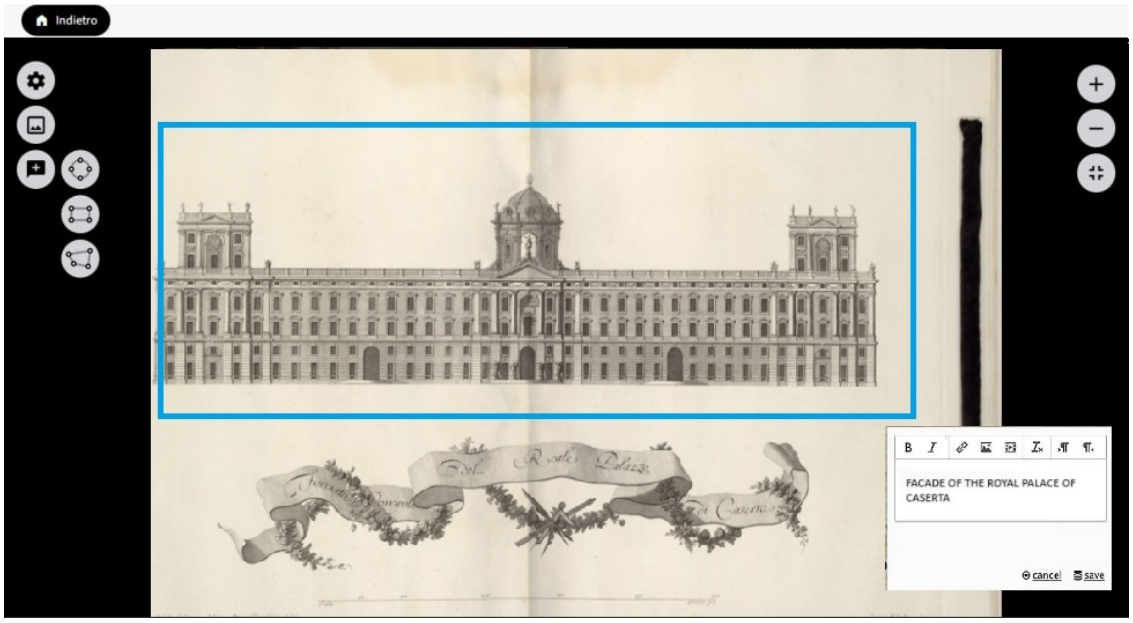


Figure 4.33: Example of Digital Content associated at Scene Object ”Drawings of the Royal Palace of Caserta” in IIIF Format.

4.3 Automatization of CR8 Based to Text

After realizing the tool and the supporting architecture a further advancement was the autometric identification of the characterizing elements of a text, using Machine Learning and Natural Language Processing (NLP) techniques. The goal is to develop an algorithm that allows the automatic division of a text into sequences, based on a multi-objective function. Each sequence will have well-defined boundaries, and the rule describing the transition from one sequence to another will be clear. To achieve the set goal, the project was divided into subgoals:

1. **Preprocessing and data search**: a preliminary search of the data to be analyzed was conducted, and then the preprocessing techniques mentioned in the state of the art were applied.

2. **Named Entity Recognition (NER)**: a preliminary analysis of the text was performed by filtering the entities representing some elements of the narrative text [141].

3. **Date Structure Analysis**: after identifying the entity “date,” it was necessary to reconstruct the date structure to understand whether there were multiple instances referring to the same time interval.

4. **Part-of-Speech (POS) analysis and identification of pseudonyms**: after identifying the entity “people,” it was necessary to develop a technique for grouping the various names

referring to the same actor, creating as many clusters as there are actors identified, including their respective pseudonyms in each cluster.

5. **Sequence Identification**: exploiting the features that determine the transition from one sequence to another, a rule-based approach was used to identify the structure of the narrative text in terms of sequences.

6. **validation of results**: once the evaluation metrics are defined, they will be applied to the results obtained to estimate the correctness of the developed algorithm.

4.3.1 Methodology

To achieve the objectives listed above, a clear methodology has been established, illustrated in Fig.4.34, which will be described in this chapter.

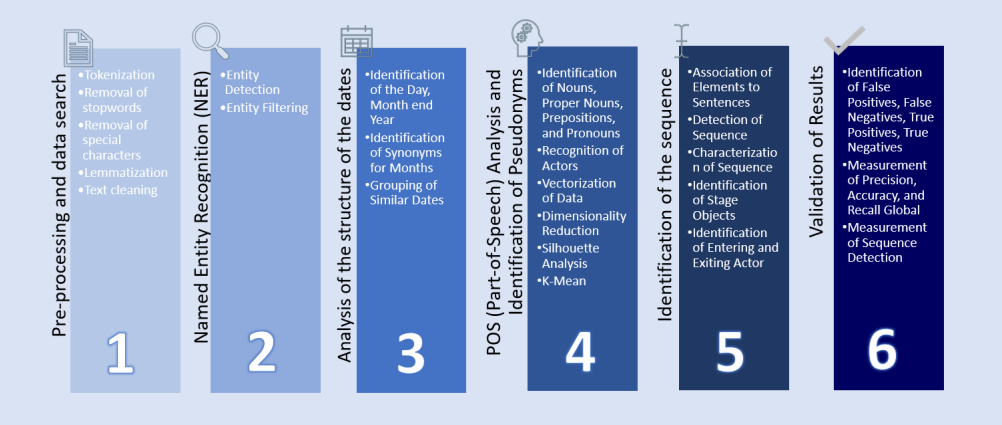


Figure 4.34: Description of the methodology applied to achieve the objectives

Pre-processing and data search were conducted preliminarily. Given the lack of Italian datasets labeled with narrative text elements, it was decided to work on the texts provided by the RASTA project. The data were pre-processed by tokenization, removal of stopwords and special characters, lemmatization, and text cleaning. The transition between sequences is identified by the change of actors, place or time. During **Named Entity Recognition (NER)**, the entities extracted are actors, places, dates and times. However, only a subset of these entities is relevant to the goal, and so only those needed are filtered and extracted. These entities are then archived for use in later steps. In **date structure analysis**, the instances of the category "DATE" recognized in the previous step are analyzed to avoid errors related to variations in the representation of the same dates (e.g., misspellings). Day, month and year are extracted where available, also recognizing synonyms of months. Dates with similar characteristics are grouped together to evaluate them as referring to the same time period. For historical texts or texts referring to real people, it is common for a person to

be mentioned by various names (e.g., first name, last name, title, or nickname). This represents a limitation in simple textual comparison. For this reason, a **POS (Part-of-Speech) and pseudonym identification** analysis was performed, using POS labels along with NER labels to recognize the entities of "Actors." The textual data were vectorized using the *Term Frequency - Inverse Document Frequency* technique. To simplify the data, size reduction was performed, and *Silhouette Analysis* [142] was used to determine the number of clusters. Final clustering was obtained with the *K-Means* technique, and the results were validated by human intervention. To **identify the sequence**, the elements identified in the previous steps were used. Each recognized entity was associated with the corresponding sentence, and through sentence-by-sentence comparison, variations indicating the transition from one sequence to another were identified. Subsequently, non-consecutive sentences were also examined to identify the continuation of a sequence. Finally, a characterization of the elements of the sequence was made, including *objects of the scene* and the definition of actors as *entering* or *exiting*.

4.3.2 Implementation and Testing

To achieve the goals of the project, *Python* was used as the programming language, and the main libraries employed were: *Numpy*, *Pandas* [143], *Scikit-learn* [144], *NLTK* and *Spacy*. The project was implemented on *Google Colab*. During the **pre-processing and data search** phase, the data were processed using the NLTK and Spacy libraries [145]. For **Named Entity Recognition (NER)**, the *it_nernerIta_trf* model was employed, based on the *hseBert-it-cased* transformer, which recognizes 18 different entities (e.g., PER, GPE, DATE, TIME). User-supplied input text was processed with the Spacy model, and only relevant entities were filtered and stored for further processing. In the **date structure analysis** phase, date information was extracted and stored in a DataFrame, with columns representing day, month, and year. To handle misspellings or synonyms, the function *levehenstein_distance* was used to correct for variations. Finally, dates with similar characteristics were grouped in the same cluster In the **POS analysis and pseudonym identification** stage, actors were recognized using NER and part-of-speech (POS) labeling. Defined rules made it possible to identify proper and common names that referred to actors. Next, textual data were vectorized using *TfidVectorizer*, and dimensional reduction was performed using *Principal Component Analysis* (PCA). Clustering was then performed using *K-Means*, with silhouette analysis to determine the optimal number of clusters. The results were manually validated to correct any errors. In the **sequence identification** phase, the recognized entities were associated with their respective sentences, and changes between sentences were monitored to identify the transition between sequences. The transition between sequences was detected when a change in actors, places, dates, or times occurred. Once identified, sequences were characterized by including information

about participating actors, places, dates, props, and the entry or exit of characters. Further processing identified verbs that could indicate a character's exit, and it was tested whether multiple sequences with the same characteristics should be combined into a single sequence.

4.3.3 Validation

In the evaluation of algorithms, especially in classification contexts, several metrics are used, including the four basic outcomes: true positive (TP), true negative (TN), false positive (FP), and false negative (FN) [146]. From these, accuracy, precision and recall can be mathematically defined as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{Precision} = \frac{TP}{TP+FP} \quad \text{Recall} = \frac{TP}{TP+FN}$$

The values obtained for the three selected metrics were as follows:

- Global Accuracy = 0.82
- Global Precision = 0.85
- Global Recall = 0.83

These values were obtained by considering the four entities that discriminate the transition from one sequence to another. Taking these values into account, it was possible to evaluate the recognition of the sequences globally, obtaining a value of 0.82. In addition, to obtain a clearer view of the more difficult-to-identify elements, more specific vertical measurements were made on individual elements in addition to a general measurement. Below, Table 4.24.34.4 shows how the various parameters were measured. These are just some of the measurements made to evaluate the algorithm. Looking at these results, we can draw some important conclusions. In particular, we can see that dates and times return much more reliable results than places and actors. This result was expected, since the latter, due to nicknames, pseudonyms, and other factors, are more difficult to identify than a string referring to a date or time. Also, in many cases, the accuracy and precision of actors decrease, often due to a higher number of false positives, caused by the fact that in some texts the subject is not a person but an object.

Entity	Accuracy	Precision	Recall
Actor	0.85	0.96	0.88
Location	0.84	0.85	0.88
Date	1.00	1.00	1.00
Time	1.00	1.00	1.00

Table 4.2: Text1 - Accuracy, Precision, and Recall

Entity	Accuracy	Precision	Recall
Actor	0.58	0.69	0.79
Location	0.86	0.75	1.00
Date	0.94	0.94	0.94
Time	0.89	0.33	1.00

Table 4.3: Text2 - Accuracy, Precision, and Recall

Entity	Accuracy	Precision	Recall
Actor	0.71	0.83	0.82
Location	0.82	0.68	0.92
Date	0.98	0.97	1.00
Time	0.98	0.67	1.00

Table 4.4: Text3 - Accuracy, Precision, and Recall

Chapter 5

Machine, Deep and Federated Learning

In recent years, learning technologies, in all their forms, such as machine, deep, reinforcement, and even federated, have revolutionized many sectors, enabling advanced solutions [147] for data management, process optimization, and predictive analytics. In this era of big data [148] [149], these technologies are a key focus for the ever-increasing demand for intelligent systems that can make autonomous decisions and optimize complex performance. In this context, throughout the Ph.D. period, different projects on different application domains were defined, designed, implemented, deployed, and tested with the aforementioned software-supporting architectures. The research path began with studying energy communities in collaboration with ENEA, focusing on applying the main machine learning models, with particular reference to clustering. This initial phase provided a solid understanding of analytical tools, fostering initial experience in using advanced data analysis techniques. This work culminated respectively with two publications at the 16-th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2022) [150] [151] and the publication of a report on the ENEA website entitled: "Tecniche di Machine Learning, Big Data Analytics e Natural Processing con applicazione all'analisi di Social Media". Subsequently, the work was extended to the military defense domain, where an in-depth study of key cloud agnostic patterns was conducted. The goal was to define an architecture to support a military supply chain in a federated context, considering the challenges and opportunities distributed infrastructure management offers. This work was published at the international conference "2023 IEEE International Workshop on Technologies for Defense and Security" [152]. At this stage, an advanced understanding of the architectural paradigms needed to ensure security and efficiency in highly critical environments was established. Continuing, at the IIIA in Barcelona, the topic of air quality was explored in depth using federated algorithms and cloud-edge patterns. This work, published at "The 19th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing", provided additional expertise in distributed architectures and the management of data from multiple

sources in an environmental monitoring context, to recognize recurring patterns associated with some actions of the inhabitants. Finally, in the last phase of the research, the definition, implementation, and deployment of different architectures, centralized and federated on real machines, was carried out. These architectures were used for monitoring and analysis of diabetic patients, to compare the different solutions in terms of architectural advantages and disadvantages, as well as deployment efficiency. Part of this work has been published respectively at "The International Conference of Advanced Information Networking and Applications (AINA 2023)" [16] and in the journal "Connection Science Volume 35, 2023" [92]. Specifically, in these 2 papers, a preliminary analysis was performed by analyzing as many machine and deep learning algorithms as possible on real data from diabetic patients. The aim was to indivisuate the best algorithm that balanced computational load, execution time, and correctness of the result. Instead, the rest of the work with deployment on real machines will soon be published in Q1 journal

5.1 A Platform for Monitoring Social Media for Energy Communities

The first work illustrated describes the research activities carried out in collaboration with ENEA aimed at defining a pipeline capable of examining news collected from varied textual sources in the Italian language and, from these, filtering only content related to the topic of energy communities and retrieving information related to the events described. Two conference papers have been published on this work, which are respectively, "Machine learning, big data analytics and natural language processing techniques with application to social media analysis for energy communities" [151] and "ECListener: A Platform for Monitoring Energy Communities." [150]. The system developed, therefore, is based on Natural Language Processing (NLP) [153], Process Mining [154], and Sentiment Analysis technologies [120] and is designed to operate in Batch mode. This approach allows large volumes of data to be processed at different times of the day while maintaining compatibility with a streaming analysis thanks to a scalable and stream-ready architecture. The treated news corpus exhibits characteristics of unstructured texts, with large data requiring prenormalization processes. In particular, as we will see later, neural networks and Word Embedding [155] techniques were employed to classify texts, validate ontologies, and identify semantic clusters relevant to the analyzed domain. One of the central issues and, simultaneously, the most complex aspects encountered in the analysis concerns the difficulty of unambiguously defining the concept of Energy Communities, an area of study that is constantly evolving. This required the development of automated solutions capable of recognizing relevant news items through the use of key concepts, both predefined and identified through subsequent semantic analysis. For this

reason, the software was designed to follow multiple approaches in text relevance analysis, event extraction, and Sentiment Analysis to provide a complete and reliable picture of the extracted information. The structure of this work will be divided into several sections. In the first part, the general objectives of the software will be explained, and a detailed description of the architecture and main components, including the modules for text analysis and semantic classification, will be provided. Next, each system component will be described in terms of its specific functionality, technologies employed, and results obtained. Finally, the work cited here will conclude with a critical analysis of the results achieved, highlighting the system's future potential and outlining possible developments to further refine Energy Community news analysis.

5.1.1 Objectives and Workflow

The principal objectives that were completed in this work are listed below:

- **Identification of News Relevant to Energy Communities**: to effectively analyze news, it is essential to distinguish between relevant and irrelevant news. This process, called "Relevance Analysis," uses a component called "Validator." News identification is an essential preliminary step that includes retrieving and cleaning the texts to prepare them for the subsequent analysis, described in a dedicated section.

- **Collection of information on events related to Energy Communities**: regarding Energy Communities, we are interested in different types of events including: conventions, conferences and workshops, which involve the national scientific community in research and provide opportunities for public outreach, raising public awareness.

- **Verification and ontology enrichment**: during software development, the ARAKNE ontology, based on concepts from Wikipedia to describe Energy Communities, was available. Although rich, this ontology needed validation through analyzing a document corpus provided by ENEA to verify its adherence to the topic. However, a new ontology, currently in its initial stages, was created to better describe the relevant events, enriched with data collected from the various analysis components.

- **Visualization and accessibility of information**: finally, the information collected through the software must be easily accessible and usable by users. Therefore, several visualization mechanisms have been developed based on standard technologies and data representation techniques, which will be discussed in more detail later.

To achieve these goals, a pipeline of tasks was defined to be performed Fig 5.1. The first task, called "Data Ingestion," deals with collecting news data and performing cleanup tasks, as the texts

contain HTML structuring that is irrelevant for analysis and needs proper pretreatment to be appropriately used in the following steps. Next, the "Relevance Classification" task aims to determine whether the collected news items are relevant to the topic of interest. To this end, a component was developed using two techniques: one based on the Spacy library and the other exploiting Word Embedding and neural networks. Another crucial step is the Named Entity Recognition (NER) task, which aims to recognize events of interest and associated information. Several approaches are taken here, including using Spacy and specially written regular expressions to complement existing tools. The "Semantic Expansion" task analyzes terms in the ontology and corpus texts to validate semantic concepts and relationships through techniques such as Wordnet and TF-IDF (term frequency-inverse document frequency). These data were cross-referenced with those derived from the previous task, looking for synonyms, hypernyms, and hyponyms. However, only the TF-IDF technique produced satisfactory results, particularly validating the relevance of concepts in the ontology. The next task, "Sentiment Analysis," analyzes opinions expressed in texts regarding positivity, neutrality, and negativity. Several techniques were tested, including Textblob, which proved ineffective, and a Bert-based neural network, which offered better results. In parallel, the "Populating Ontology" task proceeds without depending directly on the result of Sentiment Analysis, focusing on inserting the information extracted from the texts and visualizing it in an ontological structure. Finally, the "Visualization" of information is a critical step in making the analysis results usable. Several tools and technologies were examined, all based on known standards and libraries, to ensure the effective presentation of the processed data.

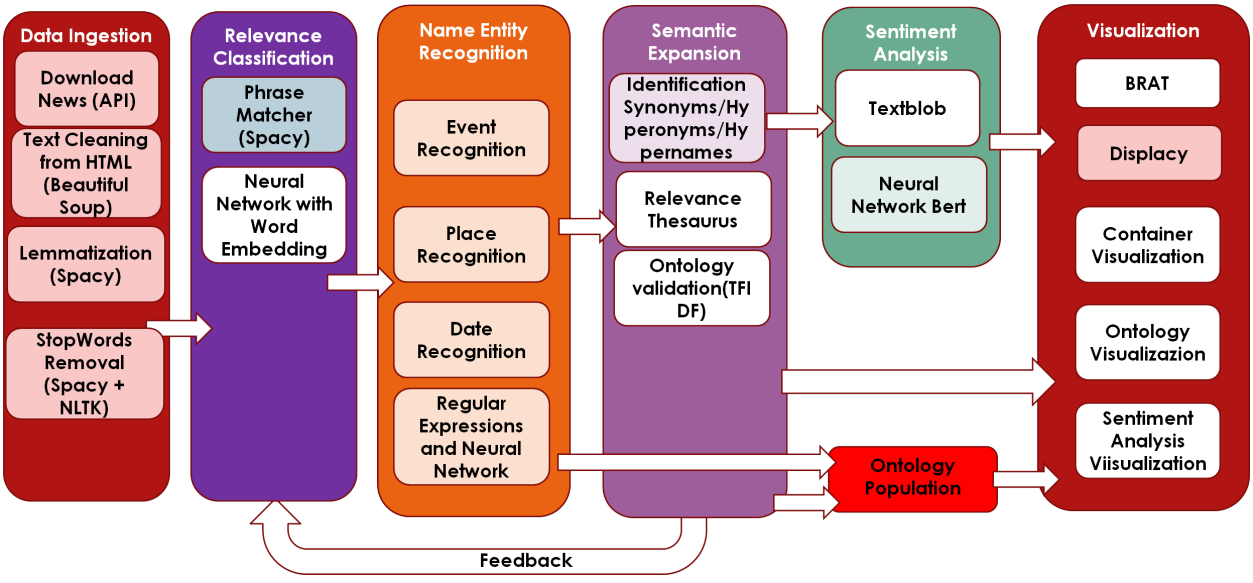


Figure 5.1: Description of the pipeline of all activities to be carried out to complete the various objectives

5.1.2 Architecture

Properly defining a system's architecture is a crucial aspect of its design and, consequently, its efficient operation. It provides an understanding of how different components interact and facilitates the management and maintenance of the system itself. A robust architecture is essential to ensure that the system is scalable, secure, maintainable, and performant over the long term. To clearly and completely describe the structure and behavior of the developed system, we divide the architecture into [156]:

- **Logical Architecture**: by which we outline the data flow, the operating rules, and how information is processed within the system, thus obtaining an overview of the relationships between the different software components.
- **Physical Architecture** focuses on the concrete representation of the system implementation by defining how the logical components are translated into reality.

The distinction between logical and physical architecture allows for a comprehensive system analysis, facilitating design and implementation.

5.1.3 Logical Architecture

The logical architecture capable of satisfying the tasks mentioned in the previous section is structured as follows 5.2:

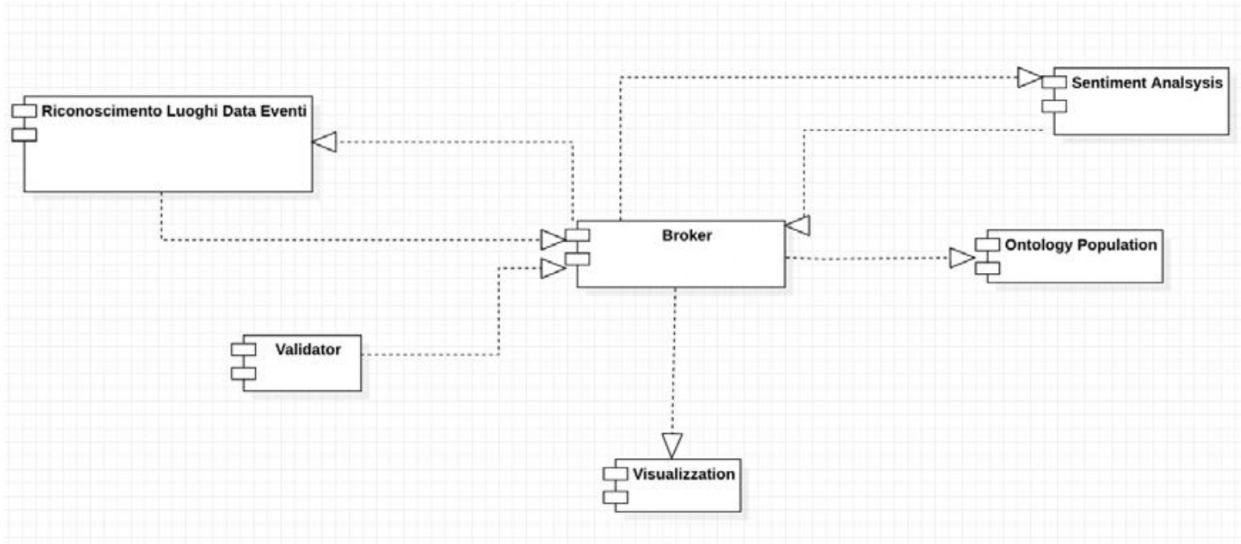


Figure 5.2: Component diagram describing the architecture of the system

The system consists of six main components, which are:

- **Validator**: it collects data from the News API and analyzes it for relevance to the application domain, solving Tasks 1 and 2.
- **Recognizing Places and Dates Events**: it is responsible for extracting information regarding events from texts, identifying places and dates, if any. In this way, it satisfies Task 3.
- **Sentiment Analysis**: this component analyzes the sentiment of the data that satisfied, in whole or in part, Task 3 and performs operations related to Task 5.
- **Ontology Population**: it collects the events extracted from the previous component and populates the ontology, thus satisfying Task 6.
- **Visualization**: this component takes the results produced by the "Sentiment Analysis" and enables the visualization of the results through various graphical interfaces, performing the operations mentioned in Task 7.
- **Broker**: it is responsible for managing the flow of data between the various components, facilitating the exchange of information between them.

.

Finally, it is essential to note that a component was not created for Task 4, as the results obtained were insufficient, and the component would have been redundant for the final project. For what concerns the functioning logic, this was described through the use of a sequence diagram that can be divided into four basic steps Fig 5.3: The process starts with the **Validator**, which has the task of fetching articles from the News API. This component applies a series of relevance filters to select only relevant information concerning the application domain. Once the articles have been evaluated and filtered, the data is sent to the **Broker**, which acts as an intermediary for managing and distributing the information among the various modules of the system. Therefore, the Validator ensures that only relevant data is transferred to the other components. Next, the **Location-Data-Event Recognition** comes into play. This module takes over the articles received from the Broker and analyzes them in depth. Its main goal is to extract specific information about the events mentioned in the articles, such as the locations and dates related to these events. If it can locate such information, it organizes it in a structured format and sends it back to the Broker. At this point, both the **Ontology Population** and **Sentiment Analysis** components retrieve the data from the Broker to perform their operations. The **Ontology Population** uses the extracted event data to populate an ontology, helping to build a structured knowledge base. Meanwhile, the **Sentiment Analysis** analyzes the texts associated with the events to determine the emotional tone or sentiment, trying to determine whether the content is positive, negative, or neutral. Once both components have completed their respective analyses, they send the results to the Broker.

Finally, the **Visualization** component takes the results from the Broker and displays them through graphical interfaces, allowing the user to consult the data intuitively and interactively.

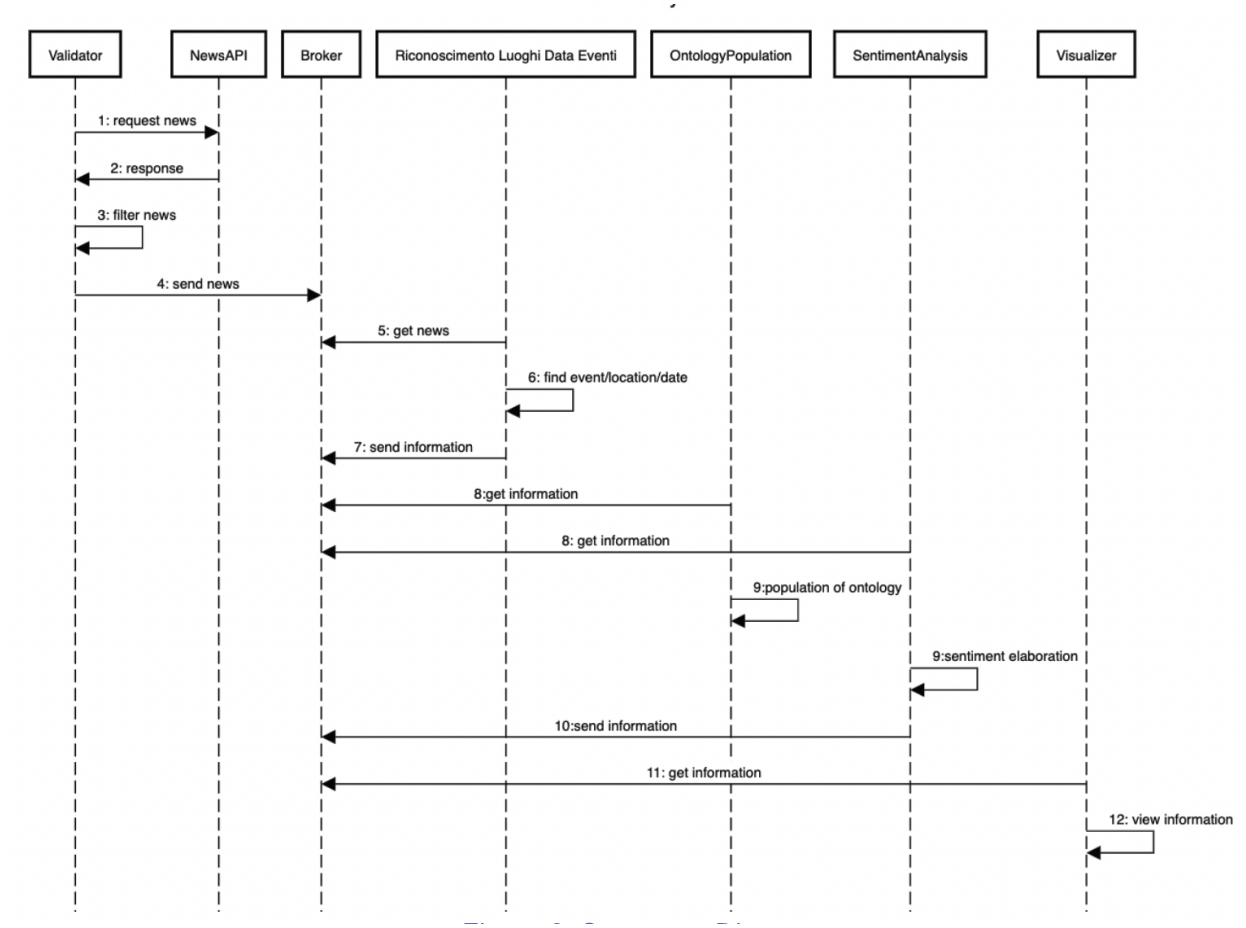


Figure 5.3: Sequence diagram for describing the interaction logic between the various components

The architecture of this project is based on a microservice model, a design methodology that involves the use of independent components, each of which performs a specific process. The main difference from the traditional monolithic architecture lies in the application being broken down into its core functions. Each function, called a service, can be developed, implemented, and managed separately. This allows the services to run independently without affecting the operation of the others in case of problems. In monolithic architecture, however, all processes are interconnected and run as a single block. As a result, making changes or adding new functionality to a monolithic application becomes more complex as it increases overall code. This complexity reduces the ability to experiment and makes it more challenging to introduce new ideas. In addition, monolithic architectures present a greater risk to application availability, as an error in a single process can significantly impact the entire application due to close interdependencies. In contrast, in a microservice architecture, each application process runs as a stand-alone service that can commu-

nicate through well-defined interfaces, usually using lightweight APIs. Each service has a specific function and can be updated, deployed, and scaled independently to meet particular application needs. The main nonfunctional requirements that the microservice architecture must meet are as follows:

- Agility: Development teams work in smaller, more defined contexts, allowing them to operate independently and quickly, thus reducing development time.
- Scalability and Flexibility: Each microservice can scale independently, ensuring the application meets specific functionality needs [157].
- Simplicity and Deployment: With microservices, continuous integration and continuous deployment can be implemented, making introducing new ideas and restoring previous versions easier in case of problems.
- Technology Freedom: Development teams can choose the technologies best suited to solve specific problems, allowing greater flexibility in technical choices.
- Reusable code: Dividing software into well-defined modules allows for greater code reuse.
- Resilience: The independent nature of microservices [158] increases the application's resilience. If a single service fails, the error can be isolated without affecting the operation of the entire application, unlike in a monolithic architecture [159].

5.1.4 Physical Architecture

.

Docker was used to implement this structure, which meets the nonfunctional requirements described above. Docker is a software platform for creating, testing, and deploying applications quickly and efficiently. The software is packaged in standardized units called *containers*, which include everything needed to run the application, such as libraries, system tools, and runtime code. Each microservice runs inside a *container*. The concept of a *container* is similar to that of a virtual machine, but *containers* are much lighter, require fewer CPU resources, and can be started in very little time. This makes them particularly suitable for handling highly time-varying workloads with unpredictable peaks. The data flow of this pipeline will be managed by Apache Kafka, a distributed system consisting of servers and clients communicating via a high-performance TCP network protocol. Kafka follows an architecture based on producer-consumer logic. Some key concepts of Kafka are presented below:

- **Broker:** the *broker* Kafka receives messages from producers and stores them using a unique offset. It also allows consumers to retrieve messages by topic, partition, and offset.
- **Message:** A unit of data in Kafka, similar to a record in a database.
- **Topics and Partitions:** A topic represents a named stream of messages, which can be divided into one or more partitions. Partitions allow Kafka to scale horizontally by distributing data across brokers.
- **Producer:** The entity that sends messages to the Kafka broker.
- **Consumer:** The entity that receives messages from the Kafka broker.

The pipeline will be physically implemented using containers, following the logic depicted in Figure 3. Each task will communicate with adjacent stages via the Kafka broker so that each service will have a specific queue to insert its results.

5.1.5 Methodological definition and implementation task by task of the System

In this section, we will analyze in detail, task by task, all the methodological and implementation aspects related to the activities described earlier in 5.1.1 section. We will delve into the methodologies adopted to achieve the individual objectives and compare the different techniques applicable in other contexts, highlighting the technologies used and the stages of implementation.

Task 1: Data Ingestion

Task 1 aims to collect news from various Italian sources, filtering them to retain only those relevant to the context of Energy Communities. Among the different possible approaches, we chose to use the public News API for collecting specific news, as this method offers the necessary flexibility. The module developed in Python uses the API to download news based on predefined keywords, such as “energy communities” or “collective energy.” The API is invoked 18 times per keyword, three times per day, staying well below the limit of the free version of the API (100 calls per day). A typical call retrieves a JSON with the URL, title, and a short description of the news. The complete news content is downloaded using the urllib library and cleaned up with BeautifulSoup to remove unnecessary HTML tags and Javascript. Finally, the cleaned text is passed to a second module that analyzes the relevance of the news to the Energy Communities domain.

Task 2: Relevance Validation

Relevance validation refers to a binary classification in which a text is evaluated as "Relevant" or "Not Relevant." Only relevant texts are transferred to the next steps of the process via a queue managed by Kafka [160], while irrelevant texts are discarded. Two main approaches were considered. The first is based on existing Natural Language Processing (NLP) techniques in predefined libraries, which analyze text content to determine its relevance. The second approach, more related to Machine Learning, uses binary classifiers such as Decision Tree, Support Vector Machine or neural networks. Given the data's textual nature, using neural networks in combination with Word Embedding techniques [161] is particularly effective, although it requires preliminary text transformation operations. Both approaches have advantages and disadvantages. The NLP method requires no prior training and can be used immediately if text features, such as keywords, are defined. This approach is simple and effective but should be monitored and evaluated over time. On the other hand, the Machine Learning-based method requires a corpus of texts to train the classifier, which results in greater use of computational resources. However, Word Embedding makes it possible to create clusters of relevant terms without relying solely on predefined keywords. The Spacy library offers two main advantages [162]: it does not require training an algorithm, and it uses a set of keywords to obtain accurate results without large computational resources. However, the approach is limited by the use of predefined keywords, although this can be partially overcome through semantic expansion. Other methods of avoiding using keywords alone have been explored, such as creating a semantic network based on Word Embedding, which can analyze a corpus of documents classified as relevant or irrelevant. To properly train the neural network, a set of about 2,500 relevant texts was used alongside about 500 texts labeled as irrelevant, obtained from external sources with keywords disconnected from the topic of Energy Communities. To perform classification using NLP, the developed module uses the "Spacy" library and its natural language processing capabilities to analyze the text in detail, removing punctuation marks and stopwords and applying lemmatization to vocabulary. Spacy, in particular, can use the stopwords of the Italian language provided by another library, "NLTK" [163], which provides a predefined set of such words. After configuring the system to perform lemmatization with the Italian vocabulary and defining the appropriate stopwords for the text to be analyzed, Spacy allows for the identification of specific keywords within the text itself. In particular, the "PhraseMatcher" module will enable you to perform this search, working on one or more sets of words for which Spacy performs the same lemmatization operations previously applied to the text and verifying their presence. It is possible to define several sets of words, known as "Patterns" in the Spacy library, and the Matcher locates these words in the text, reporting which set they belong to and providing the initial and final positions of the detected word in the text. This approach, while simple, proves to be very effective, and its operation has been verified by tests to assess its accuracy. On the other hand, as

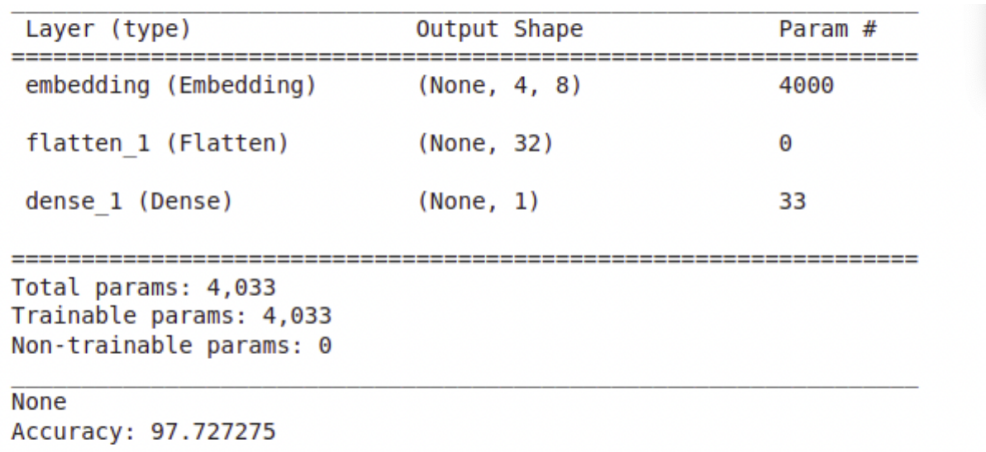


Figure 5.4: Scheme of Neural Network for Word Embedding

far as the use of neural networks is concerned, these were defined to perform the classification, which was constructed using a sequential model, to which were added an Embedding layer, a Flattening layer, and a final Dense (fully connected) layer, with a Sigmoidal activation function. The optimizer exploited for the network is the classical Adam, which has good convergence speed and strong stability. The network described in Figure 5.4 has about 4000 free parameters and, during training, which lasted for 50 epochs, achieved an estimated accuracy of 97%. These results, though preliminary, are promising, but they need further validation on other texts before they can replace the Spacy-based system. One advantage of using the neural network is breaking free from a fixed set of words. However, a significant disadvantage is the demand for more computing power, which would make the containers more resource-heavy. Nevertheless, using the Word Embedding technique, some interesting considerations can be drawn by looking at word clusters projected in a three-dimensional space.

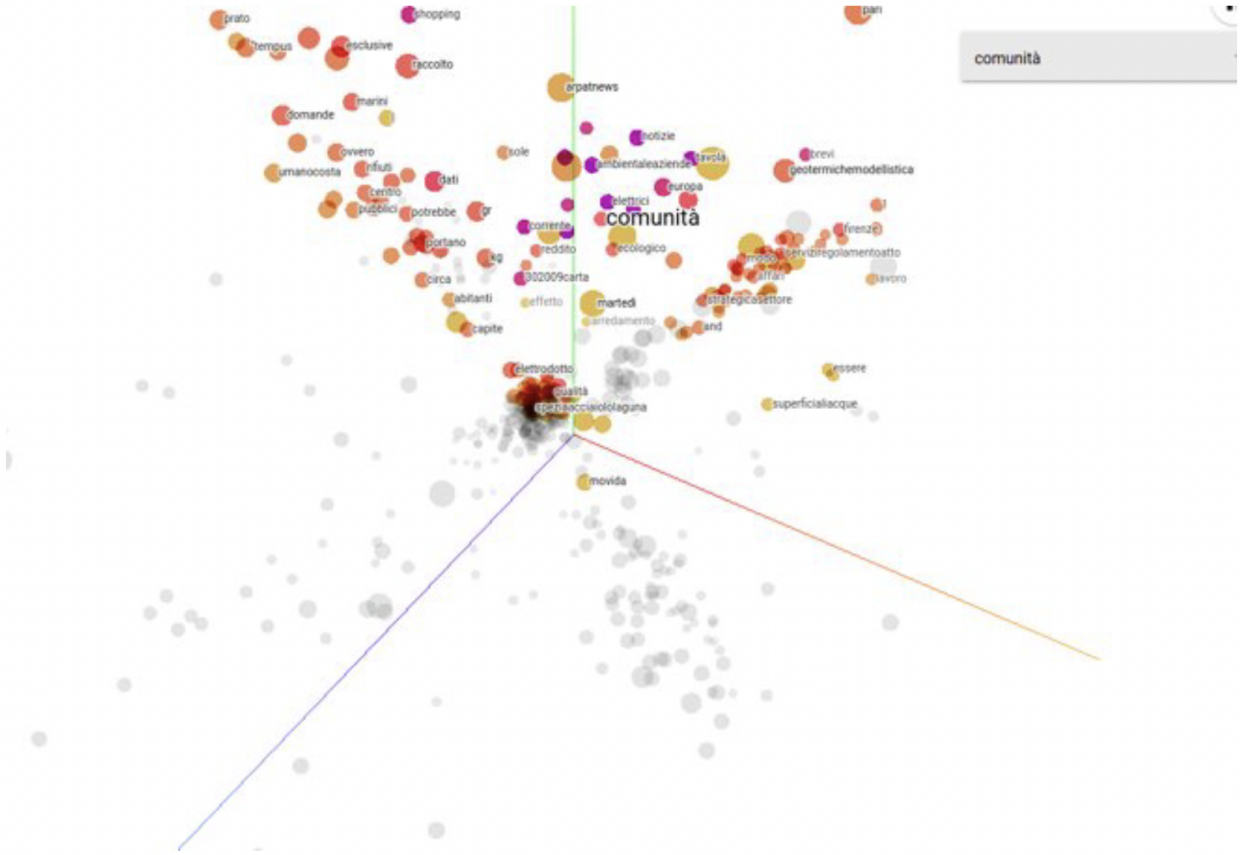


Figure 5.5: Clustering of embedding using Principal Component Analysis

In particular, from Figure 5.5, where clustering was performed by applying a Principal Component Analysis (PCA) algorithm [164], the formation of groups of terms that are very close to each other emerges, suggesting the predominance of some components over others.

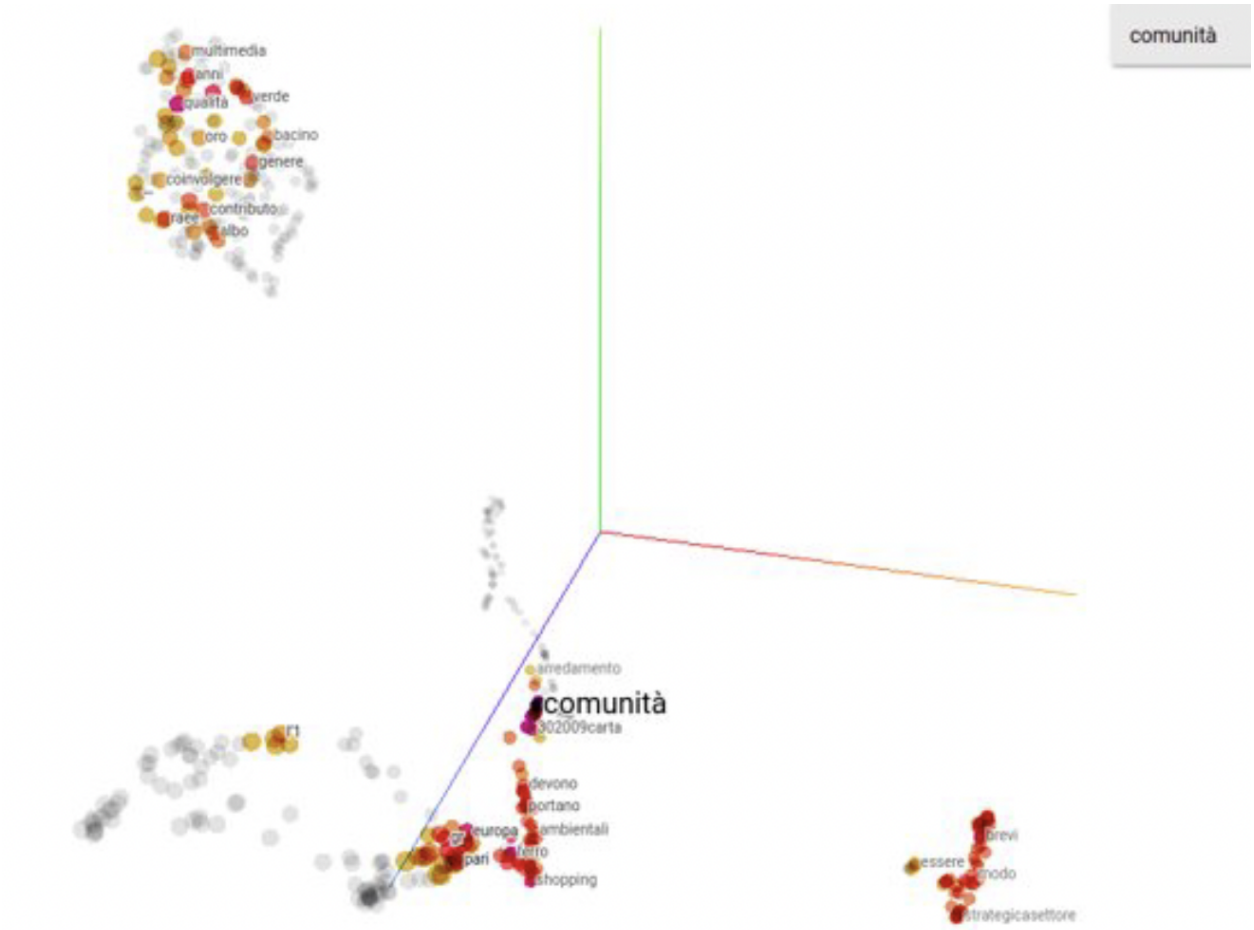


Figure 5.6: Cluster of Embedding using t-distributed stochastic neighbor embedding

In contrast, in Figure 5.6, the clusters appear rather scattered and ill-defined. However, looking at the same representation obtained with the TSNE technique [165], as shown in Figure 5, it is evident that there are well-defined and spaced clusters that characterize the corpus of documents analyzed. Given the relatively small number of texts examined, especially regarding Non-Relevance, there is a margin for improvement in the results obtained with Word Embedding to obtain more meaningful and representative clusters.

Task 3: Recognizing Places/Dates/Events

The objective in this part of the work was to search for each of the texts taken as input, whether there were references within it to specific events (such as workshops, conferences, or meetings), and whether these were associated with places and/or dates. There are three types of recognition: Events, Places, and Dates. After doing this, the next step is the relationship between Events/-Date/Location and, finally, a final subtask relative to the analysis and selection of the results.

Throughout this process, the main libraries used to implement the different scripts were SPACY and NLTK. Therefore, all the subtasks will be listed below:

1. **Event Search**:The first step was to search for the desired events, entered manually into a vector by the user. This was accomplished through WordNet synsets from the NLTK library; next, all identified synsets were entered into a vector used to search for events in the text using the "PhraseMatcher" command from the Spacy library [166]. With each match, the text and the location within the document were retrieved and saved in a second vector.

2. **Research Places**: then, the search was extended to places, understood in two ways: geographical locations and online platforms. Different approaches were used for each of the two cases. Regarding online locations again with the use of the PhraseMatcher keyword search was used; in this specific case, with all terms from all digital platforms loc-online = ["online," "remote," "Teams," "Zoom," "Skype," "video conferencing,"...]. Regarding the identification of geographic locations, two different approaches were taken. Initially, the implementation of open-source libraries, such as GeoText, which analyzes an input text to identify and save locations in a vector, was tried. However, this method did not produce the desired results. The second, more structured approach involved using Named Entity Recognition (NER) technology provided by Spacy, which offered significantly better results, which is why it was chosen.

3. **Search Dates** Some difficulties arose in the search for dates. Initially, the NER methodology was used with Spacy, but it correctly recognized the DATA entity only in English and not in Italian. Subsequent attempts with libraries such as defender and date parser produced inaccurate results. An attempt was made to translate the text into English and apply NER, but the available translation libraries (such as googletrans and deep_translator) handled only a limited number of texts. After these difficulties, the solutions adopted were to split the text into words, partially translate it using English synonyms, recompose to allow NER to be applied in English, and use RegEx regular expressions to recognize dates in standard formats [167], with better results than the previous methods. Finally, the regular expressions method was used to test various matches.

4. **Relation: place, date, event**: research on detecting relations between event and place was carried out using token distance and patterns. In the first approach, we analyze the location of the event token and check, in a predefined interval, for the presence of a date or place, then extend the search to combine the three elements (event, place, date). In the second approach, with the Spacy library, patterns associate verbs with events, places, and dates,

exploiting attributes such as ORTH for events and ENT-TYPE for places and dates. Tests were conducted to evaluate precision, recall, and accuracy.

5. **Analysis and Selection of Results** At this point, we have developed a tool that can analyze a text and return the following data: Physical Locations identified (1 type of search), Online Locations identified (1 type of search), Dates identified (3 types of search), Events identified (1 type of search), Event/Date Relationships identified (2 types of search), Event/Location Relationships identified (2 types of search), Event/Location/Date Relationships identified (2 types of search), Event/Location Online Relationships identified (2 types of search), Event/Location Online/Date Relationships identified (2 types of search). From observing these results, we can draw three important conclusions: regular expressions return the best results for the date, and the token distance method returns better results than patterns. In almost all results, the Recall returns acceptable values. We cannot say the same for precision. The first two points highlight the need to use the appropriate functions for better results. The third point, however, highlights the problem that all functions are executed in the `main` without a clear structure, making the results' analysis less efficient. However, considering our goal, it is not necessary to obtain many high-precision results to analyze later; instead, we are interested in obtaining a single result that identifies events potentially related to specific dates, locations, or both. The last implementation, therefore, focused on identifying a logic that would not return all the events, dates, and places found but only the most likely ones. After observing the test results, it was necessary to determine a way to call the most effective functions to get the best results. The logic for calling functions was structured as follows:

   (a) A search is conducted within the text to locate the presence of a desired event (e.g., workshop, conference, convention, meeting).

   (b) If the event is found, you proceed to search for a match for both the date and an "online location."

   (c) At this point, four possible scenarios can occur:

      i. **Match Event/Online Place Found, Match Event/Date Found**: whether an Event/Online Place/Date triple exists is checked. If positive, the triple is returned; if negative, two doubles are returned.

      ii. **Match Event/Location Online Not Found, Match Event/Date Found**: an Event/Location match is searched, and two possibilities can occur:

         - **Match Event/Date Found, Match Event/Location Found**: a possible Event/Date/Location triple is searched. If positive, the triple is returned; if not, two doubles are returned.

- **Match Event/Date Found, Match Event/Location Not Found**: the Event/-Date match is returned.

iii. **Match Event/Location Online Found, Match Event/Date Not Found**: the Event/Location Online match is returned.

iv. **Match Event/Location Online Not Found, Match Event/Date Not Found**: an Event/Location match is returned. If positive, a double is returned; otherwise, nothing is returned.

The final result (`final_match`) was tested on about 100 texts, obtaining the following results:

- Accuracy = 71%.
- Recall = 83%.
- Accuracy = 82%.

Task 4: Semantic Expansion

This task involves the analysis of terms in the ontology and texts in the corpus related to the topic of energy communities to validate the ARAKNE ontology. Two approaches were used: comparison of synonyms, hypernyms, hyponyms, and TF-IDF averaging. In the first approach, terms in the ontology were collected with Owlready2, and for each, synonyms, hyponyms, and hypernyms were generated. However, this methodology did not bring significant benefits, as the derived terms deviated too much from the original meaning. The second approach focused on calculating the TF-IDF [168], which means assessing the inherency of ontology terms concerning the corpus. After constructing a dictionary of the words in the corpus and calculating the IDF of each term, the average IDF was compared with that of the ontology terms. If the IDF value of a term was higher than the average, the ontology was considered valid. However, this part is only a hint; we will not dwell on it further since it represents another part of the overall work.

Task 5: Sentiment Analysis

After prewashing the texts appropriately described by the pipeline above, in this phase, we will focus on applying sentiment analysis to identify and extract opinions about your posts, associating a positive, negative, or neutral sentiment. To do this, two different approaches were applied:

- The first approach was to use the TextBlob library [169] to compute the polarity of articles quickly and easily; however, due to poor compatibility with the Italian language, this library returned not very reliable results from preliminary tests

- As a second approach, we chose to use a neural network based on the BERT (Bidirectional et al. from Transformers) [170] model, made available by the "hugging face. co" platform. Also, another detail to highlight is that to receive more precision, the sentiment was not applied to the whole text but to the surroundings of the identified keyword to increase accuracy in measuring the sentiment using the PhraseMatcher. The workflow of this task is as follows: after declaring a model and a tokenizer, the function "Sentiment_score" was called, which tokenizes the text under analysis by generating a tensor of type 'torch', a multidimensional matrix containing elements of a single data type. The result can be fed to the neural network, which returns a value subsequently normalized from 1 to 5. Finally, the associations were as follows: values 1 and 2 were associated with a negative sentiment, values 3 to identify a neutral sentiment, and values 4 and 5 a positive sentiment.

Task 6: Population of Ontology

"Ontology Population" refers to inserting instances and properties within an ontology. Here, the purpose is to enrich an ontology with relevant instances previously extracted. In this regard, two different approaches for populating the ontology were tested:

1. **Populating the Ontology by Word Embedding Techniques**: the ARAKNE ontology is populated using concepts extracted from the document corpus. Through appropriate *Text Similarity* metrics, entities closest to the ontology's classes are identified.

2. **Population of Ontology-based on Recognized Entities via NER**: the results obtained from the application of *Named Entity Recognition* (NER), described in the section on Recognition of Places, Dates, and Events, are used to populate an ad hoc created mini-ontology.

The first approach used Word Embedding techniques to analyze a documentary corpus of 2161 texts related to energy communities. The goal was to identify clusters of concepts related to the classes in the ARAKNE ontology and populate them with the most relevant concepts. The Word Embedding model was created using FastText embedded in Gensim, with preliminary text preprocessing operations: cleaning HTML, lemmatization, conversion to lowercase, and removal of stopwords and punctuation. After training the model, which produced 491,431 vectors, cosine distance was used to find the words closest to specific concepts. Despite the attempt, the clusters obtained were too general and irrelevant enough to populate the ontology, thus leading to the abandonment of this method Fig 5.7.

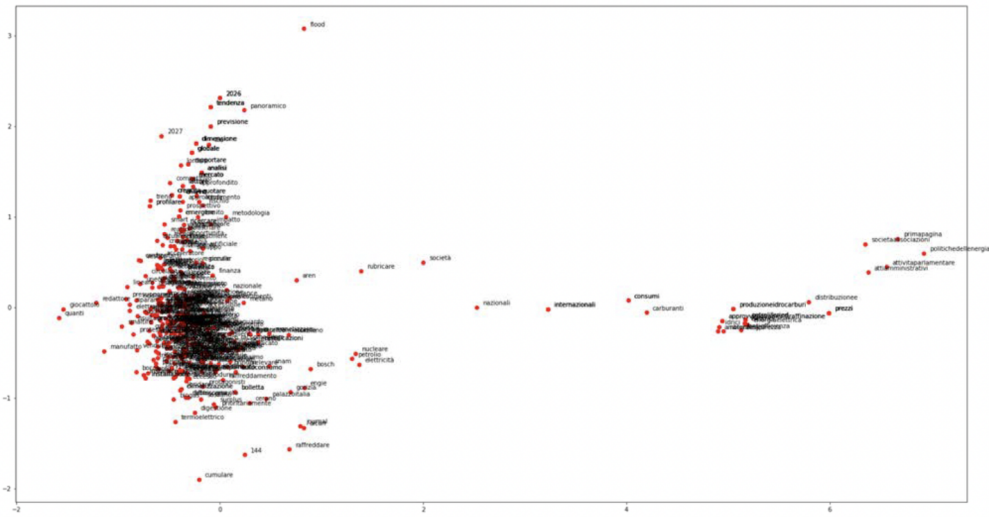


Figure 5.7: Representation in Vector Space of the 46 clusters identified

The second approach, on the other hand, focused on identifying entities in the text to create a mini-ontology. Using Named Entity Recognition (NER) techniques, three classes of entities were identified: dates, places, and event types. Next, a module was developed in Python using the Owlready2 library to create and populate the mini-ontology. Attributes were added to the entities, such as start and end indexes in the texts, allowing more precise placement of the entities in the textual context. This approach was most successful, allowing the mini-ontology to be populated with 1345 individuals. While Word Embedding offered too general results, the entity identification approach proved more effective for ontology populating.

Task 7: Visualization

The last part of the pipeline is to analyze strategies for visualizing entities within the text. From this point of view, three different approaches were studied:

1. The first solution was to use the Displacy module of the Spacy library in Python. This tool accepts an object of type “Doc,” generated by processing text using a Spacy model, and displays entities recognized by the NER using colored labels. However, in this particular case, the entities were identified earlier, and there is no NER Spacy model in this step. Nevertheless, it was easy to create an empty Spacy model (Blank) and populate it with objects of type “Span,” one for each of the entities recognized in the previous step. An example of using Displacy is shown in Fig 5.8, which illustrates the display of entities within a text with Id_Alert 8.

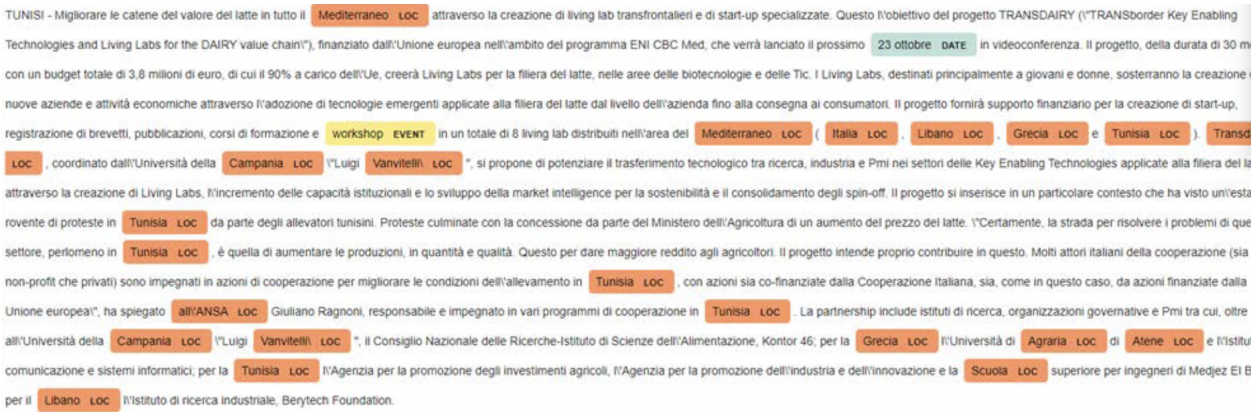


Figure 5.8: Example of Displaying Entities in a Text with Spacy's Module c

A significant limitation of Spacy is "overlapping entities," which essentially does not allow multiple annotations to be given to the same token, increasing the risk of errors. The advantages certainly include the speed with which the visualization can be generated. In addition, it is easy to integrate this visualization into a Colab or Jupyter notebook by enabling the Jupyter=true option in the render() method of Displacy. It is important to say that several open-source libraries based on Spacy's Displacy module have been made available to anyone. The most successful one is the Dispacy-ent JavaScript library, which provides APIs that allow a Web programmer to graphically render an analysis generated by Spacy services from any text in any language recognized by Spacy. In Fig 5.9, its use is shown using the same text as before as an example:

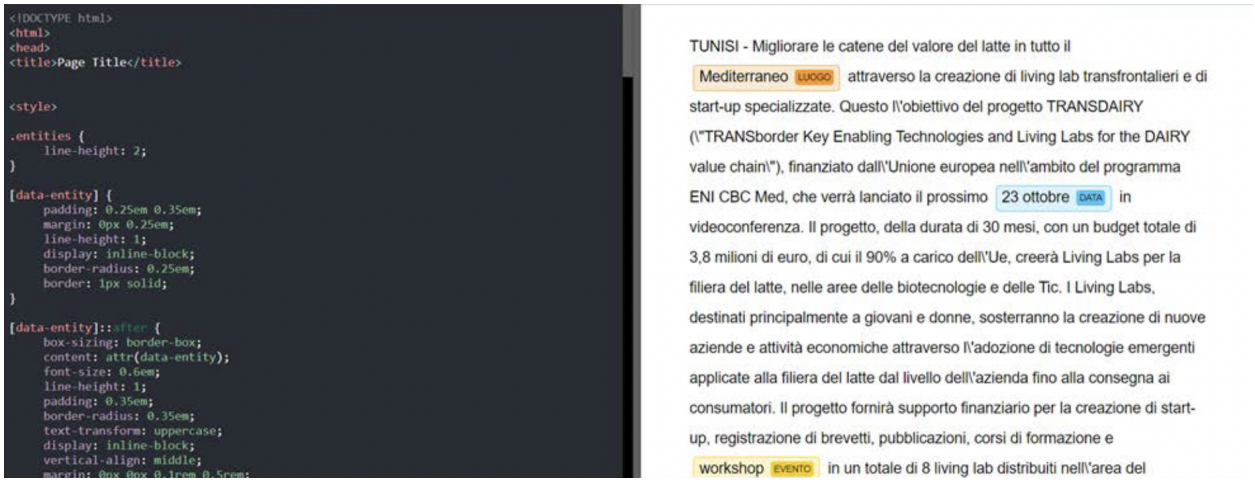


Figure 5.9: Example of Displaying Entities in a Text with the Displacy-ent Module

In addition, there are online versions of Fig 5.10, which make these visualization APIs available, as well as pre-trained NER templates for different languages, which can be used to try

this visualization.

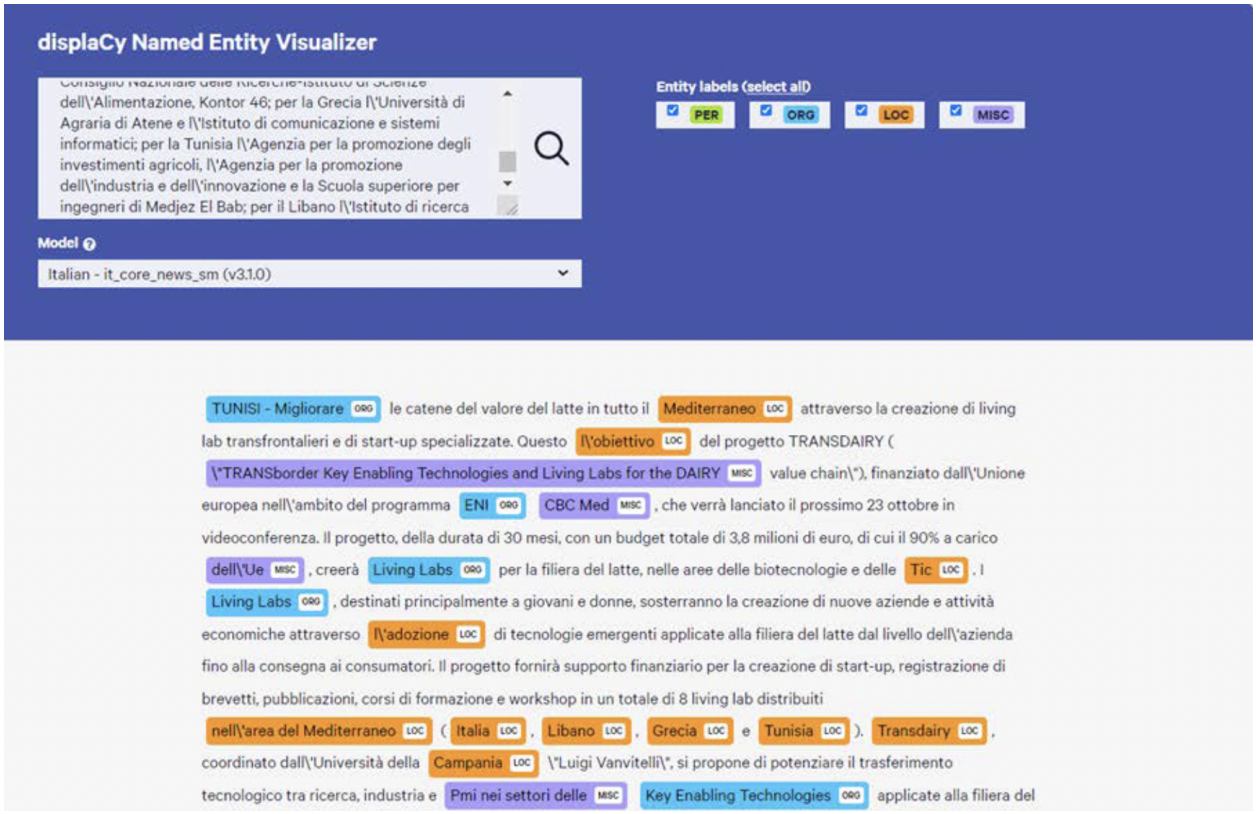


Figure 5.10: NER online made available by Displacy-ent.

As can also be seen from the figures shown above, there are very few labels to label in the Italian language. So, while simple and intuitive to use for the Italian language, it is not the best choice.

2. An alternative approach was to use Brat, a web-based text annotation tool available both online and in an installable version. The project is open source, and an example of its use is shown in Fig 5.11.

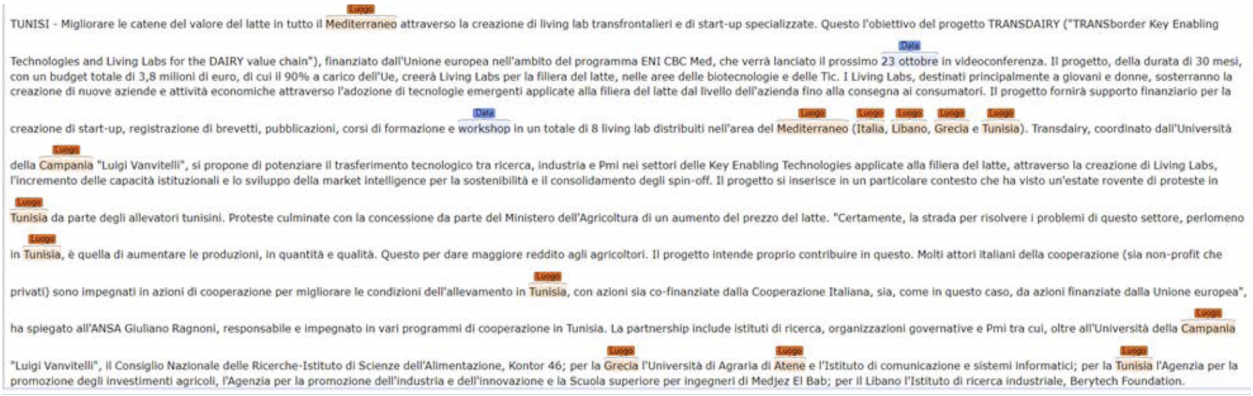


Figure 5.11: Example of Entity Display in a Text with Brat style

Annotation labels are inserted above the word they refer, delimiting them using a curly bracket. Such visualization is made possible through the use of a file containing NER results strutted in a standard format called CoNLL-U, which contains the following fields:

- ID Progressive of the annotation;
- Name of the annotated entity;
- Beginning and ending index of the entity in the text;
- Label used to annotate the entity;

this visualization was generated from a parser that, from CoNLL-U format [171], builds the HTML, which in turn displays the text with the entities appropriately tagged through a brat-like visualization.

3. The Last type of experimental visualization is the geographic visualization, which can give the user a general overview of all events found by placing them within a map by associating each event/place/date relationship with a marker, positioned on the map by the coordinates associated with the location of the triple. The color of the marker can be red and blue, depending on whether the sentiment was positive or negative. Clicking on the Individual Marker first shows a preview and then redirects to a page with all the information about the individual event, specifically: Title, Description, Relationships Detected, Sentiment Analysis Fig. 5.13.

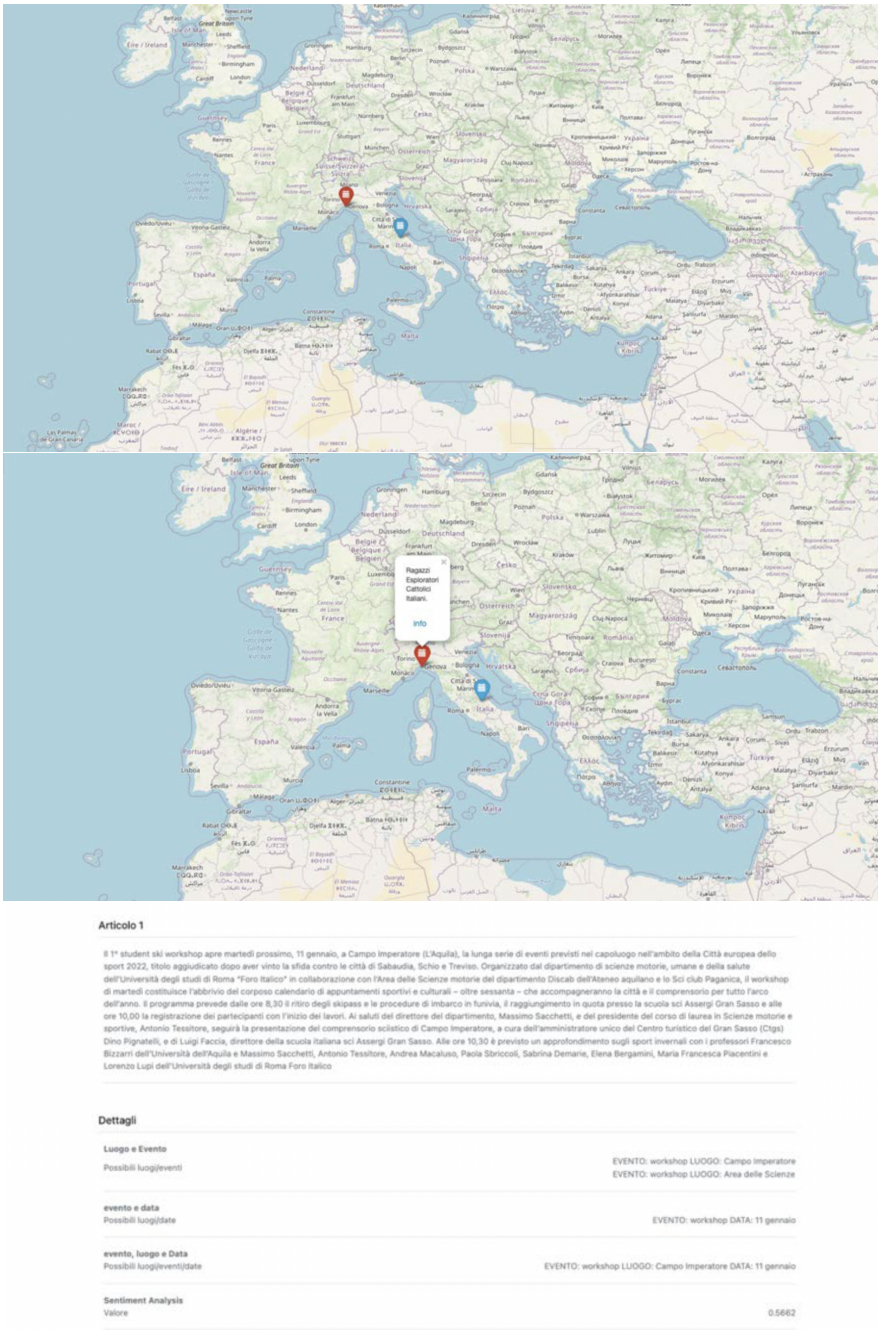


Figure 5.12: The three images show the different intefaces of the single map implementation

After doing so, this part was then integrated within an application that would allow filtering of the results obtained from the pipeline according to 3 different modes, which are the possibility of selecting events from a given month and/or a given year. It is possible to choose the type of event (workshop, conference, etc...) according to the kind of place where it is held (online, presence). The filtered results can then be viewed on the map or through a list.

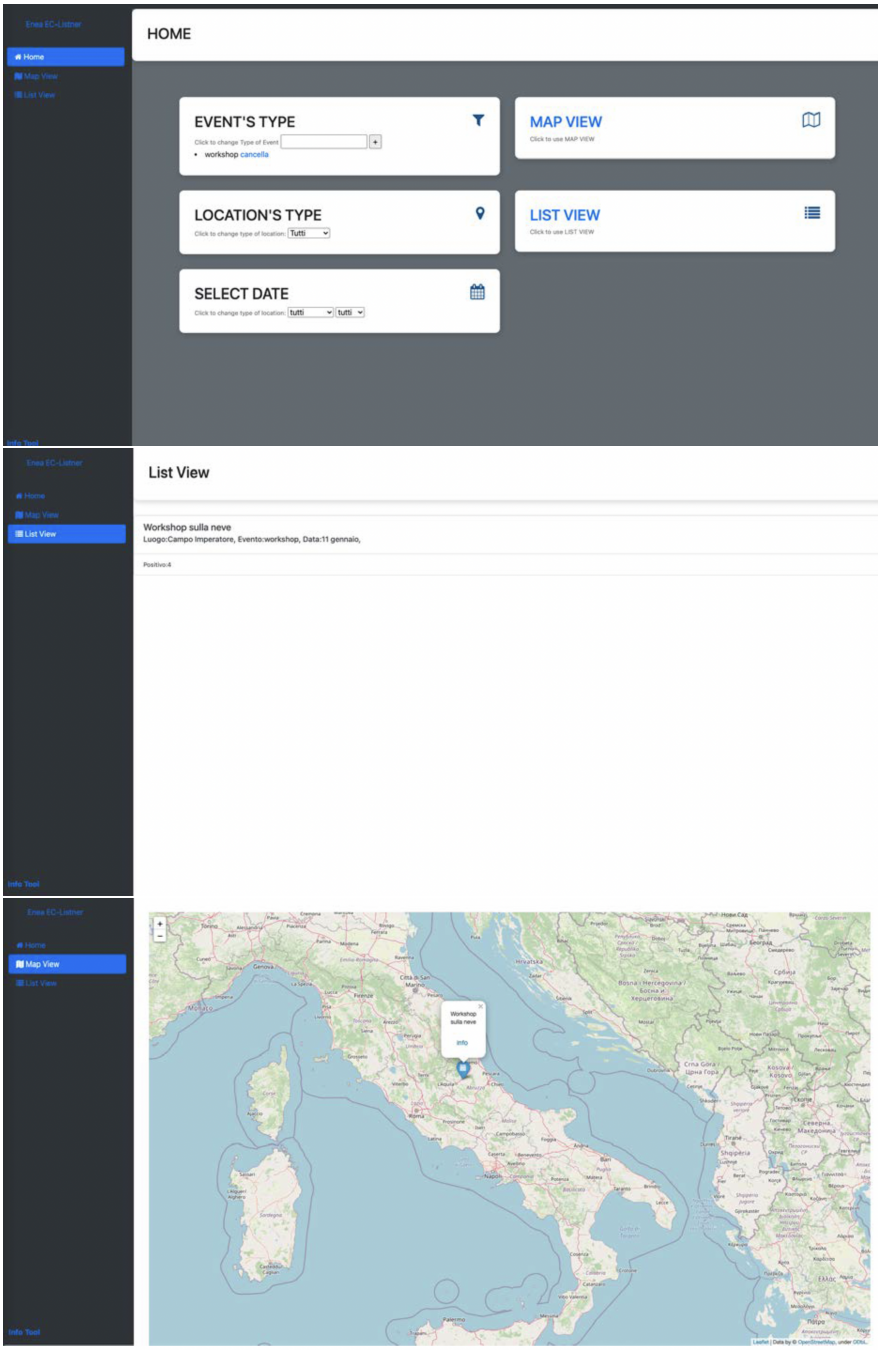


Figure 5.13: The three images show the different interfaces of the ad hoc visualization tool

5.1.6 Results and Conclusions

The experimental results section describes the quantitative evaluations of two critical tasks: Relevance Validation and Place/Date/Event Recognition. A Spacy-based classifier was applied to a corpus of 2348 texts related to Energy Communities. The results show 100% accuracy, 96% recall, and 96% overall accuracy. The system was evaluated as sufficiently accurate for the problem,

showing that the classifier can effectively distinguish between relevant and non-domain texts. Various tests were conducted to evaluate the system's ability to recognize places, dates, and events in texts. The results varied depending on the technique applied: Event Recognition: 89% accuracy, 100% recall, 96% accuracy. Place Recognition: varying accuracy (56% offline, 94% online), accuracy between 56% and 98%. Date Recognition: lower results, with accuracy ranging between 18% and 62% depending on the method. Combined Event/Location/Date Search: accuracy up to 97% in online cases. The project developed a Big Data pipeline for analyzing news and other online sources, integrating Natural Language Processing (NLP) and Machine Learning techniques. The microservices architecture built with containers offers scalability and flexibility, enabling rapid integration and replacement of software components. However, there are areas for improvement, especially in the area of date recognition, and there is a need for further testing and comparison with other techniques. Word embedding is proposed to improve binary classification of relevance in Ontology Population and future implementations to expand event search and improve data visualization capabilities.

5.2 Military Supply Chain on Federated Cloud Continuum

After exploring the main algorithms of machine learning with the work set forth above, the next step was to delve into the topic of main cloud patterns. To achieve this, a military case study was selected, focusing on defining a federated supply chain. Based on this, a local architecture was first described and then mapped using cloud-agnostic patterns to enable future deployment. This work was published at the IEEE Technologies for Defense and Security conference. Therefore, we will discuss below what was done using part of the published paper [152].

5.2.1 Supply chain: an overview and in-depth analysis of military operations

Starting with the definition of a supply chain [172], we can say that this is a group of entities involved in a process chain concerning the acquisition of raw materials or components, their conversion or assembly into a product, and the distribution of the final product to a customer Fig. 5.14.

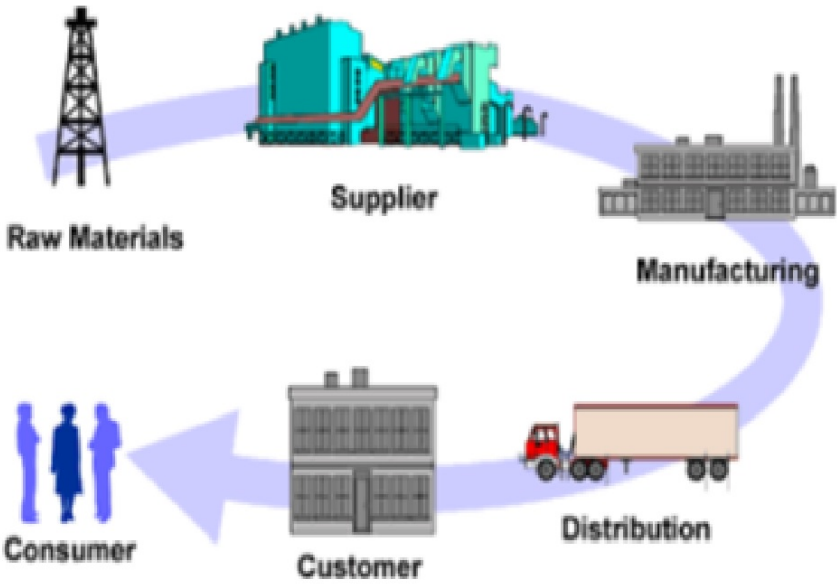


Figure 5.14: Typical supply chain

Supply chain optimization is crucial for business development, involving all stages, from production to final sale. Today, no company alone can completely meet customer needs, especially in the context of rapid technological changes and increasingly shorter product life cycles. To meet these challenges, companies have begun collaborating in an "extended enterprise, " allowing them to share skills and resources to improve efficiency and gain a competitive advantage. Supply chain integration allows for greater flexibility and cost reduction, which is crucial to addressing market demands. Companies focus on their core business, outsourcing other functions to supply chain partners. The real competition today is between entire supply chains rather than individual companies. The fundamental flows in the supply chain are materials, information, and cash, and they must be coordinated to ensure effective service to the end customer.

5.2.2 Supply Chain Proposal

The traditional, linear representation of the supply chain is now considered too simplistic, as materials and information flow through a complex network of organizations. A "supply network" [173] includes interconnected entities that transform resources to deliver goods and services to end markets and is generally viewed from the perspective of a central company. However, this view ignores interdependencies between suppliers and buyers, which can extend beyond traditional connections, involving relationships with external entities such as governments and NGOs. Research on industrial networks underscores the complexity of these relationships and the importance of examining the entire network to understand the dynamics of supply chain node disruption and

the sustainability of practices. Managing supply networks poses more significant challenges than traditional supply chain management, as leading companies must address the complexity of information, the coding and transmission of that information, and the ability of organizations to meet transaction requirements. To address these challenges, a military supply chain simulation model was developed using the "Stella Architect" software Fig. 5.15.

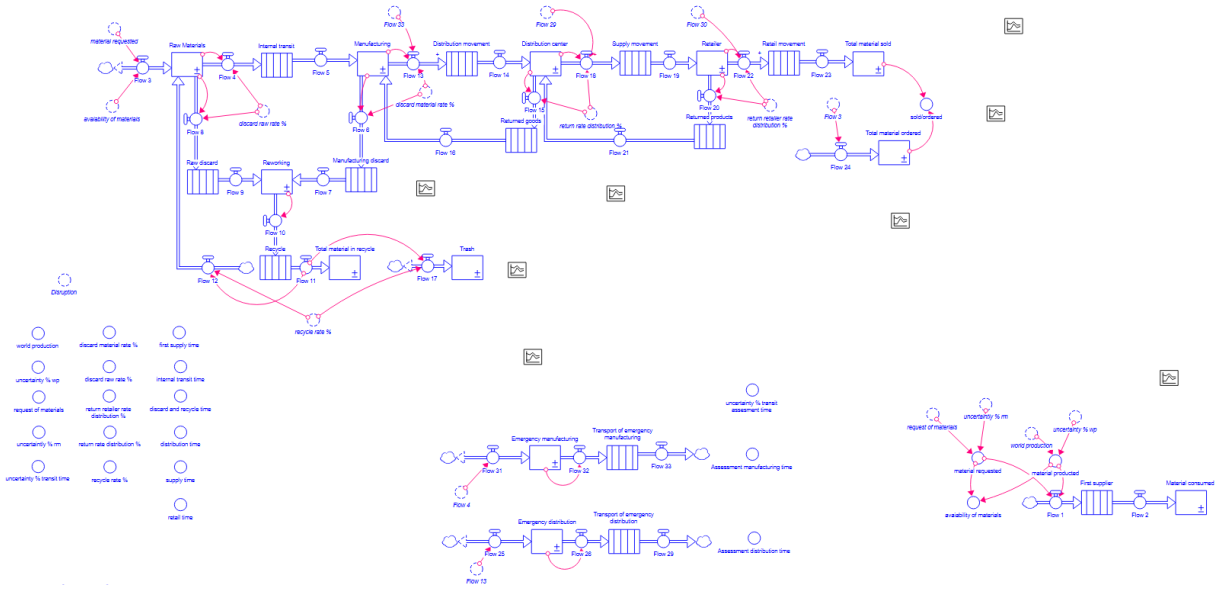


Figure 5.15: Starting model made with the Stella Architect tool

5.2.3 Proposal Architecture

Before delving into the system architecture design, dwelling on a crucial point of the proposed supply chain is necessary. Namely, the introduction of the collaborative approach ensures collaboration among multiple chains in a scalable manner to meet an improvised increase in market demand. The proposed architecture in Fig. 5.16

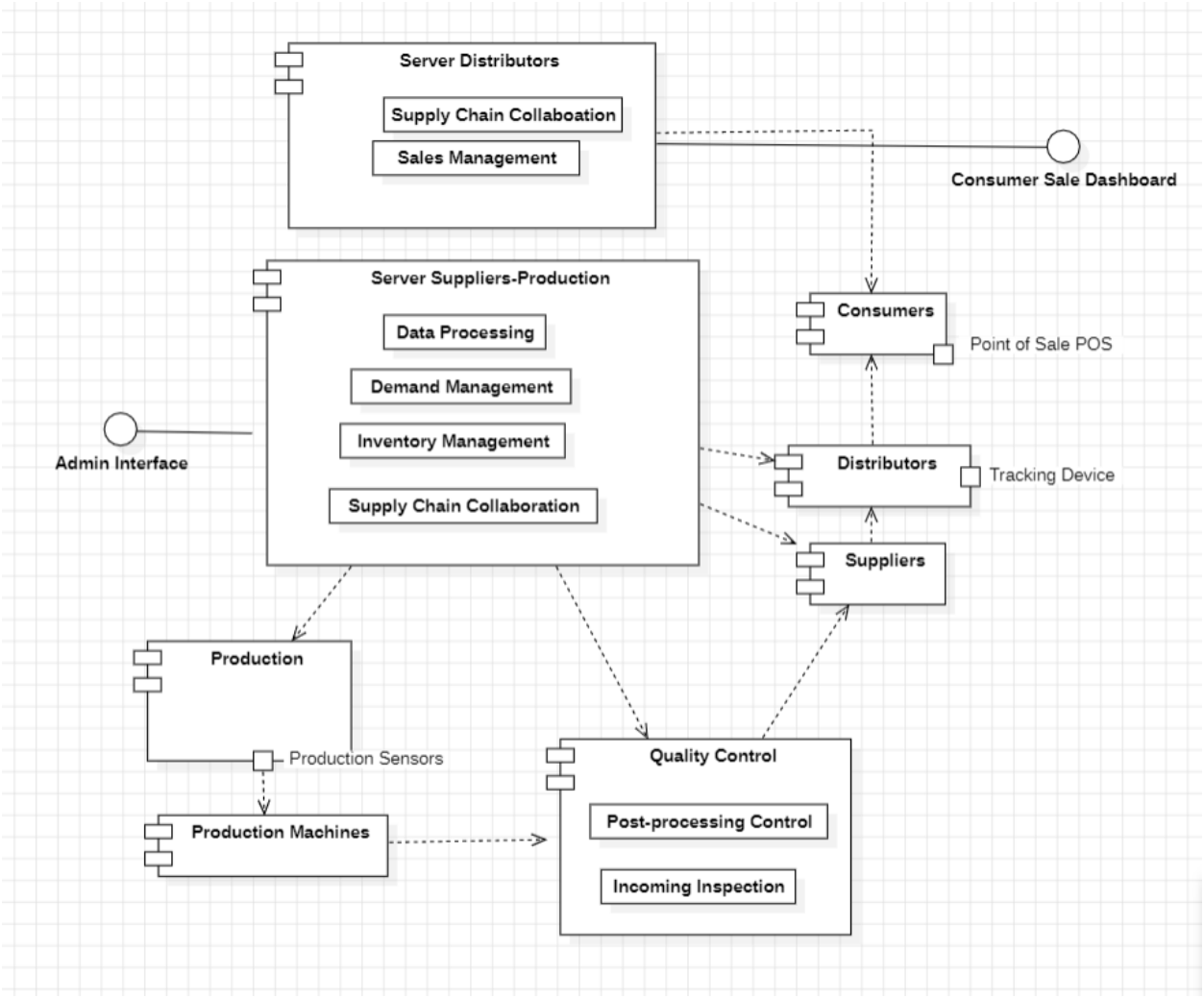


Figure 5.16: Component diagram based on supply chain architecture

At this stage, the focus is on understanding why and how all elements have been included in the supply chain. The architecture starts with the production process, which is based on global market demand. To monitor and manage this process in real-time, an element called “Production Sensors” was introduced within the “Production” module. These sensors, which are part of the Internet of Things (IoT), provide immediate updates on the status of machines, allowing rapid intervention in case of anomalies. The “Product Machine” component represents the production process. Once the product is produced, it moves to the next step: Quality control. This step ensures that each product meets the required standards before continuing. For this purpose, a module called “Quality Control” was created, which includes the subcomponents “Input Control” and “Post-Processing Control,” edge devices that check products both before and after processing. If problems are found, products are reintroduced into the process. After passing quality control, products are ready for the Product Distribution stage. Here, they are transferred to suppliers and

distributors through two specific components called “Suppliers” and “Distributors.” To monitor the quantity of products at this stage, an edge device called “Tracking Devices” was added within the “Distributors” module. This device provides real-time data, ensuring that products are always available where demand is high and that there is no excess where demand is low. Products are now ready for sale to consumers. To manage sales and monitor consumer needs in real-time, an edge component called “Point of Sale” was introduced within the “Consumers” module. This tool collects immediate information on consumer trends and communicates it to the server. Finally, two additional components are needed to ensure centralized management and analysis of the data collected from the various edge devices. The first, “Supplier-Producer Server,” includes functions such as “Data Processing,” “Demand Management,” and “Inventory Management,” which work together to analyze, forecast, and respond to market needs. The second, “Consumer Server,” is a central platform for managing the storage and sale of products to end customers. It is separate from the “Supplier-Producer Server” for security issues. This server centralizes data to facilitate real-time distribution. Both cloud servers offer interfaces to monitor data and include a module called “Supply Chain Collaboration,” which facilitates communication and exchange of resources between different supply chains, thus supporting a federated architecture.

5.2.4 Porting to the cloud with Agnostic Vendor Pattern

Based on the architecture just implemented, the next goal was to map the components for porting the entire infrastructure to a Cloud Continuum environment from the edge to the core. To achieve this goal in an agnostic form, we will use cloud computing patterns by mapping them in a manner consistent with the proposed architecture. Below in Fig.5.17 is therefore shown the mapping done.

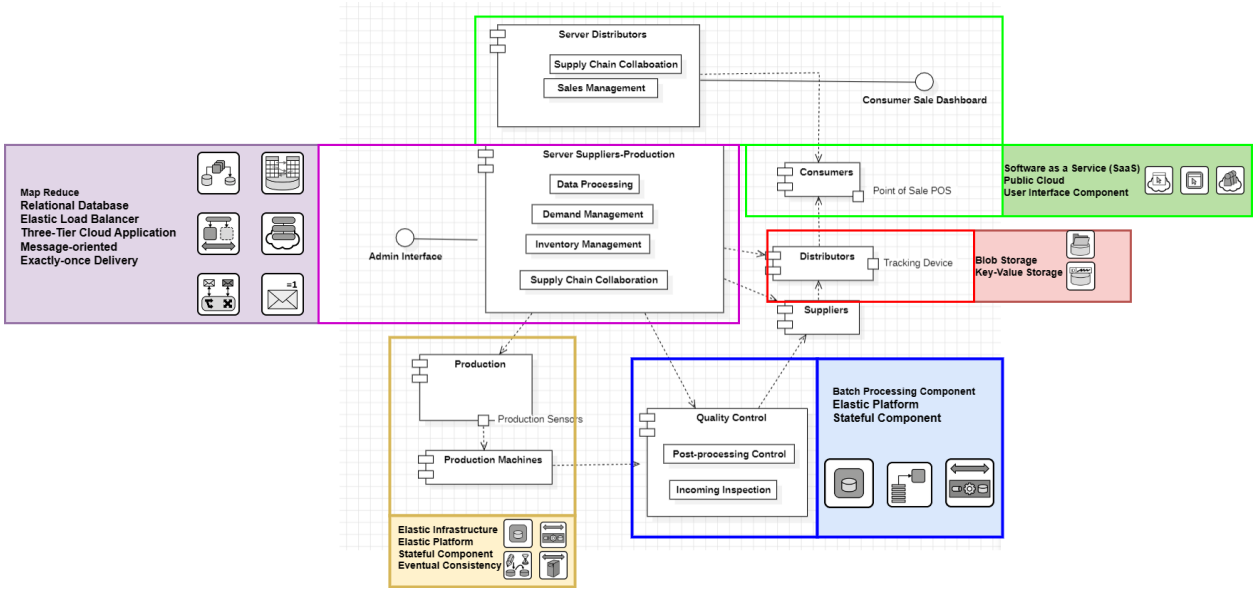


Figure 5.17: Porting to the cloud with Agnostic Cloud Vendor Pattern

Going into detail, the production and quality control process is designed to be flexible and adaptable to fluctuations in market demand. For this reason, an elastic infrastructure and platform was chosen to be used, with the ability to increase or decrease capacity dynamically. Data traceability and consistency are critical for efficient production, so components such as Stateful and Eventual Consistency were introduced for suppliers and distributors. Edge devices, which track products, generate a large amount of unstructured data. As a result, it was decided to use Blob Storage to manage and store this data, while Key-Value Storage [174] provides quick access to information using unique keys. Turning to consumers, which is the final stage of the process, Point of Sale systems, especially in the retail sector, need constant updates and real-time functionality. In this regard, the Software as a Service model was chosen to ensure real-time updates and an intuitive user interface for both sales staff and customers. The two primary servers, the Supplier-Producer Server and the Consumer Server, handle centralized data management. The first server handles large amounts of information from production, quality control, and tracking, so distributed processing via Map Reduce is essential. In addition, the Three-Tier Cloud Application approach [175], which separates user interface, processing, and data storage, is critical for efficient management. Another crucial aspect of the architecture is the collaborative supply chain model. To build a federated architecture on the cloud, message-oriented middleware, and exactly-once are used to avoid duplicate messages. Finally, quality control, which often involves large batches of products, uses batch processing, ensuring efficient parallel processing.

5.2.5 Conclusion

In conclusion, the work presented provides an in-depth view of military supply chain dynamics in a federated cloud continuum. The proposed architecture combines edge and cloud technologies to optimize manufacturing, quality control, and distribution operations, addressing the challenges of managing complex networks and the flexibility required by rapid changes in market demand. By adopting a collaborative model across multiple supply chains, supported by efficient data management and secure communication between different entities, greater scalability and resilience are ensured. In addition, porting to the cloud with an agnostic [176] approach ensures independence from specific suppliers and adaptability to different operational scenarios. Using technologies such as MapReduce, blob storage, batch processing, and message-oriented middleware-based collaboration models, this architecture is a step forward in modern military supply chain management, offering greater efficiency, reliability, and responsiveness to changing operational needs.

5.3 Pattern Recognition in Air Quality Monitoring

After addressing the federated cloud continuum through the previous case study in the next work we will use learning algorithms in this context. This third work was carried out at the IIIA office in Barcelona on the K-HiA: Knowledge for improving indoor Air Quality and Health (E.U. Project). It culminated in the publication of a paper at the 3GPP conference. In this section, we will discuss both the methodology and preliminary tests already published and the experimental results obtained, which will soon be submitted for publication in a scientific journal. When we talk about air quality, there are two strands of study: causes and effects. In the work, we will show that the focus will be on the first objective. This study aims to develop a method to understand how daily activities affect indoor air quality in homes in Valladolid, using a partially labeled real-world dataset. The main purpose is to analyze and model environmental factors by applying artificial intelligence techniques, with a specific focus on indoor air quality. Through the use of pattern recognition and machine learning methods, it aims to identify variations in air quality related to different activities in the monitored rooms. The ultimate goal is to better understand the causes of air quality fluctuations and identify recurring patterns to recognize the factors that contribute to poor air quality and its negative impacts on human health. Different approaches, global and federated, with related supporting architectures will be defined.

5.3.1 Data Collection

The project's first phase was initiated from data collected through sensors placed in six homes' kitchen and living room. The sensors are capable of measuring the following variables: humidity,

CO2 (ppm), formaldehyde (µg/m3), TVOC (ppb), TVOC index, PM 1.0 (µg/m3), PM 4 (µg /m3), PM 10 (µg/m3), PM 2.5 (µg/m3). The data collected by the sensors are recorded at regular 10-minute intervals for a period of 14 days. In this study, we will mainly focus on the sensors installed in the kitchen. This choice is dictated by the fact that by preliminarily observing the data, it is clear that it is the room where the most activity takes place. Moreover, in addition to the data captured by the sensors, each household in the monitored homes was asked to fill out detailed reports on the activities performed during the monitoring period.

5.3.2 Methodology

After collecting all the data, the first step was to establish a methodology to detect patterns. This consists of the following steps, which we will analyze in detail.

Pre-processing of data

The initial phase involves pre-processing of the data, a crucial step since the information in the reports is expressed in natural language. Although a human being easily understands this, a computer may encounter many difficulties in analysis, with the risk of generating errors. There are two approaches to address this problem. The first involves using artificial intelligence techniques, such as natural language processing [177], to identify key points within the scenarios. The second approach involves performing this task manually. The latter method was chosen to minimize errors arising from data formatting, to obtain the most accurate measurements possible. Although it is a more time-consuming process, it was considered the most suitable. Another crucial aspect was to treat the data as time series [178]. This choice proved advantageous for several reasons:

- **Continuous Measurement**: in many monitoring applications, data develop continuously over time. The use of time series makes it possible to capture trends that emerge over time. Since parameters can vary significantly, this approach allows a better understanding of their future implications.
- **Time Scales**: time series analysis can be carried out on different time scales. This allows both the single event and the entire time series to be modeled, providing greater accuracy during testing.
- **Temporal Correlation**: data collected at successive times are often correlated with each other. Identifying and exploiting such correlations can significantly improve the accuracy of analysis.

Some transformations were applied to the time series to make the data more usable: null values were replaced with mean values. This was necessary because null values may be recorded due to

malfunctions such as power outages or sensor errors, compromising the analysis. In addition, a moving average was applied to each series. This technique proved useful for two main reasons. First, smoothing the data: the moving average allows short-term variations to be smoothed out, highlighting long-term trends. Second is noise reduction, which facilitates the identification of clearer and more relevant patterns. This technique allows for better observation of underlying trends, ignoring minor fluctuations that might be irrelevant to the overall analysis.

5.3.3 Definition of algorithm and hyperparameter

To identify patterns, work was carried out to determine the largest possible number of hyperparameters and to select the most suitable model for detecting similar patterns. Regarding the first point, the following algorithms were analyzed.

1. **Euclidean Distance**: A metric that measures the distance between two data points. It is calculated by extracting the square root of the sum of the squared differences between the corresponding coordinates of each point [179]. The formula is as follows:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

   where p and q represent the data points in n-dimensional space and n is the number of dimensions. Despite its usefulness, Euclidean distance is remarkably sensitive to noise and outliers. These sensitivities can limit its effectiveness, especially in environments such as homes, where constant external events influence data. Although we have implemented techniques to significantly reduce these noises, Euclidean distance may not always be the most reliable method for assessing the similarity between data in these contexts.

2. **Dynamic Time Warping (DTW)**: A technique used to measure the similarity between two temporal sequences that may vary in time or speed. Unlike methods such as Euclidean distance [180], DTW dynamically aligns sequences to minimize temporal differences and calculate an optimal path that highlights the most similar alignment. The formula is as follows:

$$dtw\_matrix[i,j] = \min \begin{cases} dtw\_matrix[i-1,j-1] + d(ts\_A[i], ts\_B[j]), \\ dtw\_matrix[i-1,j] + d(ts\_A[i], ts\_B[j]), \\ dtw\_matrix[i,j-1] + d(ts\_A[i], ts\_B[j]) \end{cases}$$

3. **Symbolic Aggregate Approximation (SAX)**: A time series analysis method that transforms complex data into simple symbols [181]. The series are divided into segments of equal

length, the average value of each segment is calculated, and they are associated with symbols based on predefined quantiles of a normal distribution. This method is widely used for time series pattern recognition, clustering, and anomaly identification.

$$\alpha[i] = j \text{ if } q_{j-1} \leq v_i < q_j$$

- α: represents the string of symbols resulting from the SAX transformation.
- i: an index that scrolls through the time series segments.
- j: symbol index, determined by the position of the segment average value relative to the breakpoints.
- q_{j-1} and q_j:are the breakpoints that define the intervals for the symbolic transformation; these are the quantiles of the normal distribution that divides the value axis into a equally probable intervals.
- v_i: the average value of the i-th segment of the time series used to determine the corresponding symbol.

4. **Pearson correlation**: A technique that quantifies the linear relationship between two variables [179]. It is used to identify the strength and direction of a linear relationship between two time series. In formulas:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

5. **Levenshtein distance**: Also known as edit distance, it is a metric for measuring the difference between two strings [182]. It represents the minimum number of single modification operations (insertions, deletions, or replacements) necessary to transform one string into another.

$$d(a,b) = \begin{cases} |a| & \text{se } |b| = 0 \\ |b| & \text{se } |a| = 0 \\ d(\text{tail}(a), \text{tail}(b)) & \text{se head}(a) = \text{head}(b) \\ 1 + \min \left\{ \begin{array}{l} d(\text{tail}(a), b), \\ d(a, \text{tail}(b)), \\ d(\text{tail}(a), \text{tail}(b)) \end{array} \right\} & \text{altrimenti} \end{cases}$$

Among this, the choice was to use Dynamic Time Warping (DTW) in combination with the K-

Means algorithm [183]. This decision was driven by promising preliminary results, which indicate that this combination is particularly effective in specific time series analysis contexts. After that, the second goal was to identify the best set of hyper-parameters to search for patterns within each series. To do this, it is necessary to define a methodology for evaluating hyper-parameters, which is as follows: let n_i be the number of activities reported in the report for house i. We define the set of possible clusters as $K_i =: k\{in[n_i, n_i + 5]\}$. For each time interval $T \in \{24, 12, 6, 3\}$ hours, we perform the following procedure:

1. Divide the time series into segments of length T.
2. For each $k \in K_i$, apply clustering to divide the segments into k clusters.
3. Identify for each cluster j, the activity $\hat{a}_{ij}$ that maximizes the frequency of occurrence in the cluster.

Comparison of the identified $\hat{a}_{ij}$ activities with those reported in the report determines the validity of the association for each cluster. We define the success rate for house i as:

$$\text{success rate}_i = \frac{\text{Number of matches between } \hat{a}_{ij} \text{ and the report}}{\text{Total number of activities in the report}} \times 100$$

To identify the pattern associated with an activity after grouping the various time series according to the best parameters, it is necessary to identify the pattern. Therefore, through the use of specific techniques, the associated motif for each best outcome will be identified. These will then be compared between the different houses to see if, given two identical activities, they have the same motif. After obtaining all the information previously outlined, the final goal will be to create a report that can associate for each activity which patterns were found relative to which series with a trash-hold of 65% positive results. This will allow us to understand how effective this algorithm has been in numerical terms. Next, the different patterns will be compared to isolate only recurrent patterns in the series. Once a clear methodology was defined, the next step was to implement it to verify the correctness and validity of this algorithm.

5.3.4 Implementation

The first thing to be defined for the implementation was the development environment. In this case, due to the type of project and algorithm, the choice fell on Google Coolab. The reasons for this choice were mainly 2:

- **Ease of use of libraries**: We need to use different techniques in this project. In the implementation, this results in the use of different libraries. In this case, this environment makes it much easier to use since, in most cases, we do not need to install them but only import them

- **Distribution not required**: For this type of implementation, there is no need to distribute or split the application into frontend and backend so the choice to use single working pages in a single programming language was considered the best choice.

Python was used exclusively as a programming language. The flexibility of this language, combined with the vast number of libraries for manipulating data frames and the use of artificial intelligence techniques, certainly made this language the best choice. Among the main libraries to be used we have: pandas for all the dataframe manipulation part and in general to develop what was mentioned in the methodology in the "Preprocessing" section, matplotlib for the visualization of the various graphs, and tslearn regarding the use of KMeans. Specifically, in this particular case, a function specialized for TimeSeries called TimeSeriesKMeans was used. Regarding the algorithm described in the methodology above, we will not report the entire code for the sake of brevity but describe the process using the following pseudocode:

Algorithm 6 Analysis of activities for home

for each house i **do**
    $n_i \leftarrow$ number of activities in the report
    for $K_i = n_i - 2$ **to** $n_i + 2$ **do**
        for $T \in \{24, 12, 6, 3\}$ **do**
            Divide the time series into intervals of T hours
            Perform clustering with K_i cluster
            for each cluster j **do**
                Counts the number of activities r_{ij}
                Finds the most frequent activity in cluster j
                Associates the most frequent activity with cluster j
            end for
            Compares the cluster activities with those in the report
            Calculates the success rate
        end for
    end for
end for

The one iteration of this pseudocode provides the following output as a result Fig 5.20:

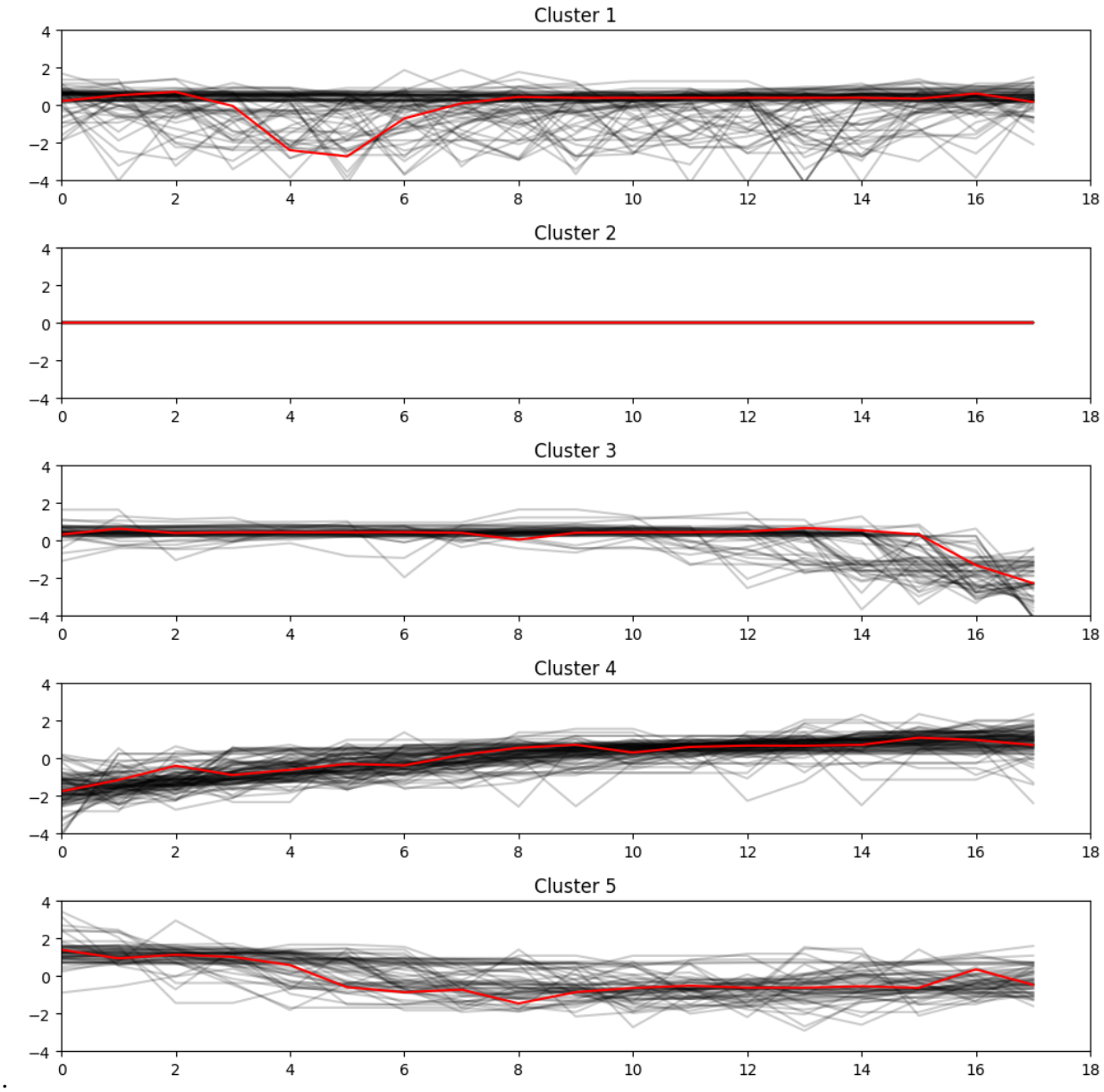


Figure 5.18: A representative example of one of the 900 iterations evaluated within the executed simulations

5.3.5 Validation and results

After pre-processing the data in all houses, the following activities were isolated: Household appliances (A1), Ventilation (A2), Eating (A3), Cooking (A4), Cleaning (A5), and Smoking (A6). Among these, smoking is the only activity that is not present in all houses, and ventilation is present everywhere except in house 4, and smoking is present only in house 4. As mentioned in the methodology in the table for each house, only the models with more than 65% accuracy will

be reported. The models analyzed for 24, 12, and 6 hours did not return optimal results. The best scores were given only by the 3-hour models, so to avoid making the writing too heavy, 3H will not be mentioned explicitly but will be implied. Below for each house and activity, this is the table produced with the various results obtained with the help of over 900 simulations:

Table 5.1: TimeSeries Report Results - Activity Report

H and A	**Variables**	**N.C.**	**M.C.C.**	**Score %**
H1-A4	pm4	6	1	87.50%
H1-A5	Humidity (%)	7	1	80.00%
	Formaldehyde (µg/m3)	8	0	80.00%
	TVOC (ppb)	5	1	80.00%
	pm4	5 & 6	0	80.00%
	PM 10 (µg/m3)	5	0	80.00%
	PM 2.5 (µg/m3)	8	1	80.00%
H2-A1	pm4	6	0	85.71%
	PM 10 (µg/m3)	5	0	85.71%
	PM 2.5 (µg/m3)	7	0	85.71%
H2-A5	Ventilation Indicator	5	0.0	80.00%
	CO2 (ppm)	5	0	80.00%
	TVOC (ppb)	7	1	80.00%
	PM 10 (µg/m3)	6	0	80.00%
H3-A1	Formaldehyde (µg/m3)	6	0	85.71%
H3-A3	Formaldehyde (µg/m3)	6	0	76.92%
H3-A5	IAQ Indicator	10	0	80.00%
	Humidity (%)	6	3	80.00%
	TVOC (ppb)	7	1	80.00%
	pm4	5	0	80.00%
H4-A1	Formaldehyde (µg/m3)	5	0	87.50%
H4-A3	TVOC (ppb)	6,9 & 10	0	80.00%
	PM 1.0 (µg/m3)	5	0	80.00%
	pm4	6	0	80.00%
	PM 10 (µg/m3)	7	0	80.00%
H5-A1	CO2 (ppm)	6	1	85.71%

Continue on the next page

Table 5.1 –continues from the previous page

H and A	Variables	N.C.	M.C.C.	Score %
	TVOC (Index)	9	0	85.71%
H5-A5	CO2 (ppm)	6	1	85.71%
	TVOC (Index)	5,6	0	85.71%

Based on these results, the motifs of the individual series were displayed in the table shown. Below are the same motifs associated with the same series based on the same actions in different houses. The following table summarizes the time series associated with different activities and provides information on whether they have the same motif.

Activity	Measurement	Present Houses	Same Motif
A1	Formaldehyde (mg/m3)	H3, H4	2/2
A5	Humidity (%)	H1, H3	1/2
	TVOC (Index)	H1, H2, H3, H5	3/4
	pm4	H1, H3	2/2
	PM 10 (µg/m3)	H1, H2	2/2

Table 5.2: Summary of the same measures in different houses with the same activity

The above table 5.2 shows that most of the time series analyzed have the same motif given the fixed clusters. Below are the reasons for each activity for some of the best results:

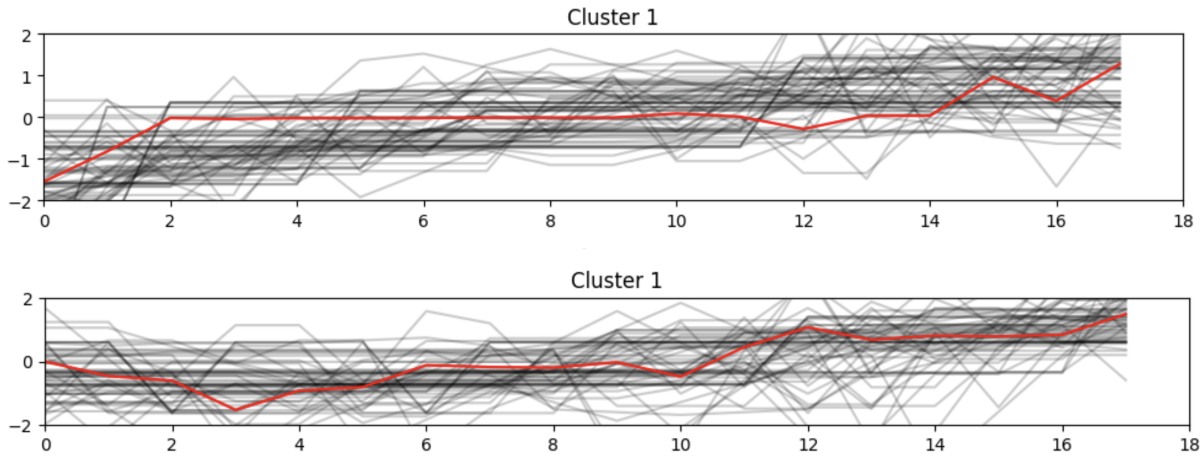


Figure 5.19: Pattern identified for the Formaldehyde data series associated with Activity 1 in houses 3 and 4

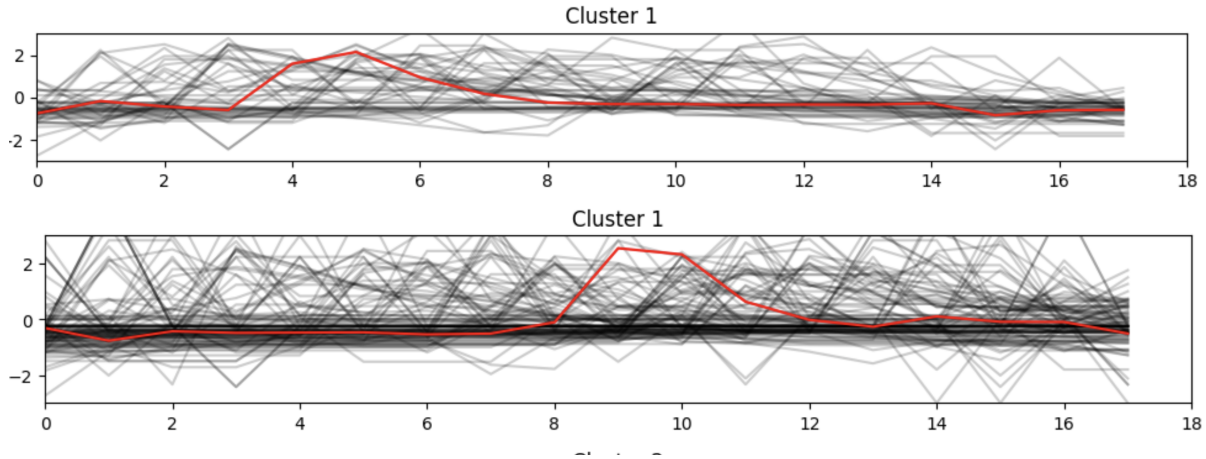


Figure 5.20: Pattern identified for the PM4 data series associated with Activity 5 in houses 3 and 4

5.3.6 Federated Learning Approach

The use of a federated approach in such a context is a key resource for two main reasons: it allows for privacy on devices by not directly sharing data, but rather the weights of the associated models, and also for moving a portion of computation to the server side. At the same time, it allows for the generalization of results by comparing different data from different houses. In this case, the algorithm changes slightly since the activities are first identified in individual houses and then aggregated. The basic idea is to send only the models with their cluster centroids and then be able to aggregate and average the centroids [184]. This research can represent a preliminary work on creating an architecture in a federated scenario for a specific air quality problem case study. The ultimate goal of the entire work is to create a secure, robust, and scalable infrastructure for analyzing environmental data using learning models.

5.3.7 Architectural Choices and Mitigation of Challenges in Federated Learning Using Patterns

The architecture of this system is therefore defined in Fig.5.21 in which the parts of the code described in the various deployed components are mapped

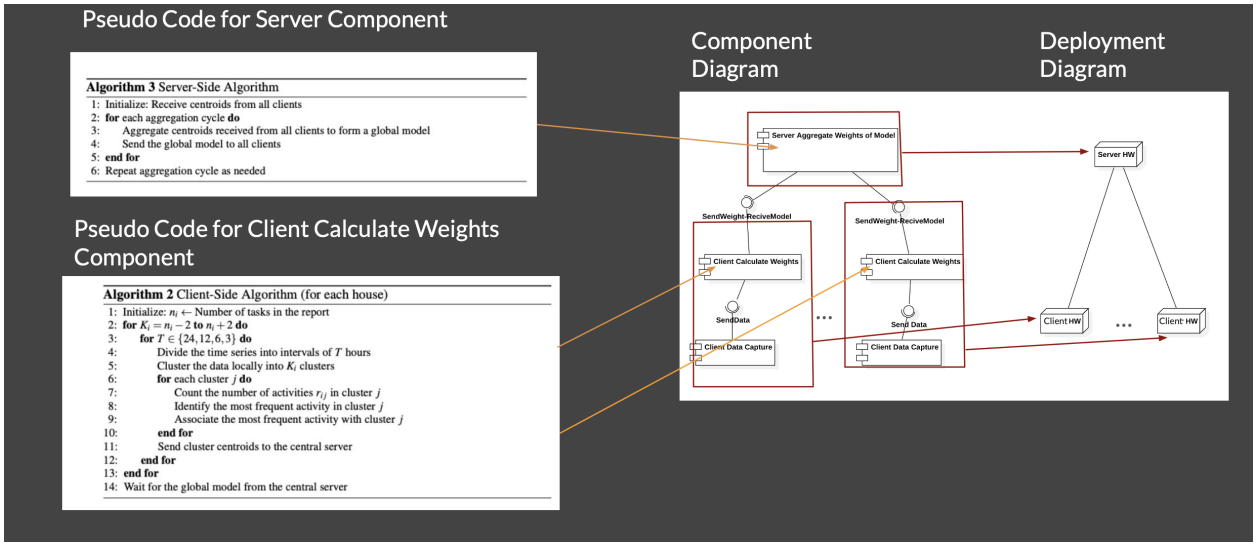


Figure 5.21: Source code mapped to individual components, which in turn have been mapped to the individual nodes of the deployment

In more detail, the components are: Server Aggregate Weight of Model: responsible for aggregating and returning the model weights to the clients. Client Calculate Weight is responsible for re-calculating the model from the weights and returning the new psi to the server. Client Darta Captured, which instead only takes care of capturing the data and sending it to the Client Calculete Weights. In edge computing and the Internet of Things (IoT), the analysis and use of patterns, or generalized solutions to common problems, play a crucial role. The application of patterns is essential to maintain system flexibility, scalability, and robustness while addressing emerging complexities. In this specific case, let's see for each of the problems mentioned in the previous chapter what they might cause and what patterns are best to solve the following challenges Fig 5.22.

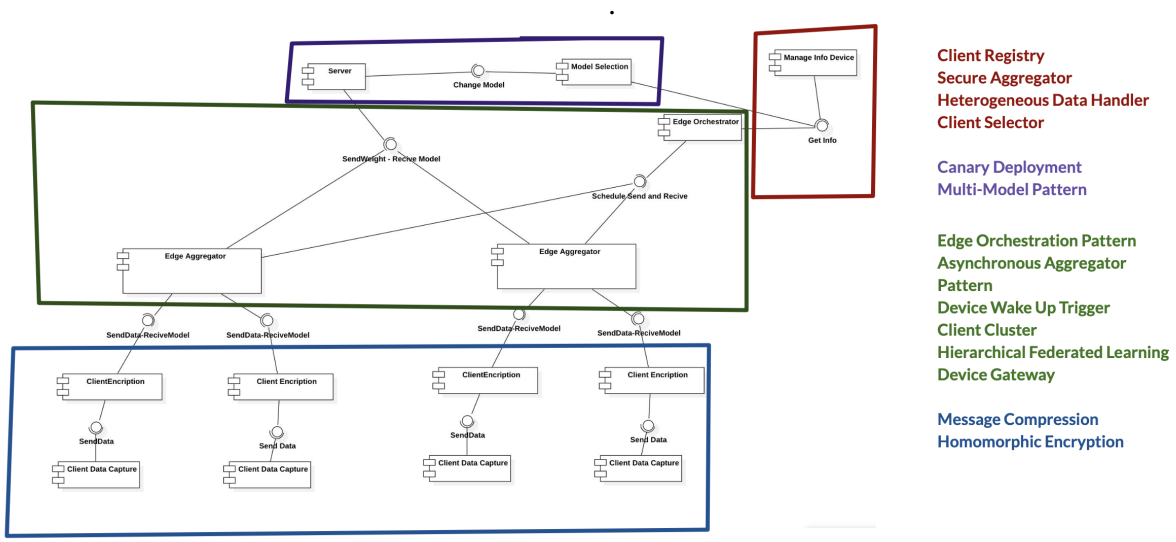


Figure 5.22: Component diagram of the internal architecture mapped with the respective agnostic patterns for cloud deployment

Management of data heterogeneity

Data heterogeneity is one of the main problems of federated learning. Because the architecture is distributed across different geographic sites, sensors may be from different vendors and use technologies that are not similar to each other. This results in data acquisition in different ways and at different times (at intervals of 5, 10, and 15 minutes). If this problem is not handled properly, it could give rise to numerous errors. The main problem is the difference in data acquired at a given time interval, T, that could result in different weights between 2 clients. One solution is to use the model **Heterogeneous Data Handler** [185]. This deals with balancing the distribution of data within clients through a process called federated distillation. In addition, another possible approach could be to use different patterns based on quantitative and qualitative analysis of [186] data. The pattern **Multi-Model** could then be combined with the **Client Registry**. Different models would then be distributed, and the client registry would be used to decide which of these models the individual client should use. Another problem could be caused by the fact that if people are not at home, the sensor does not detect activity, and therefore, the measurements could negatively affect the other models. The solution could then be to set up a shutdown trigger in case people were not at home through the implementation of the pattern **Device Wake Up Trigger** [187] and, through the use of the **Client Selector** pattern, decide which clients participate in the training based on whether the device is off or on.

Security Risks

One of the main problems with federated learning is the security of the architecture. The fact that data is not shared and users are anonymous can cause a whole range of security problems. The first is the possibility of a malicious user pretending to be a client, participating in training, and entering incorrect weights into the server. This problem can be easily solved through or again using the **Client Registry**, which allows tracking of devices. Also, using a pattern for **Secure Aggregator** [188] could solve this problem. Another problem could be related to data reconstruction. This is because recent studies have shown that even if only the weights are sent, it is possible to reconstruct the data through reverse engineering techniques. Reconstructing [189] data in a case study such as the one defined above would lead to the sharing of extremely confidential information. Homomorphic cryptography could be used to solve this problem, but the **Hierarchical Federated Learning** pattern [190] could help mitigate the problem. The idea could be to perform aggregation on multiple levels; for example, in the case of apartment buildings, all thoughts could first be aggregated within a dedicated edge component and only then send the aggregated weights to the server. This might not completely solve the problem but mitigate it since, currently, with current data reconstruction techniques, it is extremely complex to reconstruct them if they come from different clients [191].

Computational Costs

Another problem is related to the computational capabilities of individual edge devices. This problem is divided into two subproblems. The first is the limitation of processing capacity, which leads to processing delays. The processing delay of a single client consequently delays the entire topology. In a context such as several houses located at different sites, the heterogeneity of sensors definitely causes a difference in performance. From this point of view, several patterns can be implemented, including **Asynchronous aggregator Pattern**, which allows aggregation asynchronously so that individual "lagging" clients do not affect the rest of the group. The second subproblem is the device's poor battery life and/or power consumption. In these cases, the pattern **Device Shadows** could help from this point of view.

Communication coordination and complexity

The last problem, which is no less important, is related to the communication and coordination problems of devices. In this context, the first that should be mentioned is the **Device Gateway** that allows even devices that cannot connect to the network because they do not support the required technological communication (many sensors suffer from this problem) to interface via a gateway. The second problem that might be of concern is the management and orchestration of different

devices. Especially if the patterns mentioned above were implemented, the multicomponent architecture would be quite complex. In this circumstance, using the **Edge Orchestration Pattern** would help manage a multi-component architecture. Another problem related to communication could be network delays. In this context, the pattern weights, although smaller than the data, could cause delays in the sending stages for some clients. To partially mitigate this problem, any message compression mechanism could be implemented to ensure less data is sent. The goal would then be to sacrifice computational load to reduce bandwidth. This solution could be applied to a small group of clients who, for geographical reasons, live in sites where the connection does not work properly. Finally, one last model that could be used in this context is **Canary Deployment** [192]. In particular, it is not certain that once the model is created, all homes will record the same results. Consequently, it would be interesting to apply this model to select the various houses that do not respond well to one model and move them to another with different characteristics. From there, test whether the new model generally performs better than the previous one by distributing the beta model to more and more people

5.3.8 Conclusions and prospects

In conclusion, this study proposes a detailed methodology to analyze the impact of domestic activities on air quality in homes. This methodology represents only a starting point and requires further validation. The next phase could include the definition of several case studies and the application of various measures to verify the effectiveness and accuracy of the results obtained. Only through this validation process will it be possible to confirm the robustness of the proposed approach and its applicability in real contexts, thus contributing to a better understanding and management of indoor air quality. Furthermore fine-grained approach, for example differentiating the types of cleaning or household appliances used, could guarantee more precise results. In addition, the use of more edge components [115] to be able to better divide the tasks and optimize the entire process could be another important factor to consider.

5.4 Predicting Glucose for Diabetic Patients

Finally, the last work done from the perspective of learning algorithms was the monitoring and analyses of diabetic patients through techniques of machine federated learning and architectures defined with the use of cloud, experimented in previous studies. This analysis culminated in two publications: one presented at the international conference Advanced Information Networking and Applications (AINA 2023) and the other published in the journal Connection Science, Volume 35. The following tasks were carried out in these works:

1. **Comparative Analysis of Machine and Deep Learning Techniques:** We conducted a comprehensive comparative analysis of machine learning techniques, including Linear Regression, Polynomial Regression, Lasso Regression, Ridge Regression, and LSTM (Long Short-Term Memory) deep learning, ARIMA models, and VARIMA and SARIMA for glycemic trend analysis and prediction. This study provides important insights into the performance of these algorithms applied to real data obtained from glucose sensors.

2. **Implementation of Federated Learning**: In this phase, the goal is to adapt and implement existing algorithms to the context of Federated Learning in a virtual environment.

3. **Defining Different Architectures**: This task aims to define different architectures that can support the proposed hypotheses, ensuring flexibility and scalability.

4. **Simulation and Performance Evaluation**: Through real simulations with a limited number of users and larger simulations using specific tools, the effectiveness of the defined architectures in terms of performance will be evaluated.

.

5.4.1 Description of the E-health case study

Before getting to the analysis's heart, let us explicate the case study. We begin by defining the term *glycemia*, which refers to the concentration of glucose in the bloodstream, measured in milligrams per deciliter (mg/dL). Under normal conditions, for adults, blood glucose levels are around 60/70 mg/dL on an empty stomach, with postprandial increases reaching 130/150 mg/dL. The human body has several adaptive mechanisms to keep glucose within these physiological limits, essential for health. Glucose is the main energy source for tissues, especially the brain, which depends on a constant supply to function properly. However, there are situations where the body's regulatory mechanisms may malfunction, leading to imbalances in blood glucose levels. One such imbalance is hyperglycemia, commonly associated with diabetes. This work focuses on predicting blood glucose levels in patients with type 1 diabetes, considering the amount of insulin administered and current blood glucose levels. In patients with type 1 diabetes, the mechanism of blood glucose regulation is impaired, with fasting values tending to stabilize around 126 mg/dL, significantly above normal (which should be less than 100 mg/dL). Figure 5.23 shows the threshold values of blood glucose. The data used in this study came from an insulin pump *Medtronic 780G* [193], which regulates insulin delivery based on data provided by glucose sensors applied to the patient's skin. The patient can monitor glucose levels and insulin administered and download the data in CSV format.

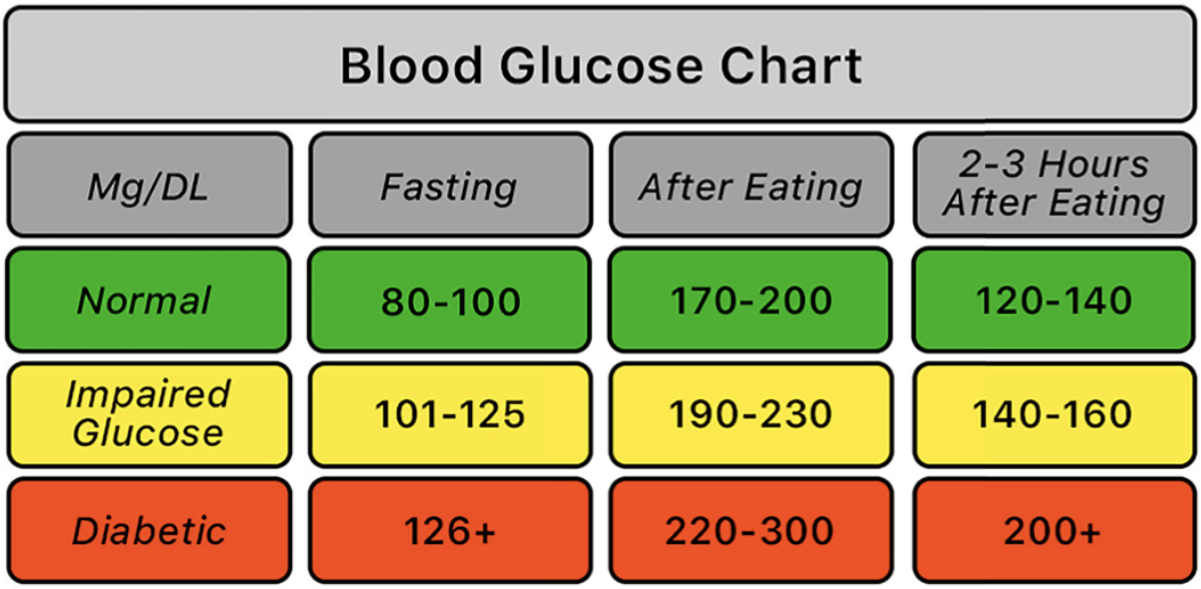


Figure 5.23: Reference glucose levels

5.4.2 Data and Metrics Used

The data collected in this study involve sensor data delivering measurements every 5 minutes over a 15-day time interval for more than 20,000 samples. The pump data include more than 50 attributes, covering various aspects of the patient's daily glycemic trends. However, not all attributes are relevant to this study. In this regard, the first step was filtering the data and selecting only the necessary information. Therefore, the data used were:

- *Bolus Volume*, which indicates the amount of insulin injected at a specific time;
- *Sensor glucose*, which represents the blood glucose levels measured by the sensor;
- *ISIG value*, a raw signal to assess the correct function of the sensor.

Even after this initial filtering, the dataset has issues. The presence of errors and outliers is likely to adversely affect the performance of machine learning models. Although these outliers can be useful in certain cases, such as fraud detection, they must be eliminated in our case. We can use several techniques to detect and handle outliers in our dataset. One of the most common is based on the interquartile range (IQR), which is calculated as the difference between the third quartile ($Q3$) and the first quartile ($Q1$):

$$IQR = Q3 - Q1$$

Next, we can determine thresholds to identify outliers. A data point x is considered an outlier if:

$$x < Q1 - (1.5 \cdot IQR) \quad \text{or} \quad x > Q3 + (1.5 \cdot IQR)$$

The Python code script for calculating quartiles and filtering outliers is shown in Code 7.

Algorithm 7 Code used to calculate the Quartiles

$q25, q75 = \text{percentile}(df, 25), \text{percentile}(df, 75)$
$iqr = q75 - q25$
$cut\_off = iqr \times 1.5$
$lower = q25 - cut\_off$
$upper = q75 + cut\_off$
print($lower$)
print($upper$)
df = df.loc[df['Sensor Glucose (mg/dL)'] ¿ lower]
df = df.loc[df['Sensor Glucose (mg/dL)'] ¡ upper]

We obtained a lower threshold value of -151 mg/dL and an upper threshold value of 252.2 mg/dL based on our calculations. As might be expected, the lower threshold does not change the dataset, while the upper threshold mainly removes some peaks. We can also introduce an alternative method based on standard deviation for completeness. This technique calculates the threshold values as follows:

$$\text{Lower} = \mu - (1.5 \cdot \sigma)$$

$$\text{Upper} = \mu + (1.5 \cdot \sigma)$$

Where μ is the mean glucose and σ is the standard deviation. Again, factor 1.5 is arbitrary but commonly used in the literature. The thresholds calculated by both methods (IQR and standard deviation) are very close, so we can use either without significantly affecting the final results.

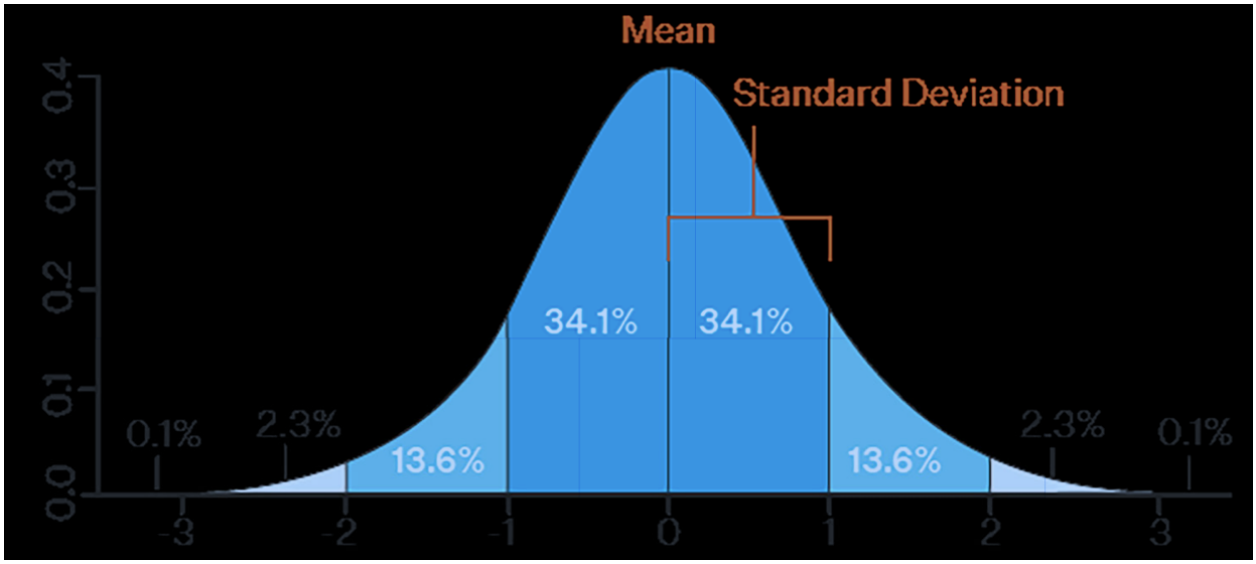


Figure 5.24: Use of standard deviation to remove outliers

.

Regarding the metrics used for the evaluation there are:

- **RMSE (Root Mean Square Error)**: RMSE measures the average magnitude of errors or residuals between predicted and actual values in a regression or prediction model. It quantifies the standard deviation of these errors.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

  Lower RMSE values indicate higher model accuracy. RMSE is expressed in the same units as the target variable.

- **R-squared (R^2)**: R^2 measures the proportion of the variance in the dependent variable (target) explained by the independent variables (features) in a regression model. The value ranges from 0 to 1, with higher values indicating a better fit.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  A value of $R^2 = 1$ indicates a perfect fit, while lower values suggest a less accurate model fit.

- **MAPE (Mean Absolute Percentage Error)**: MAPE measures the percentage difference between predicted and actual values. It quantifies the mean relative error of a model, making

it suitable for comparing accuracy between different data sets.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

Lower MAPE values represent higher accuracy, with 0% indicating perfect prediction.

- **MBD (Mean Bias Deviation)**: MBD measures the mean bias or systematic error in forecasts. It quantifies a model's average overestimation or underestimation relative to actual values.

$$MBD = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

A positive MBD indicates overestimation, while a negative MBD indicates underestimation. A value of 0 suggests the absence of systematic bias.

5.4.3 Application of Machine Learning Techniques

To apply Machine Learning algorithms, the inputs and outputs must be defined. The goal is to predict the patient's future blood glucose values from the current glucose concentration (measured by the sensor) and the amount of insulin injected (bolus). We use four inputs: date, time, blood glucose, and bolus volume, and also divide the data into train and test sets. Specifically data collected from 02/01/2022 to 03/15/2022 are used for training, while the remaining data are used for testing. Figure 5.25 shows a graphical representation of the whole dataset.

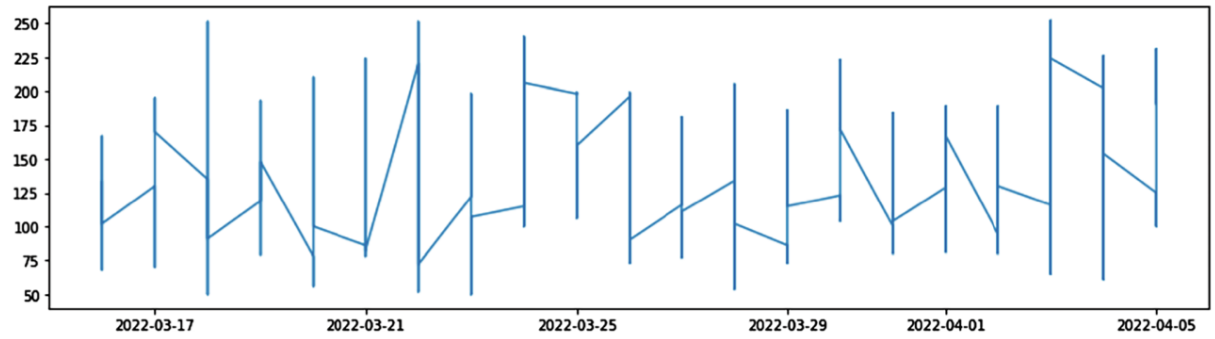


Figure 5.25: Graphic representation of the complete dataset. The x-axis shows dates, and the y-axis represents blood glucose values in mg/dL

5.4.4 Regression techniques

The first experiments used machine learning techniques, particularly regression techniques. Including:

- Classical Linear Regression;
- Ridge Regression;
- Lasso Regression.

Scikit-Learn was used as the prince library to implement all the algorithms. Figure 5.26 shows the results obtained with Lasso Regression, in which the test data are shown in blue and the predictions in red. Training data are not shown; they are used only to create the regression model.

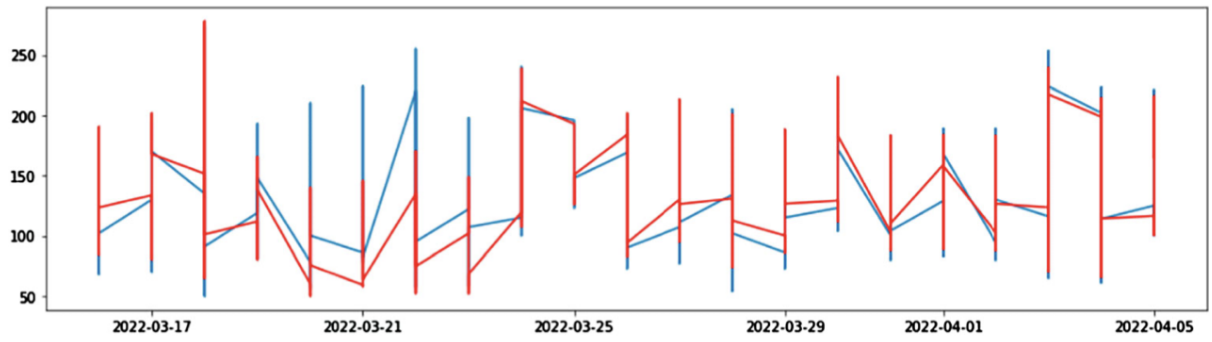


Figure 5.26: Lasso Regression results over the entire test period. Blood glucose levels are expressed in mg/dL

However, given the density of the data, Figure 5.26 is not very clear, we also show the single-day visualization, as shown in Figure 5.27.

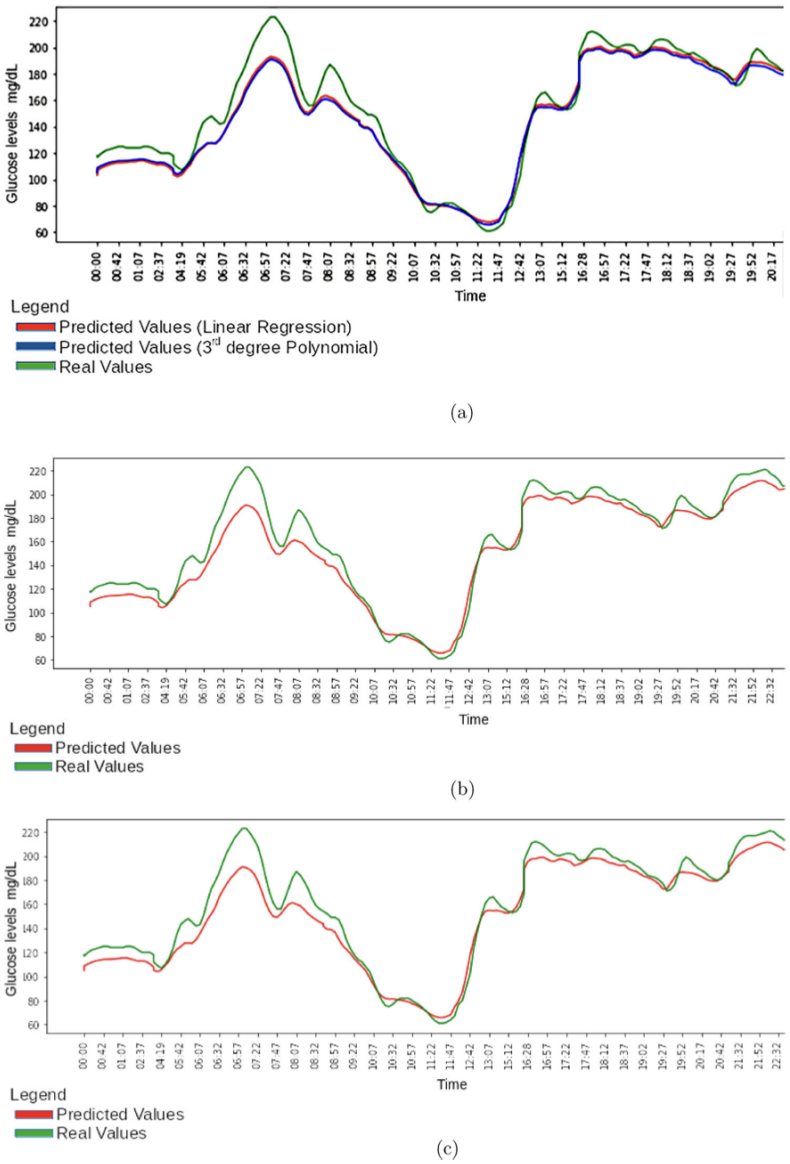


Figure 5.27: Comparison of the Regression results over a single day. (a) Results of Linear and Polynomial regression over one day, extracted from the test set. All glucose levels are in mg/dL. (b) Results of Lasso regression over one day, extracted from the test set. All glucose levels are in mg/dL and (c) Results of Ridge regression over one day, extracted from the test set. All glucose levels are in mg/dL.

.

As we can observe, the algorithms follow the general trend of the data well, but struggle to accurately capture the positive and negative peaks. Table 5.3 reports the mean square error (MSE) and R^2 values for the various algorithms tested.

Also, it is important to add that the algorithms were run in the Google Colab environment, which allows Python code to run within an online notebook. Colab's default CPU is an Intel

Table 5.3: Comparison of the performance of regression algorithms in terms of MSE and R^2

Algoritmo	**MSE**	R^2
Linear Regression	420.90 $\mu g/dL$	0.7268
Lasso Regression	423.73 $\mu g/dL$	0.7326
Ridge Regression	423.73 $\mu g/dL$	0.7326
Polinomial Regression	427.6 $\mu g/dL$	0.7302

Xeon with two vCPUs (virtual CPUs) and 13 GB of RAM. Table 5.4.4 reports the four regression algorithms' performance evaluation.

Model	**Execution Time**	**Used RAM**
Linear	1 s	6 GB
Lasso	2 s	6 GB
Ridge	2 s	6 GB
Polynomial	25 s	12.5 GB

5.4.5 Application of Deep Learning Techniques

To test the deep learning algorithms in the case study, we used the Keras library and, in particular, the implementation of the LSTM (Long Short Term Memory). The data used as input is the same as in the regression case. Figure 5.28 describes the neural network created with the Keras library.

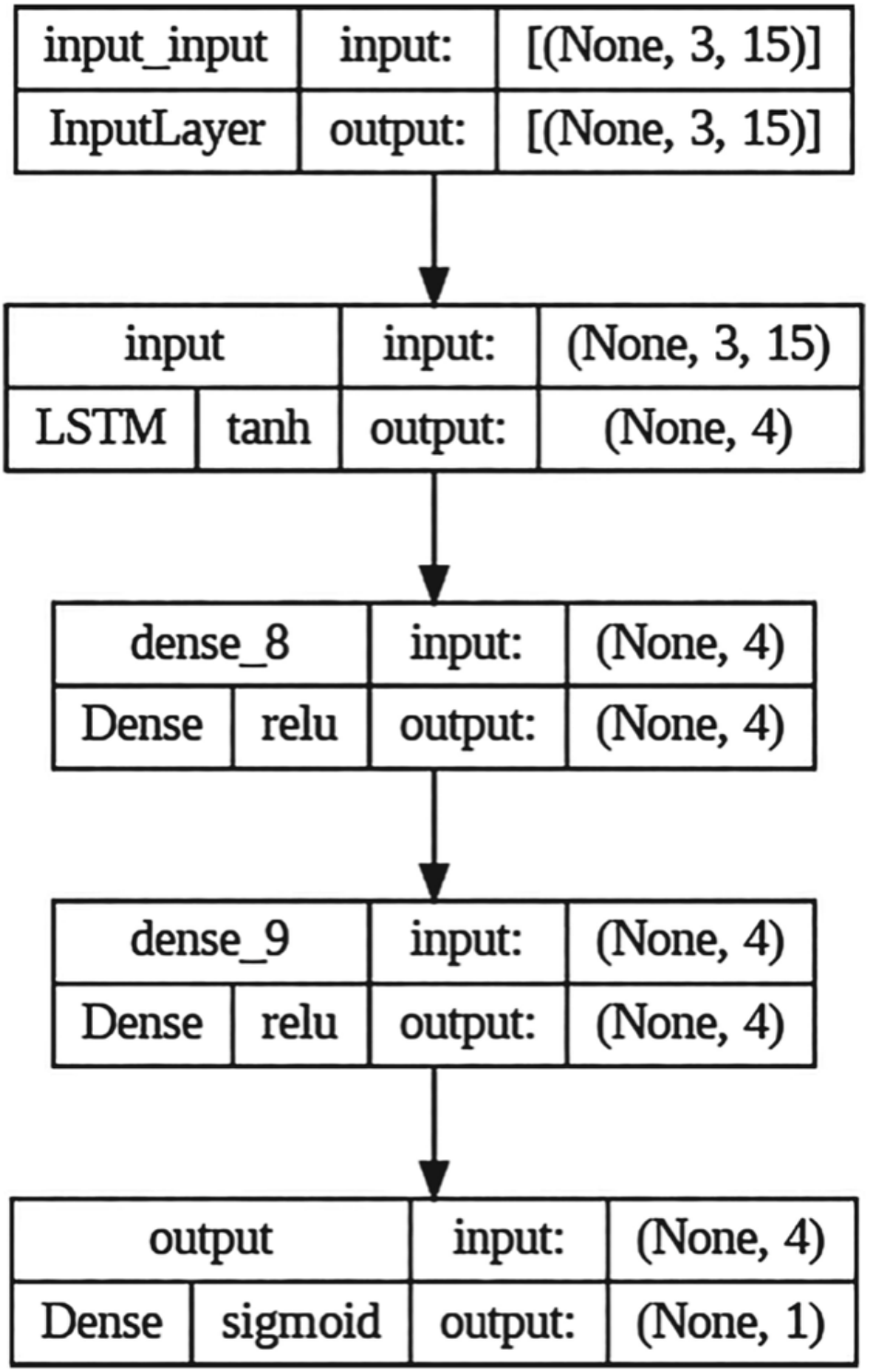


Figure 5.28: Organisation of the LSTM neural network

The network consists of an initial LSTM layer with three input neurons, corresponding to the

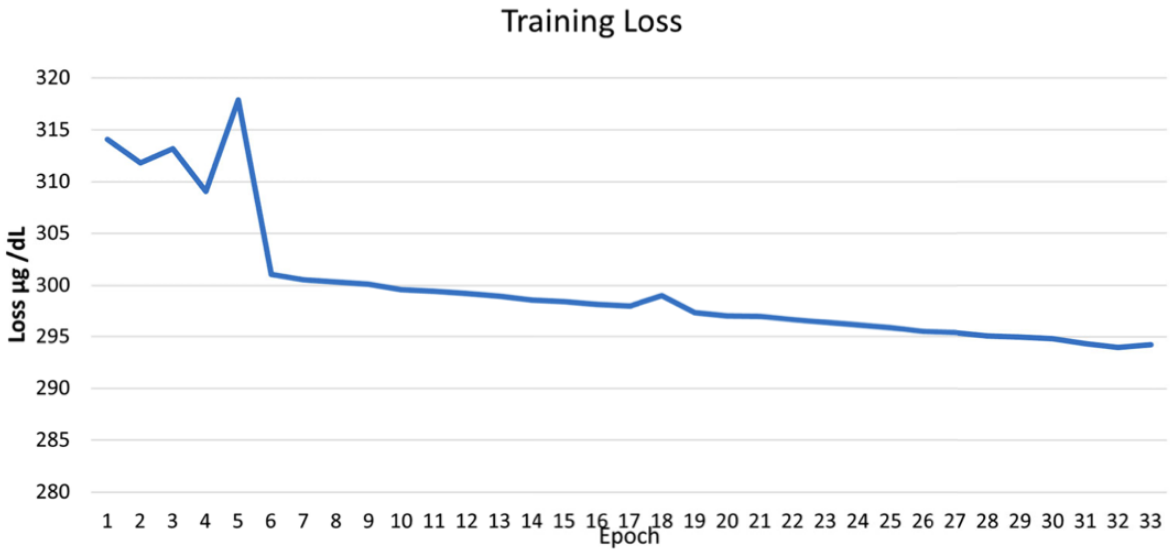


Figure 5.29: Training loss variation during the training epochs.

variables *Bolus Volume*, *Sensor Glucose*, and *Date + Time*, along with four intermediate neurons. The output layer consists of a single neuron, representing the predicted value of glucose. Two intermediate dense layers were used to construct the LSTM-based regressor. The parameter *look_back* was set to 3. This parameter instructs the LSTM level to store past information up to a specified number of previous inputs. Generally, it is set to a relatively high value, usually between 15 and 20 steps. However, glycemic values do not depend on parameters that are too far back, so a lower value of *lookback* was needed. Tests were carried out with *look_back* values equal to 3, 5, 10, 15, and 20, and the value three was the best. After 35 training epochs, the following results were obtained: *TrainingLoss* = 293.73 and *ValidationLoss* = 455.64. Figure 5.29 shows the loss trend during training. It can be seen that as early as the fifth epoch the results approach the final results. However, we continued the training for more epochs to obtain the best possible result. Although using a larger number of epochs may lead to overfitting, this did not happen in this case, as training was stopped after 35 epochs when the loss stabilized. Similar to what was done with regression models, we chose a specific date to compare the actual and predicted blood glucose trends. The selected day was April 4, 2022. Figure 5.30 shows the reference values: the red line represents the actual data, while the green line represents the predicted values.

Table 5.4: Performance comparison between the LSTM model and Linear Regression

Algoritmo	**RMSE** ($\mu g/dL$)	**R2**	**MAPE**	**MDB**
Regressione Lineare	420.90	0.7268	2.9%	3.5
LSTM	293.73	0.8520	1.8%	2.3

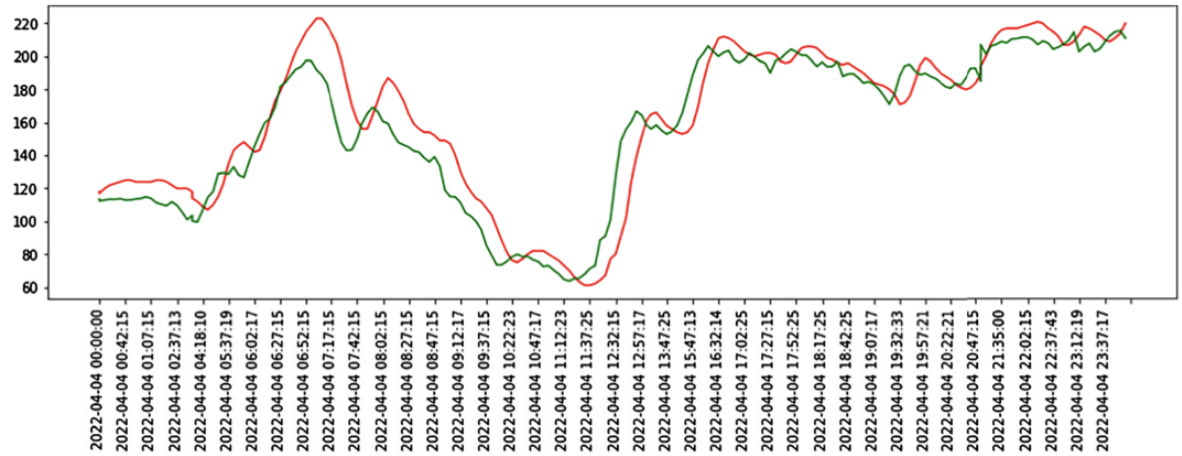


Figure 5.30: Comparison between the real glycemic trend (in red) and the predicted one (in green).

Table 5.5: Confronto delle prestazioni degli algoritmi di regressione su Google Colab (accesso gratuito).

Algoritmo	**Tempo di esecuzione (s)**	**RAM utilizzata (GB)**
Linear Regression	1 s	6 GB
Lasso Regression	2 s	6 GB
Ridge Regression	2 s	6 GB
Polynomial Regression	25 s	12.5 GB
LSTM	3 min with GPU	6 GB con GPU

An important factor to consider is the training time required for the models. Linear regression techniques require much less time than LSTM. So, finding a balance between the number of epochs and the desired loss is crucial. Therefore, linear, polynomial, Lasso, and Ridge regression algorithms have demonstrated reliability with low resource consumption and fast training times. The LSTM neural network offered better performance in terms of mean square error (MSE), allowing more accurate predictions of glycemic trends, but required a GPU to run efficiently, being more demanding in terms of computational resources.

5.4.6 Application of the ARIMA model

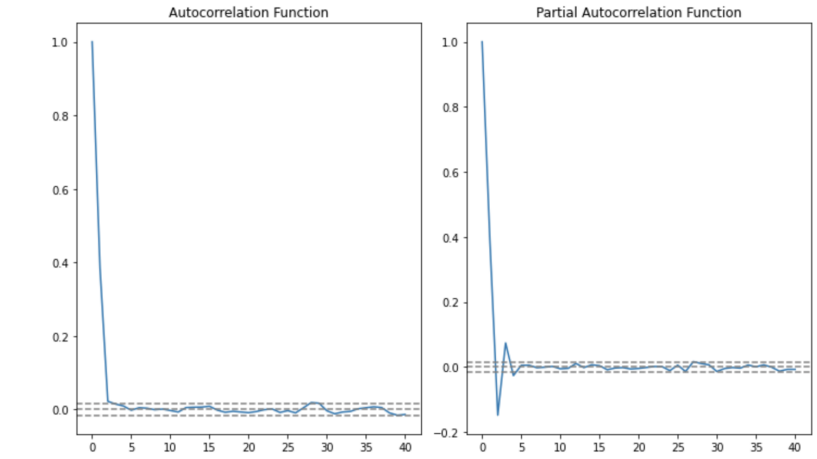


Figure 5.31: ACF and PACF of complete Bolus data

Another technique applied on the data was ARIMA. In this case, the time series were initially predicted individually. Specifically, the first data used and analyzed were for insulin boluses. At this initial stage, all data were considered, and preliminary analysis was performed to ensure the series' stationarity, a necessary condition for prediction models. Stationarity was verified by the Dickey-Fuller test, checking the p-value. If the p-value is below the significance level (x=0.05), one can reject the null hypothesis and conclude that the series is stationary. Using the statsmodels library, the test returned a p-value tending to 0, with an order less than 10e-8, leading to two conclusions: the series is stationary, and the D parameter of the ARIMA model can be set to 0 since the data do not need to be integrated. It remains to determine the P and Q parameters of the ARIMA model. We relied on two graphs: the ACF (autocorrelation) and the PACF (partial autocorrelation). Again, a script was created with the statsmodels library, and the results show that both graphs decreased toward the value 2. Therefore, the model is an ARIMA(2,0,2) or ARMA(2,2). The next step is to implement the model and analyze the results. The same analyses were performed on the glucose sensor (SensorGlucose) and ISIG datasets. The last 800 samples of the datasets were used for the test set, out of about 15,000 total. The results of the predictions can be seen in the corresponding graphs 5.32 5.33 5.34

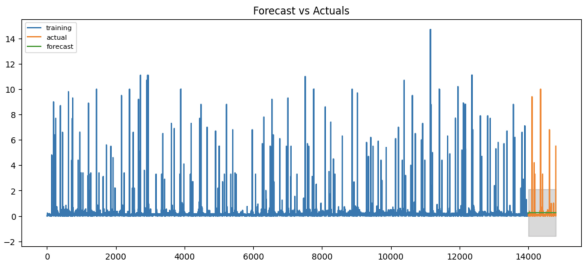


Figure 5.32: Bolus prediction using ARIMA

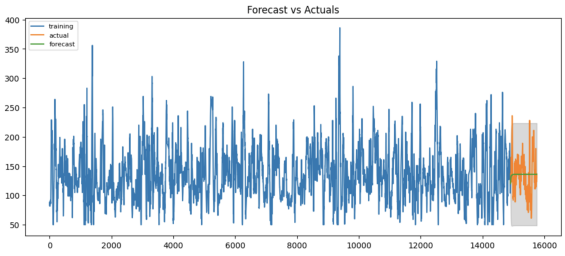


Figure 5.33: SensorGlucose prediction using ARIMA

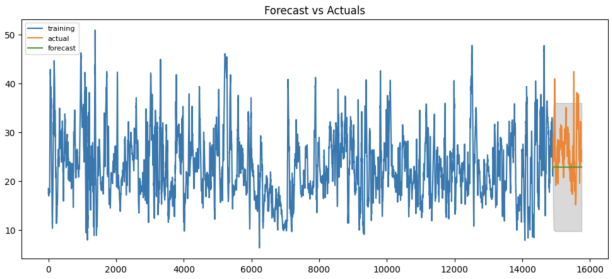


Figure 5.34: ISIG prediction using ARIMA

The results obtained will be presented in tables and by reporting all measurements for a complete view. From the analysis of the graphs, it can already be seen that the results are not particularly promising, as the predictions remain constant, generating a linear graph with no change over time. Tables 5.6, 5.7, 5.8.

Table 5.6: Bolus

Index	Value	Forecast
14000	0.100	0.203
14001	0.100	0.255
14002	0.100	0.258
...	...	...
14797	0.100	0.262
14798	0.000	0.265
14799	5.500	0.262

Table 5.7: Sensor Glucose

Index	Value	Forecast
14901	120.0	120.251
14902	123.0	121.527
14903	122.0	122.660
...	...	...
15749	137.0	136.021
15750	137.0	136.021
15751	135.0	136.021

Table 5.8: ISIG

Index	Value	Forecast
14901	22.72	22.555
14902	23.08	22.593
14903	22.99	22.629
...	...	...
15749	26.376	22.866
15750	26.12	22.866
15751	25.62	22.866

Subsequently, starting from the data, the various metrics were calculated. The results are shown in Table 5.9.

Next, the goal was to examine the data by trying to make predictions over shorter time intervals, to observe how the ARIMA algorithm performs with a reduced data set. It was therefore decided to

Table 5.9: Forecast Metrics

Type	SG	ISIG	Bolus
ME	3.31	-3.18	0.048
MAPE	0.22	0.16	inf
RSME	32.86	6.01	0.84

resample from sampling every 5 minutes to daily sampling, thus reducing the number of data from 14,000 to 115. For this experimental analysis, only SensorGlucose and ISIG data were considered, as since they were collected in the same time interval, they were easier to fit. After generating the two graphs, the results can be seen in Figures 5.35 and 5.36.

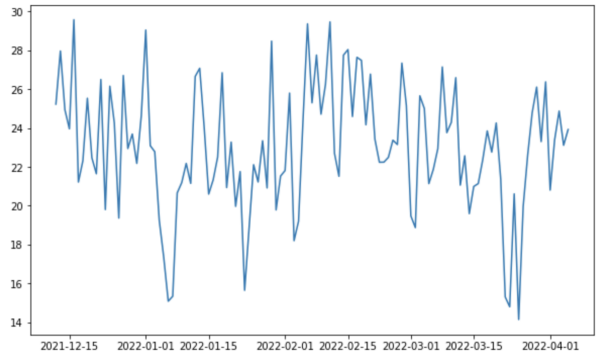


Figure 5.35: ISIG dataset after applying the re-sampling technique

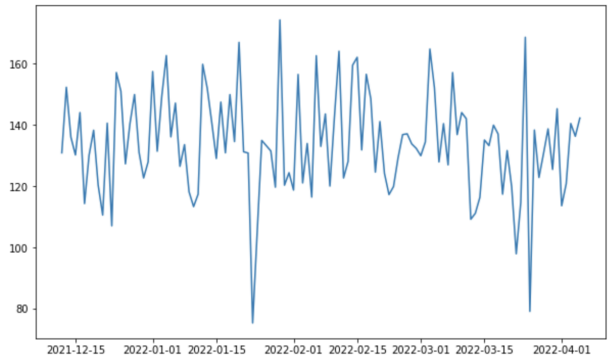


Figure 5.36: SensorGlucose dataset after applying the resampling technique

The Dickey-Fuller test was run again, and the graphs of the ACF and PCF were examined. After properly setting the parameters for the ARIMA algorithm, the dataset was divided into two parts: a training set containing 100 samples and a test set with 14 samples. From the results obtained, it is evident that accurate prediction has not been achieved. However, looking at the graphs, it can be seen that the predictions seem to be starting to better follow the general behavior of the original graph. The predictions are illustrated in the following figures 5.37, 5.38

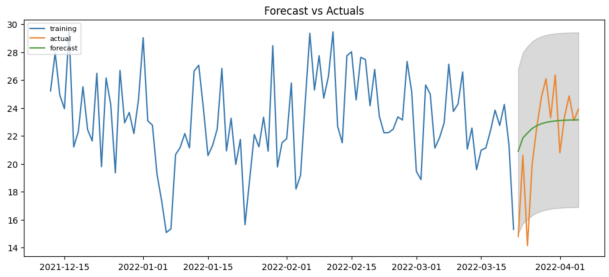


Figure 5.37: ISIG data prediction result using resampled data with ARIMA algorithm: with the train set in blue, the prediction in green, the test set in orange, and the confidence interval in gray..

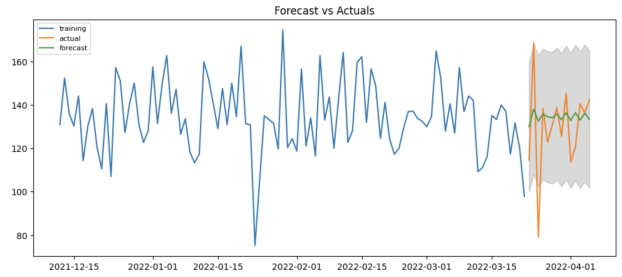


Figure 5.38: SensorGlucose data prediction result using resampled data with ARIMA algorithm: with the train set in blue, the prediction in green, the test set in orange, and the confidence interval in gray.

Moving on to the analysis in tabular form, the results are shown in Tables 5.10, 5.11, 5.12.

Table 5.10: ISIG after resample

Index	Value	Forecast
1	14.80	20.91
2	20.61	21.85
3	14.14	22.21
...	...	...
12	24.87	23.14
13	23.11	23.15
14	23.92	23.16

Table 5.11: Sensor Glucose after resample

Index	Value	Forecast
1	114.66	129.89
2	168.63	137.99
3	79.05	132.43
...	...	...
12	140.45	132.90
13	136.28	136.11
14	142.23	133.31

Table 5.12: Metrics after resample

Type	SG	ISIG
ME	4.59	0.65
MAPE	0.12	0.12
RSME	19.02	3.20

With these changes, improvements on both MAPE and RSME were highlighted.

5.4.7 Application of SARIMA and VARIMA models

In this section, we decided to explore two variants of the ARIMA algorithm: SARIMA and VARIMA. The SARIMA model adds four parameters to handle seasonality [194]. Since the data covers five months, we used five seasons for this test run on SensorGlucose data. As with ARIMA, we used the Statsmodel library in Python to implement SARIMA, and the results obtained were ME = 1.89, MAPE = 0.11, and RSME = 19.82. Although the results were similar to those of ARIMA, the downside was the long processing time required, which is why the SARIMA model was not considered further for the time being. Next, we tested the VARIMA [195] algorithm, using SensorGlucose and ISIG data as input. Before proceeding, we checked feasibility through the

Granger causality test provided by the Statsmodel library, which was successful. We then applied the VAR model, which proved the best after several tests based on the AIC values. Table 5.13 and Figure 5.39 show the forecasts promptly plus the measurement of errors.

Table 5.13: Recalculation of the metrics Mean Error (ME), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE) using SARIMA and VARIMA algorithms

Type	SG	ISIG
ME	1.89	-1.21
MAPE	0.11	0.11
RSME	18.18	3.83

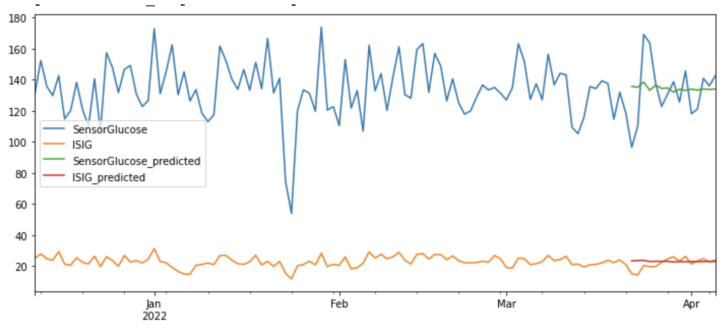


Figure 5.39: Comparison of actual test set data with predicted data for both ISIG and SensorGlucose using the VARMA algorithm

5.4.8 Defining different local, global, federated architectures

.

Finally, the last phase of the analysis, after finding that deep learning is the technique that provides the most accurate results, led to the definition of specific neural network architectures. These architectures were designed to further improve the effectiveness of treating diabetes in a cloud-edge context. The neural networks adopted for this study include several configurations optimized for processing data from medical sensors and predicting glucose levels. The models used were as follows:

- **LSTM (Long Short-Term Memory)**: These networks are particularly effective in processing time series because of their ability to remember information for long periods. They have been used to predict fluctuations in long-term glucose levels.
- **CNN (Convolutional Neural Network)**: Although commonly associated with image processing, CNNs can be effectively applied to time series by capturing spatial relationships between sequential data.

- **GRU (Gated Recurrent Unit)**: Similar to LSTMs but more computationally efficient, GRUs were adopted for their trade-off between computational capacity and performance.
- **RNN (Recurrent Neural Network)**: Traditional RNNs have been used to evaluate their effectiveness in capturing short-term dependencies in time series data.
- **MLP (Multi-Layer Perceptron)**: These feed-forward neural network models have been used for basic comparisons and benchmarking evaluations of other neural architectures.

On the other hand, as far as the architectures are concerned, there are three different ones, which we will see later in detail specifically they are:

- **Federated**
- **Global**
- **Local**

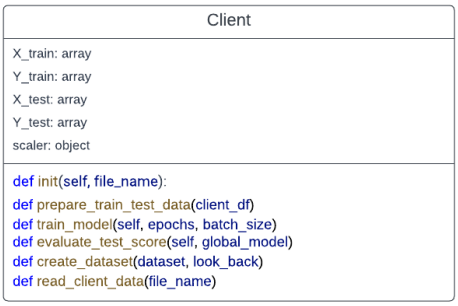
Considering 20 different clients, 3 different model types, and 2 data formats (resampled and unresampled), the number of simulations performed is 600.

5.4.9 Architectures and implementation of the of different scenarios

In this phase, local, global, and federated scenarios were defined. All scenarios were first methodologically defined using component diagrams and sequences and subsequently implemented and simulated through scripts executed in controlled environments.

Local Scenario

The first scenario defined is the local scenario. In this case, the user does not need any type of server and the collaboration of multiple clients but trains his model on his data. The single-device approach is ideal for all those scenarios that require data privacy and security or simply faster response. In such an approach, therefore, the data never leaves the device. The user therefore always has full control over all his data and the training and analysis takes place directly on the local device and in case that device is suffieciently powerful it would also perform better in terms of speed since in such scenario we save the time related to data transmission. Thus, the component of the local approach is shown below:



.

Figure 5.40: Class diagram related to local approach , showing instantiated functions on just client side

Below, we discuss both the component and its functions. In this case, as already explained above, being a unique component, we do not need relations binding it with others. So the component is self-contained. The main functions and variables are:

- Xtrain, Ytrain, Xtest, Ytest: These array-type variables repetitively represent the training and testing dataset of features and targets.
- Scaler: The object normalizes the features before training and evaluating the model.
- prepareTrainTestData: Represents the function for preparing training and testing from a DataFrame .
- trainModel: It deals with training the model and represents the process in which it learns from the training data.
- evaluateTestScore: It is used to evaluate the model's effectiveness.
- createDataset: after reading the data, it takes care of placing it within a formatted and ready-to-use dataframe. Also, by lookback parameter, we can indicate how many previous time steps to consider for each data point.
- readClientData: This reads client data from the data source the model will use for training and testing.

Global Scenario

The second scenario we are going to define is a global approach. In this case, we have a collaborative approach in which we do not worry about data sharing. Such a scenario can be useful in contexts where data sharing is not a problem for the user to remember; however, in any case,

authorization from the data subject should always be requested before proceeding with sharing. In this case we have a 2-tier type of architecture plus specifically a client/server type system. The centralized system provides a whole range of advantages over the local one in particular:

- Cost efficiency: maintaining or upgrading resources in a centralized environment drastically reduces operational costs because it's possible to cover several devices with one hardware compared to a distributed or local approach.
- Ease of backup and recovery: by owning all data in the central server, backups and recovery of systems become extremely easy owing to not requiring synchronization of various devices.
- Scalability: certainly one of the most important factors that leads many users to choose a centralized approach is scalability [196]. In particular, a centralized system that uses innovative technologies allows resources to be scaled and dynamic according to what is needed at that precise moment.

Below is shown the component diagram of a centralized type architecture where multiple clients are connected to a single server.

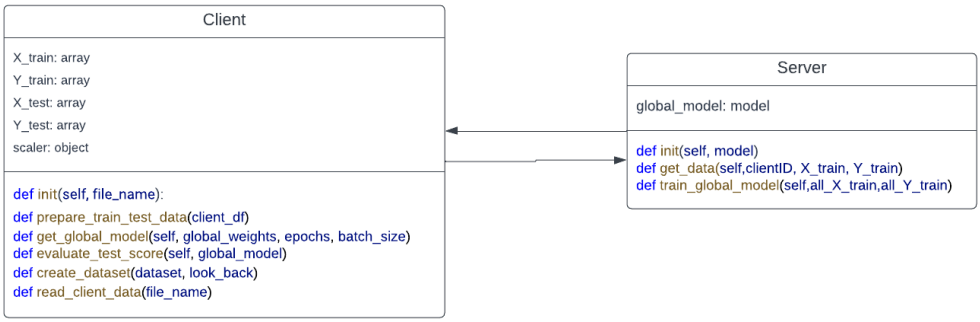


Figure 5.41: Class diagram related to global learning model, showing instantiated functions on both the client and server sides

In the following, we comment on all the functions and variables within the two components except for Xtrain, Ytrain, Xtest, Ytest, scaler, prepareTrainTestData, readClientData, and createDataset, which have already been discussed within the local approach and play the same roles in this global. Thus, the functions and variables that differ in the global versus the local approach are:

- globalModel: represents the global model updated to the last training.
- getGlobalModel: gets the global model from the server and allows training to be performed on the client for a specified number of epochs and with a certain batch size.

- getData: this function receives data from all individual clients. The operation can be of diverse types, which we will discuss later. In general, there are two approaches: a trigger that is activated once all the data has been sent from all the clients; in this case, it is assumed that the server knows how many clients need to send data. Or even a timer that schedules a time limit within which the various clients can send their data.
- trainGlobalModel: after receiving data from all clients, this approach takes care of training the global model

The scheduling or order of execution of the various functions is formally represented through the sequence diagram:

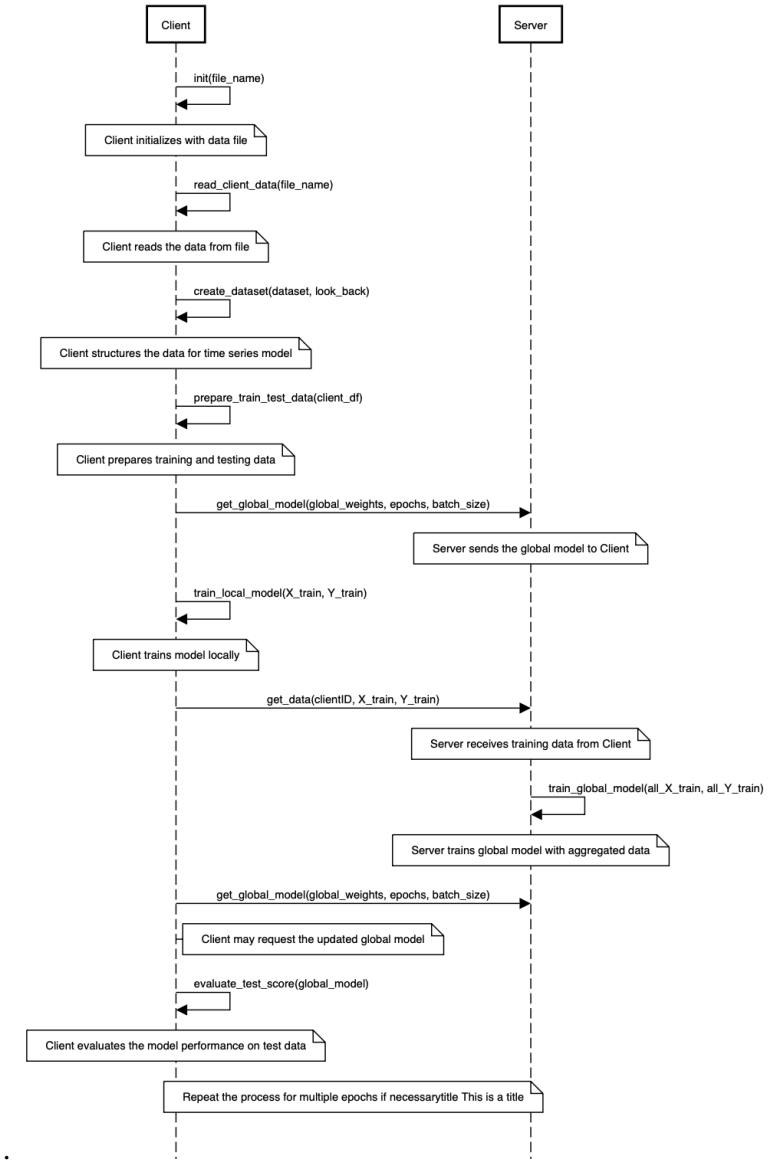


Figure 5.42: Sequence diagram of global scenarios for message and data exchange between client and server

Basically, as can be seen from the sequence, the client takes care of capturing and formatting the data; once this is done, it sends everything to the server. After some time or after receiving the data from the predetermined number of clients, the server performs the model train on the data and returns it to the clients. These use the model on their data to predict future values.

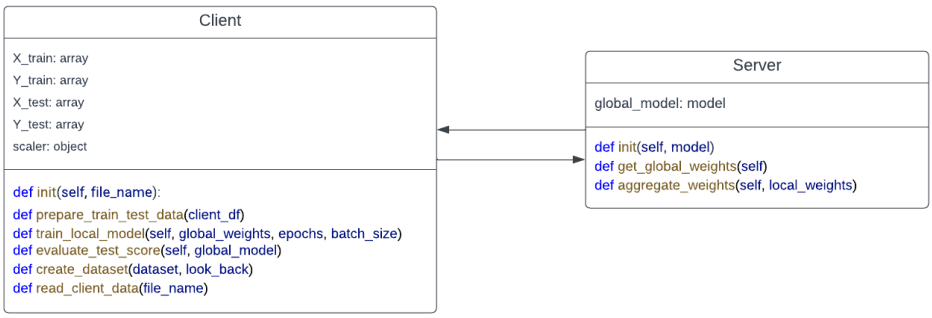
Federated Scenario

The third scenario is a distributed one. In this case, there is always a central server present, but training is done through federated learning in a distributed manner, and the server's only task is to

handle the sending, receiving, and aggregation of the weights. The advantages of using a federated approach instead of a centralized one are:

- Privacy and Data Security: In an approach such as federated learning, data never leaves client devices. This minimizes risks on data dissemination and consequently also improves compliance with data privacy regulations (such as GDPR)
- Reduction of Bias in the Data: Compared to a premises-based approach, while maintaining the same security and privacy by training the model on a wide variety of data from different users and devices, you can reduce the bias in the data. The result is usually a fairer model.
- Efficient Use of Resources: A federated learning approach reduces the need for expensive centralized computing infrastructure by moving computation to the edge side.

The component diagram of a distributed federated architecture is shown below:



.

Figure 5.43: Class diagram related to federated learning, showing instantiated functions on both the client and server sides

Again, as was the case with the other architectures, we do not dwell on commenting on the functions already seen in previous architectures but focus only on those introduced in this scenario, which are:

- aggregateWeight which has the task, once a required number of weights have been collected, of aggregating them. As we have already seen within the state of the artthe aggregation, this operation can be done in different ways depending on the needs.
- getGlobalWeight, which takes care of requesting the released weights from the clients after training the individual models

The scheduling or logical order of the functions is shown in the sequence diagram:

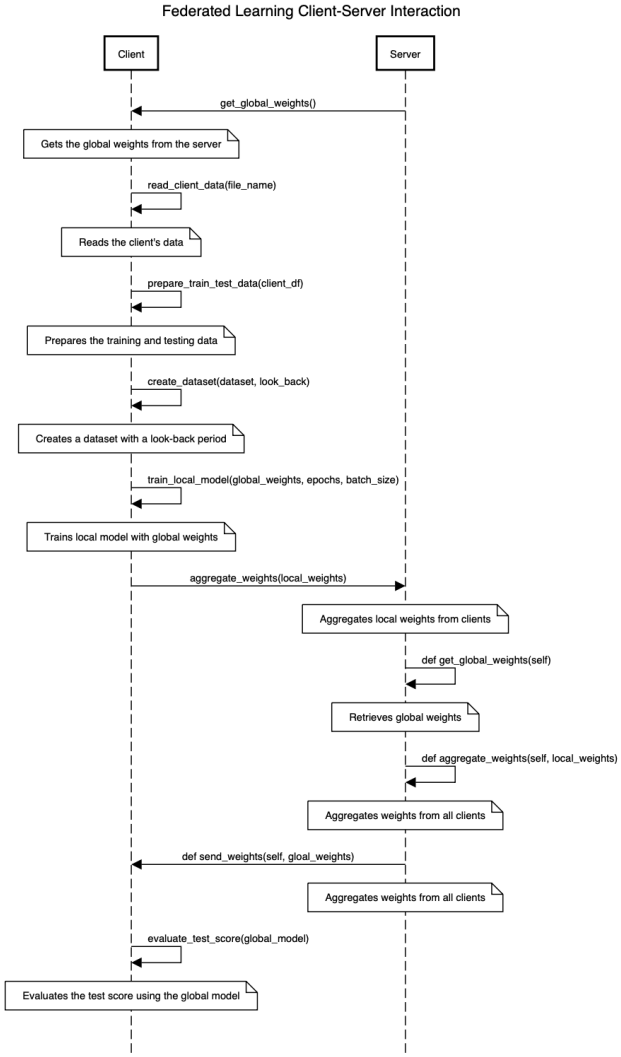


Figure 5.44: Sequence diagram of federated scenarios for message and data exchange between client and server

The idea is that after each client has train the model on its data, it sends the weights to the server; the server, once it has received either all the weights or several weights, it proceeds to aggregate them; after doing so, it sends the aggregated weights to the client which again runs the model on its data, after several iterations the model will be ready to be used.

5.4.10 Model Implementation and Evaluation

For the implementation phase, Python was chosen as the programming language since it is the most complete one among all in terms of libraries concerning AI. Google Coolab was chosen as the development environment, and the main libraries used for the development of the different

approaches were: numpy and pandas for the management of data in CSV format and for all the data formatting mentioned above; matplotlib for data visualization and representation;sklearn and Keras for the development of algorithms in AI. The architectures, global federated and local, initially defined in an agnostic manner, were then successively developed, and the different models LSTM, CNN, RNN, MPL, and GRU were used for each architecture. To that code were then passed two times the data: the first time without resampling and the second time n sampling to a 1H. One hour was chosen, and not others, due to preliminary experimental results that showed such resampling to be most suitable for increasing the speed, smoothing the data, and keeping the information intact. Performance evaluations will occur in 2 different types:

- **Comparison of Models**: Given the same architecture, an attempt was made to understand which model performed best regarding results and time.
- **Comparison of Architectures**: Given the same model, an attempt was made to find out for each client which architecture was the best performing by comparing both the information in terms of accuracy and time.

As for the parameters instead, that will be considered, mainly two evaluations have been implemented, which are:

- **Execution time**: the time required for the trains of the various models, which changes depending on the architecture and is summarized in this pseudocode:

```
BEGIN
    st_fed = getCurrentTime()
    FOR each c IN list_of_clients
        FOR round FROM 1 TO 3
            executeTraining(c)
        END FOR
    END FOR
    end_t_fed = getCurrentTime()
    exe_t_fed = end_t_fed - st_fed
END

BEGIN
    st_g = getCurrentTime()
    executeTraining(c_model)
    end_t_g = getCurrentTime()
```

```
    exe_t_g = end_t_g - st_g
END

BEGIN
    st_l = getCurrentTime()
    FOR each model IN list_of_l_models
        executeTraining(l_model)
    END FOR
    end_t_l = getCurrentTime()
    exe_t_l = end_t_l - st_l
END
```

- **Correctness of result**: obtained by calculating the MSE, which is obtained from the formula

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Graphs, tables, and mean and maximum values were drawn up for each of the measurements taken, with the goal of getting as holistic a view as possible of the capabilities of the architectures and models. The first comparison that has been made is between the models in the figure 5.46. Here, the graph shows us a microscopic visualization of the performance of the results of each model on individual clients. This is done for the global, local, and federated architectures for both normal and resampled data. From this type of visualization, it is quite evident that in the federated case, the performance of the models is equal in both data (resampled and unresampled). The same can be said for the performance within the global approach with the unresampled data. On the other hand, about the last three case histories, namely, ”global resampled”, ”local” and ”local resampled” visually, there are differences between the models. We then turn to a macroscopic view to visualize these results more clearly. Fig 5.51:

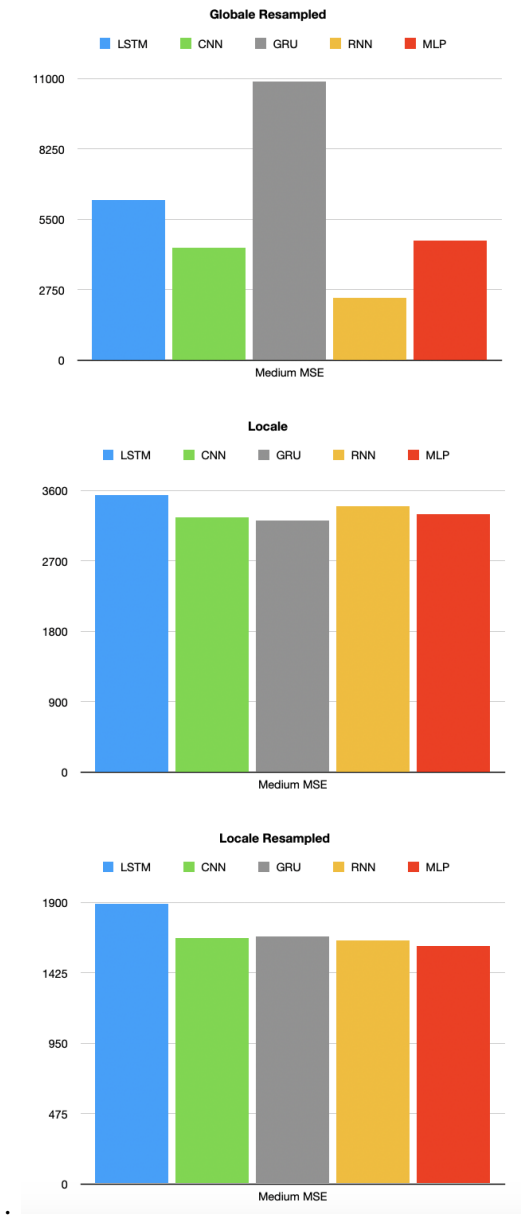


Figure 5.45: Comparison of average client RSME performance among different deep learning algorithms in the three different scenarios: local, global, and federated

From this view, we can well see that although for individual clients in some circumstances, there were peaks with some models than others in a global view, the measurements, even if with small oscillations in the local case, are quite similar, except for LSTM that have a slightly larger error. On the other hand as regards the global resampled case in this case the GRU has performed much worse than the other models, probably because of its nature unsuitable for a context of this type. In contrast, the RRN returned very good results compared to the other models in terms of the correctness. The second analysis that is intended to be done is on architectures. To pursue the latter, we will set a single model for each client and evaluate the architecture for each. The graphs

shown will have the clients on the x-axis and the MSE index on the y-axis.

Figure 5.46: Compare different models for each architecture and each client with and without resampled data in term of RSME

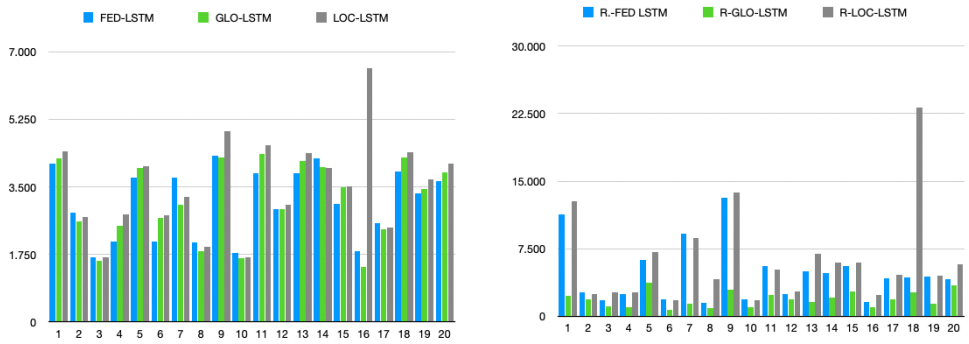


Figure 5.47: Focus comparing results with LSTM algorithm in the three different scenarios: local, global and federated.

In the case of the LSTM model Fig 5.47, we notice that all three approaches equalize in the first formatting of the data,. The one that returns fewer errors within the evaluation is the federated approach. With reformatted data, on the other hand, the local approach turns out to be significantly better. However, the federated approach always performs better than the local approach.

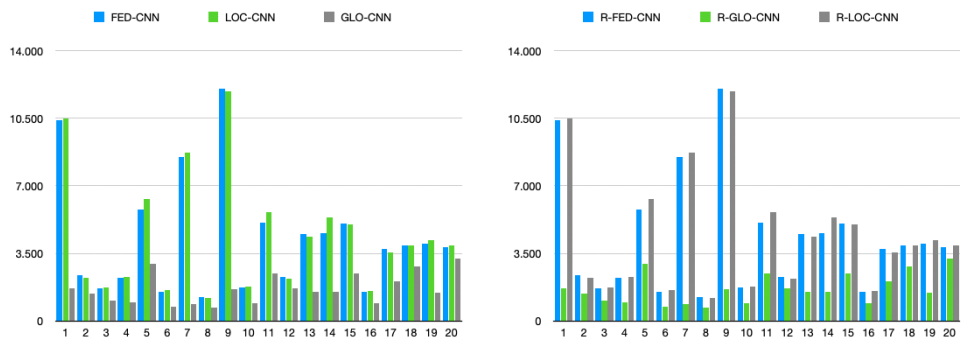


Figure 5.48: Focus comparing results with CNN algorithm in the three different scenarios: local, global and federated.

In the case of the CNN network, on the other hand, Fig. 5.48, we have a slightly different behavior; in fact, in the first case (not resampled), the local performance completely collapses while the local and global performance on the whole equalizes. In the second case, instead with the reformatted data again, the global algorithm returns markedly higher performance than the global and local approaches

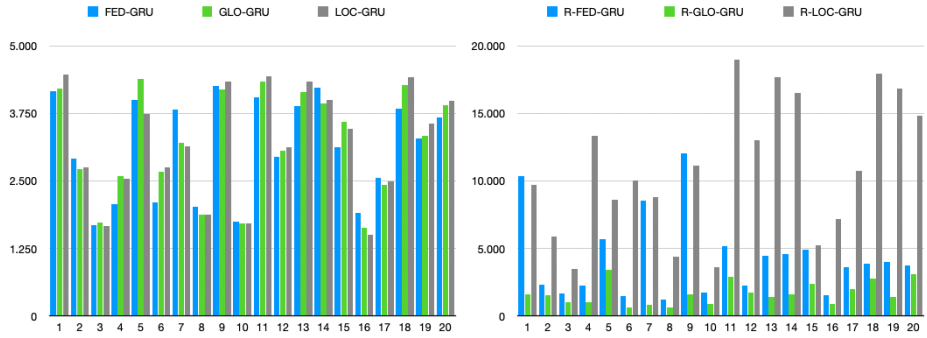


Figure 5.49: Focus comparing results with GRU algorithm in the three different scenarios: local, global and federated.

Using GRU in the first case, all three architectures return similar errors. Interesting instead are the measurements taken with the resampled case in which, although again the global approach returned the best results, the federated approach, in this case, is still quite satisfactory. Once again, however, the local approach show a complete decay in terms of correctness of the result

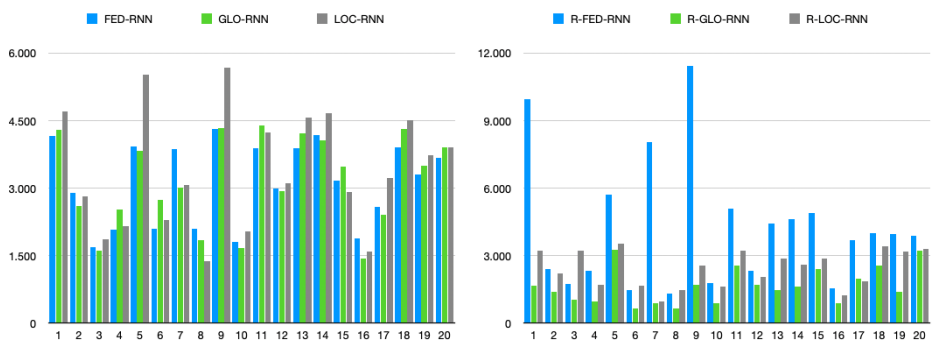


Figure 5.50: Focus comparing results with RNN algorithm in the three different scenarios: local, global and federated.

With the RNN network, on the other hand, in the first case, once again, the measurements are equalized, but this time, we see a clear elevation of error by the local approach, which nevertheless improves in the case of resampled data and is comparable to the performance of the global approach. In this case, however, the federated approach shows a drop in performance by producing many more errors than the other two architectures.

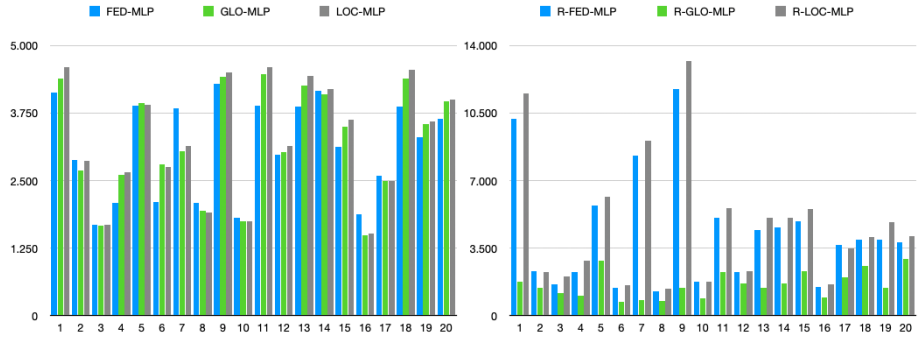


Figure 5.51: Focus comparing results with MPL algorithm in the three different scenarios: local, global and federated.

Finally, we have the MLP case in which the performances are very similar to those of the LSTM, except that in the case of resampled data, the federated and local approaches return much larger errors. Below, to get a view of the measurements, numerically and not only graphically, we report the results for each model of the first five clients for each of the 2 data. We do not report the whole dataset of measures for reasons of brevity 5.14.

These data show that although the proposed approach produces better results overall, in many cases the results obtained by the other two approaches are very similar. However, taking a broader and more general perspective, we can look at the average values of each measure for the two types of data.

	FED	GLO	LOC
LSTM	3589,41	3664,01	4217,88
CNN	3243,82	3359,29	3481,69
GRU	3679,50	4225,90	3849,76
RNN	3283,23	3371,08	3657,96
MLP	3680,65	4254,76	3801,86

Table 5.15: Performance comparison of model architectures on NOT resampled data

	FED	GLO	LOC
LSTM	4747,04	2064,73	6836,456
CNN	4345,43	1880,36	4443,22
GRU	4305,18	1870,11	12656,84
RNN	4303,70	1847,93	2563,73
MLP	4237,27	1606,34	4674,11

Table 5.16: Performance comparison of model architectures on resampled data

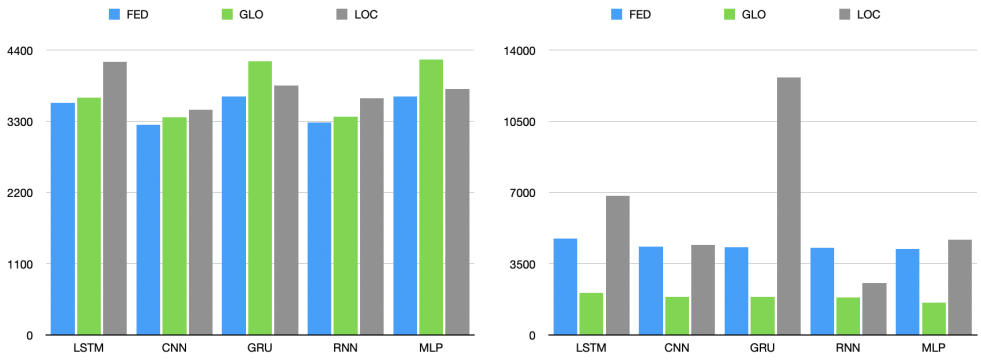


Figure 5.52: Comparison of the performance of the average client of both model architectures on resampled and unresampled data

In this case, thanks to both a restricted and tabular graphical form, we have a much clearer view of our measurements, and we can say that in the first case, the federated learning performs slightly better from the point of view of correctness of the data respect to the global model. In the second case, on the other hand, the global approach returns significantly better values. However, the federated approach continues to be superior to the local model. On the other hand, regarding the time analysis, in this case, what has been done is to measure the execution time at equal

computing power to understand which application performs better in terms of time. Obviously, in a real-world approach, edge devices possess less power than global devices; however, to make the measurements unambiguous, it was thought in this simulation to make one parameter (computing power) fixed and measure the variations in time. After applying the methodology defined above, the average results for each client are shown in tabular and graphical form.

Table 5.18: Not Resampled

Model	FED	GLO	LOC
LSTM	7.05	87.64	4.35
CNN	2.50	14.48	1.82
GRU	7.43	84.59	5.15
RNN	4.24	42.12	2.86
MLP	2.31	12.29	2.08

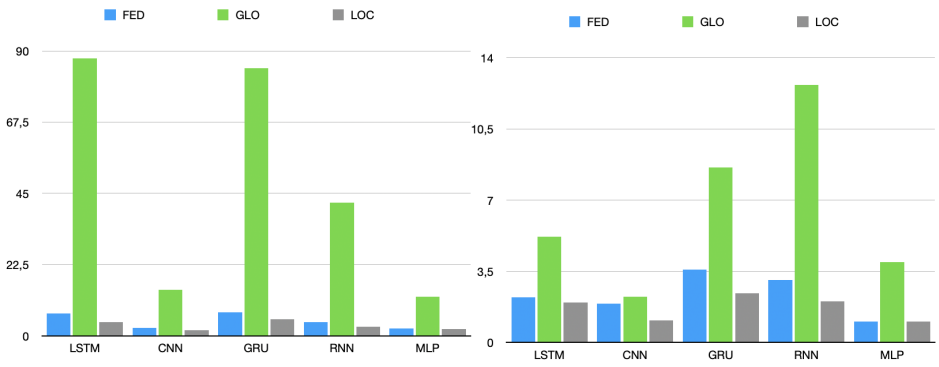


Figure 5.53: Comparison of execution speeds at the same average client computing power of all model architectures on resampled and unresampled data

Here, the situation is completely different; in fact, the global approach shows a decay of performance in terms of time and returns measures that are not comparable with the federated and local approaches. On the other hand, about the federated approach, the latter, in terms of time, is slightly higher than the local one. But we expected this; in fact, we must consider the three rounds done and the time defined by the slower model. But the performance in terms of time, although higher, is completely acceptable and considered a clear improvement in terms of performance, especially in some models. Finally, to get a complete view, the best prediction made among the various clients and models are shown below for each architecture Fig 5.54 5.55 5.56.

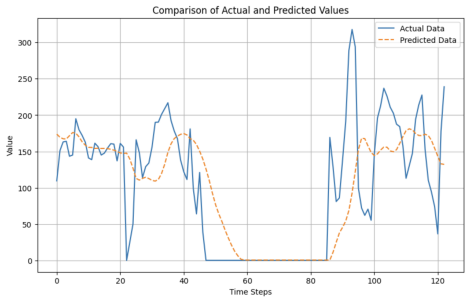


Figure 5.54: The chart shows a portion of the prediction obtained using the federated LSTM approach, highlighting how the model closely follows the trend of the real values

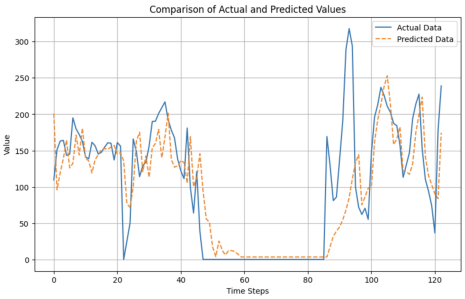


Figure 5.55: The best performance for the global approach was achieved using the CNN model, which closely captured the trend of the real values

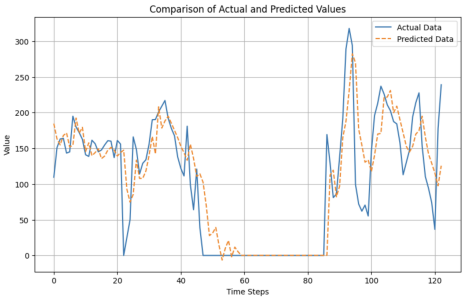


Figure 5.56: The local approach with the MLP model demonstrated solid results, accurately following the trend of the real values

5.4.11 Testing Real and Simulated Architectures

After testing the different models and architectures by harnessing the computational power of Google Colab to obtain measurements as close to reality as possible, the various scenarios (local federated and global federated) were first tested in a real environment and then simulated in order to increase the scale of the simulation with the significant increase in the number of users. The hardware components used to run these simulations were:

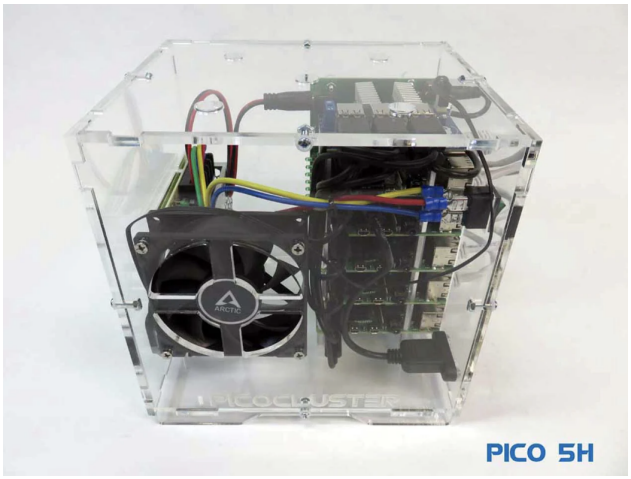


Figure 5.57: PicoCluster hardware component with which the various clusters were simulated

- PicoCluster 5H: Designed to cluster 5 Raspberry Pi 4 boards together to run a wide range of software, from web servers and databases to Big Data and NoSQL platforms. One of the most popular uses is with container orchestration tools like Docker Swarm or Kubernetes. In this configuration, each individual board represents a client. The features of the board are a 1.5 GHz Quad-Core ARM Cortex-A72 processor, 8 GB RAM and 2 GB archivation Fig. 5.57
- Proximos Server: part of a proxmos server was requested and used. Specifically, the partition used has 20 GB of RAM and over 100 GB of storage.

For the relization of the code execution and resend requests so as not to disrupt the existing code too much, flusk was chosen as the framework. Through a send/get mechanism the different scenarios were then implmented. Having measurements available to simulate 20 different clients and the minicluster being composed of 5 boards, the logic by which the simulations for each scenario were done was as follows: for the local scenario in groups of 5, clients were run to compute the model on their own data. For each of the 5 clients an average was run to have a measure to compare with the other two scenarios: federated and global. For these two in fact, the simulations were done with 5 participants at a time. The measurements made were performed using the LSTM algorithm and with the servers and the 5 picoclusters connected via Ethernet cable to two separate networks. In the case of federated learning 3 round was done. The following is both the full tabulation with associated graph 5.58 that measures the various clients and the table in which these were aggregated by five and related to the other two scenarios global and federated Table 5.20 Fig. 5.59.

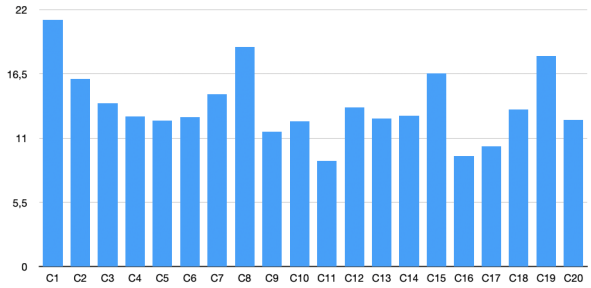


Figure 5.58: Graph associated with the measurements shown in Table 5.19

Clients	**Time**
Client 1	21,14
Client 2	16,07
Client 3	13,98
Client 4	12,86
Client 5	12,51
Client 6	12,80
Client 7	14,76
Client 8	18,78
Client 9	11,53
Client 10	12,46
Client 11	9,04
Client 12	13,65
Client 13	12,67
Client 14	12,90
Client 15	16,56
Client 16	9,47
Client 17	10,32
Client 18	13,42
Client 19	18,03
Client 20	12,54

Table 5.19: Simulation in a real scenario using Picocluster and Proximos server train and test 20 different clients trained locally

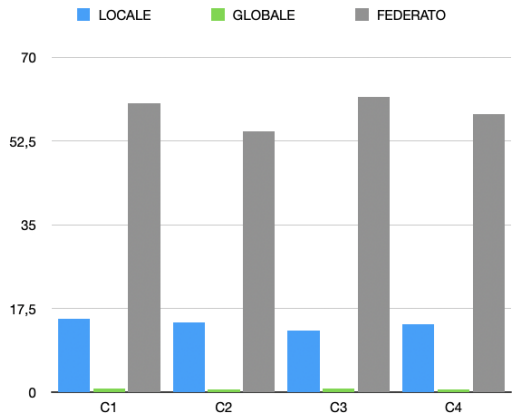


Figure 5.59: Graph associated with the measurements shown in Table 5.20

UserBase	Local	Global	Federated
UB 1-5	15,31	1,73	60,32
UB 6-10	14,66	2,34	54,49
UB 11-15	12,96	1,82	61,64
UB 16-20	14,17	1,69	58,02

Table 5.20: Simulation in a real scenario using Picocluster and Proximos server train and test with 4 different userbases with comparison of different scenarios including local, federated and global

As we expected, the global approach provided better results in terms of time due to the huge computational capacity of the server. In fact, in the case of the global approach, execution times are significantly reduced. In contrast, between the local and federated approaches, the federated shows lower performance. This was to be expected for two main reasons: the first is that the duration of each round is determined by the slowest client, since all clients must wait for each to complete before receiving aggregate weights. The second reason is related to the fact that three rounds of computation take longer than training and local testing on all the data. However, although the training phase may be longer, federated learning still performs better than the local approach. This is because although the time for training is longer, for testing on the other hand it is equivalent to local but with higher performance. Finally, the last test that has been performed is the stability of the dployed architecture on servers. This was posible through the Locust simulation tool. Specifically, differwnti clients making simultaneous requests were simulated and response delays from the server were measured. In detail, 3 scenarios 20 clients, 50 and 100 were simulated. The results are shown in Fig.

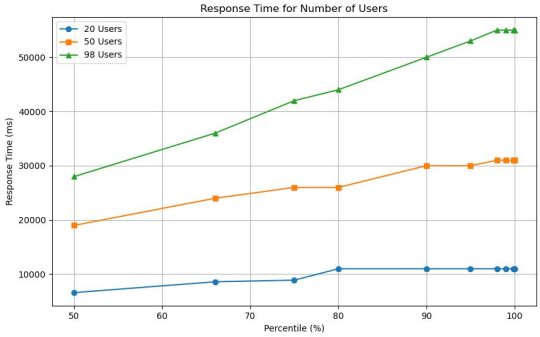


Figure 5.60: Simulation of response time in milliseconds for up to 100 users

The measurements indicate that the server responds adequately up to a certain number of users, but there is room for improvement to handle larger loads more efficiently. Implementing techniques such as load balancing, use of distributed caches, or adoption of microservice architectures can help optimize performance. In addition, the use of processing queues and a proactive monitoring system could reduce response times in peak situations. Certainly then, for proper deployment of the system, it will be necessary to implement these changes to ensure greater scalability and a better user experience even in high traffic scenarios.

5.5 Conclusion

This work explored machine learning and deep learning techniques for predicting glucose levels in patients with type 1 diabetes. Using a range of machine learning algorithms, from classical regression models such as Linear Regression, Lasso and Ridge, to more sophisticated approaches such as LSTM, CNN and GRU neural networks, the study sought to provide accurate glycemic predictions based on data collected from real patients. These models were evaluated in different computational architectures, including local, global, and federated scenarios, to analyze their performance and scalability in real-world applications. The main contributions of this study include a detailed comparative analysis of machine learning and deep learning models for predicting blood glucose levels. The LSTM model demonstrated the best results for glucose prediction, outperforming traditional machine learning models in terms of RMSE, R^2 and MAPE. The ARIMA, SARIMA and VARIMA models were also tested, but the results obtained were not as promising, especially for long-term predictions, due to their tendency to produce constant values in some cases. In addition, the implementation and testing of federated learning represents a significant advance, highlighting its potential in maintaining data privacy while allowing effective model training on multiple clients. The federated approach demonstrated good performance, particularly in preserving accuracy among distributed devices, although training times were found to be longer than for global and

local models. This trade-off between privacy, computational efficiency, and accuracy was examined in detail through real simulations with PicoCluster hardware and Proximos servers, simulating a real scalable environment. As for future developments these certainly could concern implementing different patters to improve communication such as client selectors or a hierarchical approach with a view to on the one hand improve the performance of federated learning on the other hand avoid data reconstruction attacks from the model. In addition, it would be interesting to use hybrid approaches by dividing patient characteristics that can be shared from those that cannot be shared in order to better calibrate the sceelection of model participants. Finally, it might be interesting to increase the amount of data with the goal of including more patients and thus generalizing the model even more.

Table 5.14: Results of the top 5 clients in terms of performance by comparing them for different architectures and models

Client	**FED-LSTM**	**Client**	**GLO-LSTM**	**Client**	**LOC-LSTM**
3	1681.03	16	1428.29	3	1669.62
10	1789.99	3	1588.56	10	1677.49
16	1827.99	10	1653.59	8	1937.82
8	2062.81	8	1835.47	17	2447.28
4	2071.02	17	2391.52	2	2725.61
Client	**FED-CNN**	**Client**	**GLO-CNN**	**Client**	**LOC-CNN**
3	1706.47	16	1439.86	16	1522.96
10	1803.67	3	1605.15	3	1643.73
16	1857.31	10	1669.86	10	1723.92
8	2067.54	8	1842.44	8	1912.97
6	2084.17	17	2403.13	17	2487.27
Client	**FED-GRU**	**Client**	**GLO-GRU**	**Client**	**LOC-GRU**
3	1682.55	16	1636.36	16	1499.90
10	1750.42	10	1711.07	3	1667.48
16	1909.86	3	1728.75	10	1708.71
8	2016.65	8	1877.89	8	1871.77
4	2074.21	17	2427.61	17	2488.61
Client	**FED-RNN**	**Client**	**GLO-RNN**	**Client**	**LOC-RNN**
3	1688.68	16	1439.77	8	1386.88
10	1807.19	3	1609.46	16	1599.64
16	1875.11	10	1678.15	3	1858.51
4	2082.71	8	1853.76	10	2036.99
6	2096.60	17	2418.35	4	2152.32
Client	**FED-MLP**	**Client**	**GLO-MLP**	**Client**	**LOC-MLP**
3	1687.76	16	1489.87	16	1512.91
10	1810.27	3	1664.92	3	1676.36
16	1874.09	10	1750.78	10	1747.47
4	2083.78	8	1933.52	8	1903.11
8	2093.53	17	2483.33	17	2493.70

Table 5.17: Preliminary time results using a single environment (Google Coolab) for the different models and architectures

Model	**FED**	**GLO**	**LOC**
LSTM	2.21	5.20	1.96
CNN	1.92	2.24	1.09
GRU	3.58	8.59	2.41
RNN	3.07	12.67	2.01
MLP	1.04	3.96	1.04

Chapter 6

Virtual And Augmented Reality

In recent years, the rapid advancement of technology has led to the convergence of solutions such as cloud computing and augmented reality, opening up new frontiers in human experiences. In this context, the cloud edge [197, 198] is a fundamental pillar, allowing data to be processed closer to end users and eliminating critical application latency. This work subbmitted at MDPI Journal examines the connection between cloud edge [199] and collaborative augmented reality. To this end, different architectures within the cloud continuum scenario for AR/VR environments were defined, implemented, and tested. These architectures were subsequently validated through real-world case study implementations and through the use of simulators to evaluate their scalability. All the work was entirely reported in this section.

6.1 Objectives and Methodology

The main objective of this work is to define different and valid reference architectures for deploying VR and AR applications on cloud continuum infrastructures. Remarkably, these architectures will be structured in such a way as to carry the computation load from the core to the edge respectively. The work was carried out as follows to achieve the previously described objectives. First, we defined three different architectures using UML diagrams [200]; in particular, we used the component diagrams to define the single component and sequence diagrams to show their interoperability. The next step was to determine the technologies that would implement the individual components of all architectures. Simultaneously, we developed three different applications to demonstrate the usefulness of these architectures in different contexts. Finally, the last phase involved simulation, using specific tools to validate these architectures, even with many users.

6.2 Architecture

In this section, different reference architectures will be shown. They are exposed in the following order: we start from a cloud center architecture in which almost all operations are done in the cloud and then gradually move the computational load to the edge. For each architecture, the scenarios and mechanisms of use will be exposed.

6.2.1 Centralized IoT System with Prevalent Processing in the Cloud

The first architecture deals with scanning real objects and their real-time reconstruction in virtual environments through sensors. The various components are represented through the following component diagram Fig. 6.1.

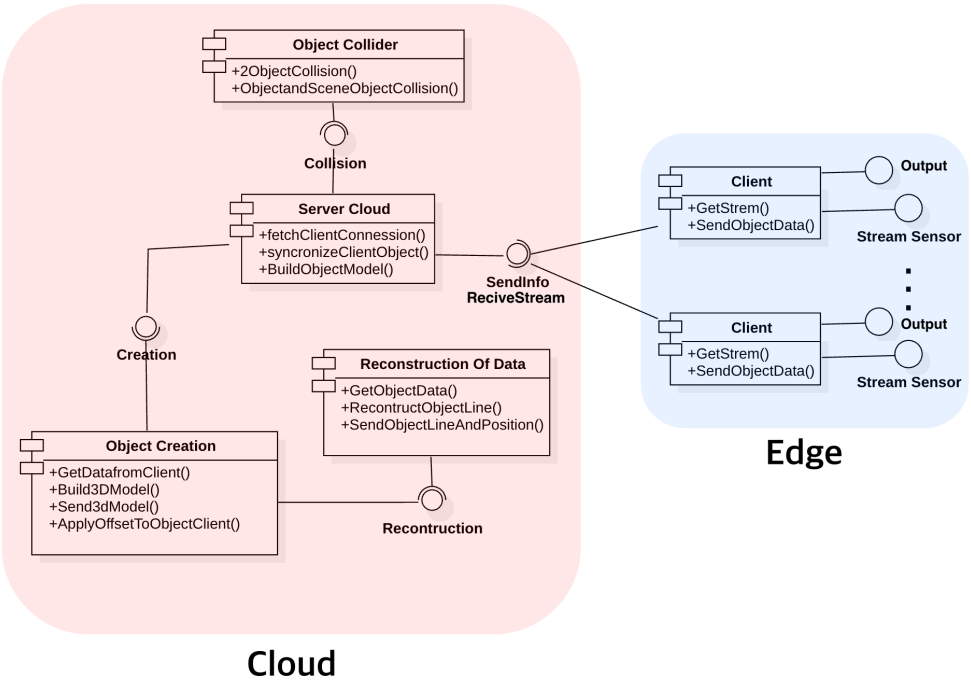


Figure 6.1: The figure shows the component diagram of the first reference architecture in which the entire computation is all shifted to the cloud.

The operation of the individual components is defined as follows:

- **Cloud Server**: The cloud server acts as a central point for managing multiplayer connections and processing the tracking data of the object to be recognized. It receives data from clients and routes it for scene processing and synchronization. It uses scalable and reliable cloud resources to ensure a stable connection and rapid data processing.

- **Client**: The client represents a new "*user*" and is responsible for connecting to the cloud server to access the scene shared with other users. It also sends and receives data relating to the object previously recognized by the sensors. The goal is to have a one-to-one movement between what the sensor detects and what the cloud reconstructs.

- **Object Collider**: This is the component responsible for detecting the movements of the user's object using the sensor. It captures the data relating to the position and orientation, then transmits them to the client for 3D processing and sending to the central server. This saving of the individual positions also allows events to be triggered when there is a collision between reconstructed objects and between a single reconstructed object and a single scene object.

- **Object Creation**: The processing component receives the object tracking data from the server and processes them by creating the associated 3D model the clients need, subsequently routing them via the server to the client.

- **Reconstruction Of Data**: to create the object, it is necessary first to reconstruct the object starting from the data that construct the sensors. A rule-based approach or machine and deep learning algorithms [92] can do this in many ways.

The sequence diagram 6.2 describes the interaction between these components and their workflow. The client component uses a sensor to capture external data sent to the cloud in real-time. The server uses the *Object Creation* and *Reconstruction Of Data* components to reconstruct the 3D object. In more detail, the Reconstruction of the Data component reconstructs the object of interest from the data, and the other is mapping it in 3D. For example, assuming you want to recognize a person's movements, the first component will identify the object in the data stream and map its movements, and the other will reconstruct the avatar based on the real person's movements. Furthermore, particular attention must be paid to the *ApplyOffsetToObjectClient* method Fig 6.3.

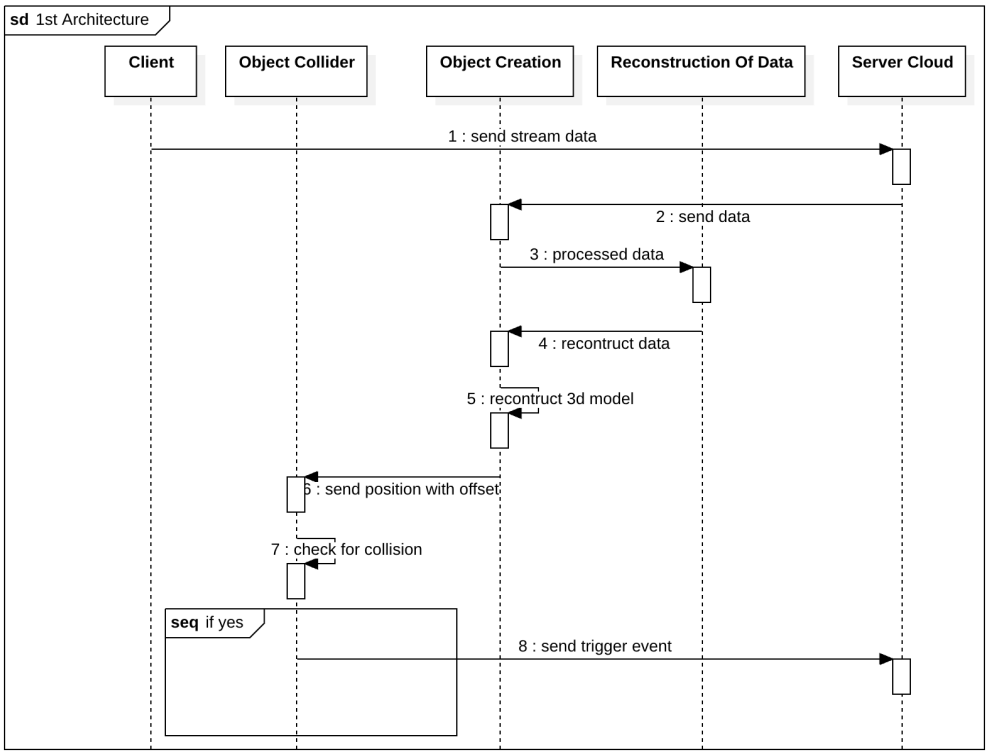


Figure 6.2: The diagram shows the flow of the first reference architecture to send data, reconstruct a 3D model, check collisions and trigger events, involving client, object creation, data reconstruction, and cloud server.

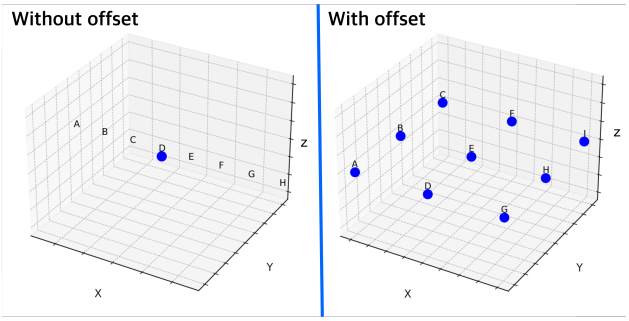


Figure 6.3: This figure allows you to visually understand what happens if an offset is not properly scheduled depending on the number of participants. In the first case, all the spawns are overlapped. In the second case, however, the spawns are perfectly ordered depending on the number of participants (in this specific case nine)

Depending on the number of clients connecting, this method allows assigning a starting offset to the single reconstruction in the virtual world. This is necessary to ensure that the spawn of 3D objects in VR is never overlapped.

6.2.2 Hybrid IoT Infrastructure with Balanced Processing between Cloud and Edge

In this second architecture there is the addition of some components that have been repositioned in an environment architecturally closer to IoT devices Fig.6.4.

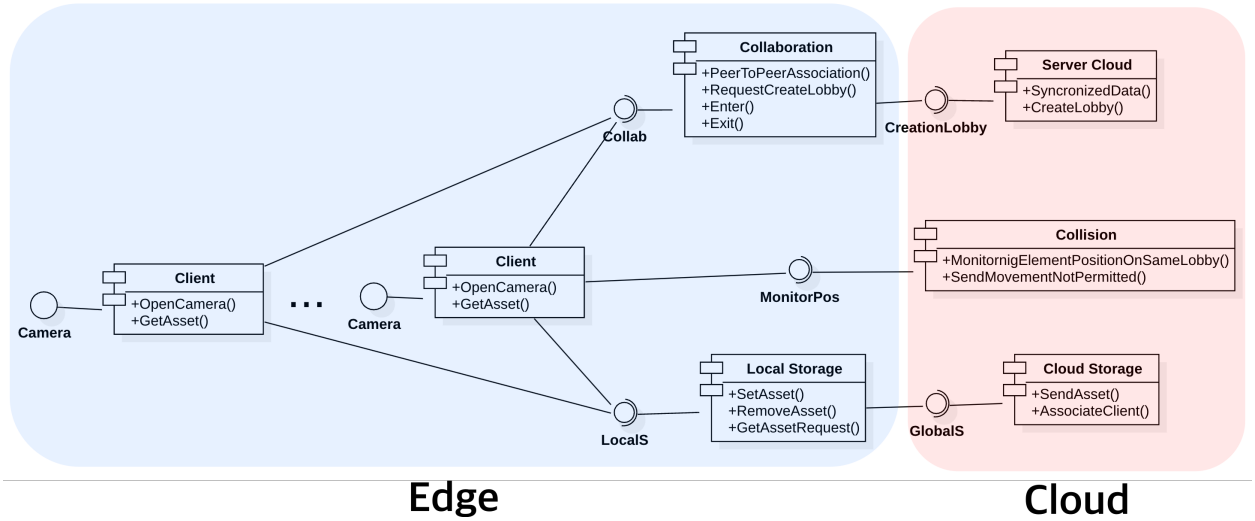


Figure 6.4: The figure depicts the component diagram of the second reference architecture, in which the computation is balanced between the cloud and the edge.

Even the devices themselves have been imagined to be computationally more efficient and capable of performing more complex operations than just sending and receiving data. All collision

and lobby creation request mechanisms have been repositioned within the ”Edge” part to speed up the execution process. Finally, the ”Local Storage” component has been added to improve performance. This component allows, through a specific algorithm, the saving of the objects intended to be deployed in the scenario, which in this case can be both AR and VR. The algorithm in question is described here:

Algorithm 8 Local Storage Management Algorithm

```
1: Input: Object_to_save, Threshold_uses, Maximum_objects
2: Output: Object saved in Local Storage, object management
3: procedure LocalStorageManagement(Object_to_save, Threshold_uses, Maximum_objects)
4:     Increment the counter of uses of Object_to_save
5:     if Uses_Object_to_save ≥ Threshold_uses then
6:         if Number_Objects_Saved < Maximum_objects then
7:             Save Object_to_save to Local Storage
8:         else
9:             Select the least used object among the saved ones
10:            Remove the least used object from Local Storage
11:            Save Object_to_save to Local Storage
12:        end if
13:    end if
14: end procedure
```

Concerning the work workflow of the various components, this is described in the component diagram Fig. 6.5 except the collider, since having the same functioning as the previous architecture, it was preferred to hide it so as not to make it hard to read.

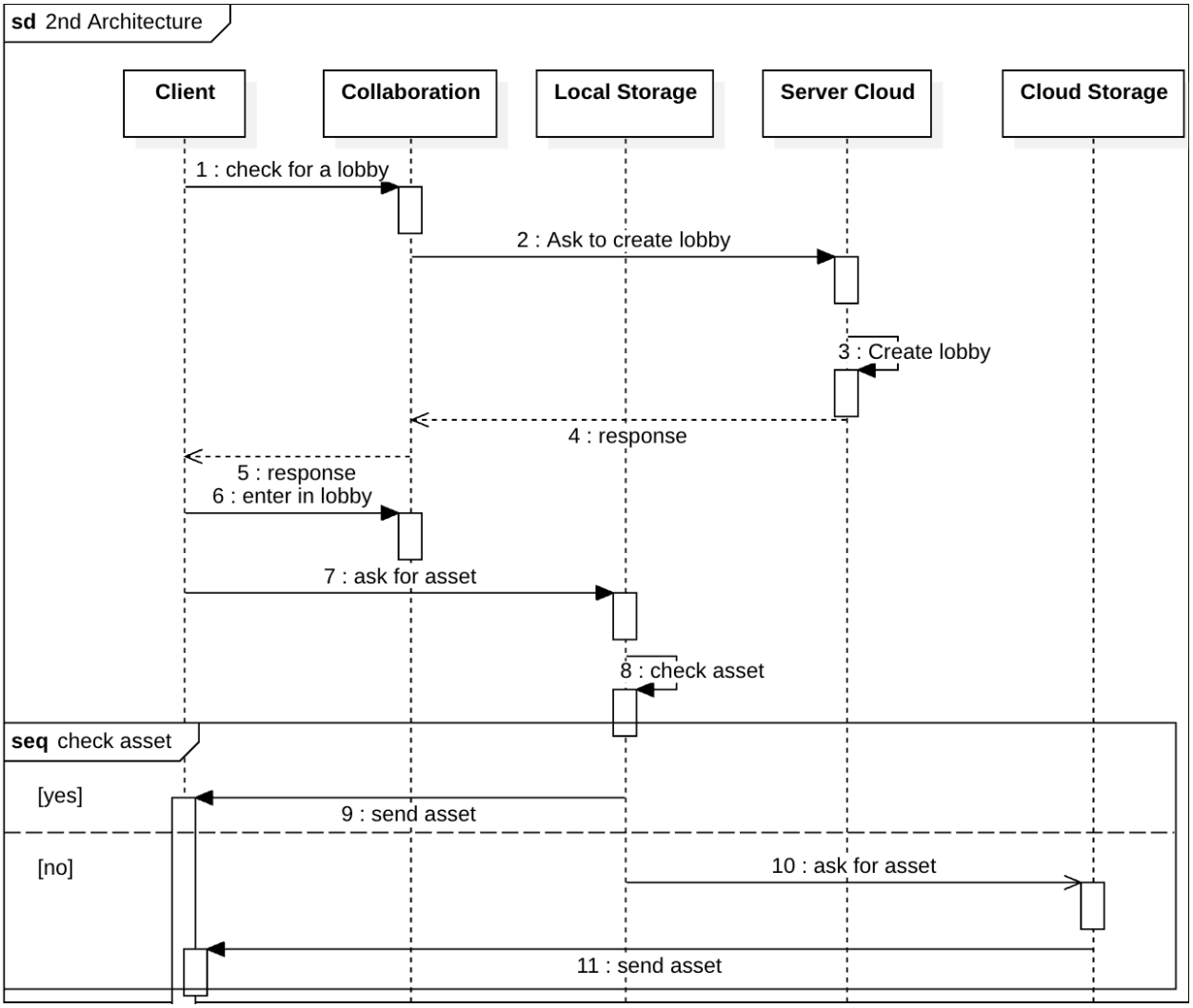


Figure 6.5: The sequence diagram shows the main operations of the second architecture including: lobby creation, asset management, and the interaction between local client storage, cloud server, and cloud storage

As you can see from the figure, the operation is as follows: starting from the client, this first requests to enter the lobby. If no lobby is active, the *Server Cloud* component is asked to create one. After doing this, different objects can be inserted into the AR and/or VR scene through a request made via the client, first to the local storage, and in the case it does not possess the requested asset to the Cloud Storage. During the entire session, the positions of the assets are continuously updated and monitored by the *Collision* component in the cloud, which takes care of unleashing triggers based on the type of collisions.

6.2.3 Decentralized IoT Solution with Prevalent Processing at the Edge

In this architecture we present, the computational load has been wholly repositioned towards the edge, leaving only storage and synchronization functions to the cloud Fig 6.6.

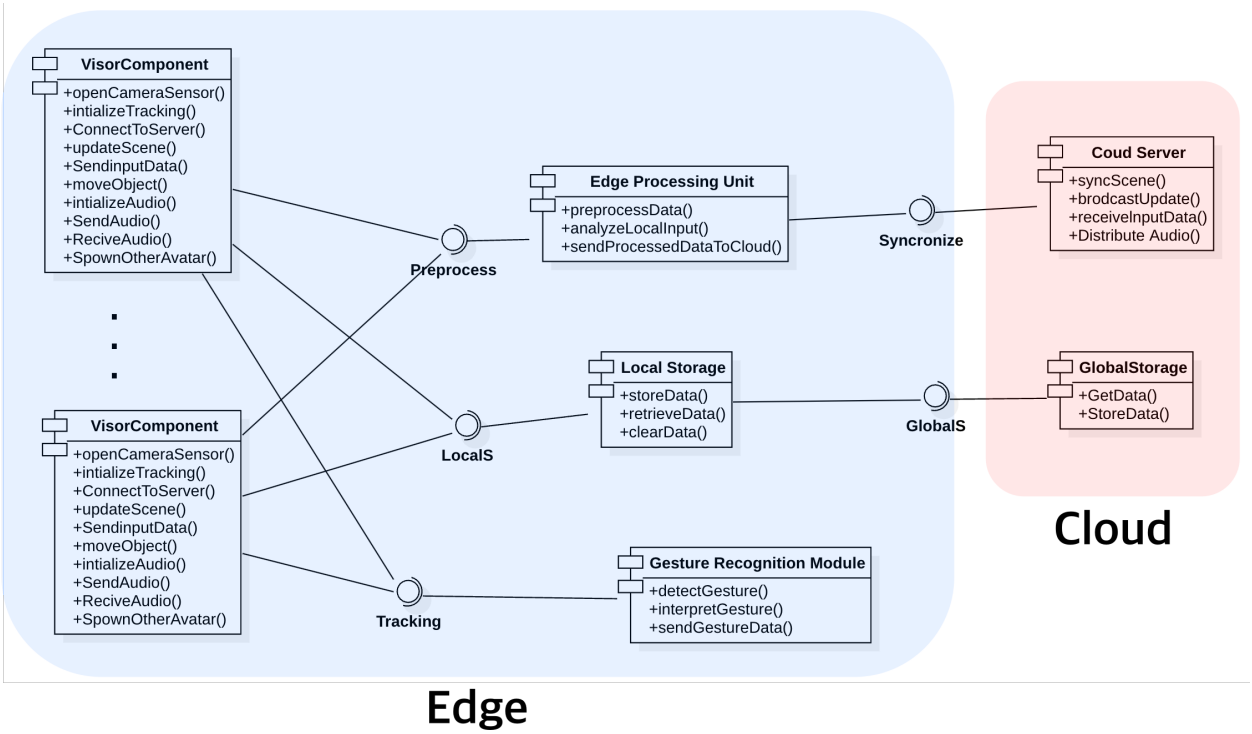


Figure 6.6: The figure shows the component diagram of the third reference architecture, in which most of the operations are shifted to the edge.

In this scenario, the main component is the **Visor Component**, which constitutes the interface through which users interact with the augmented reality environment, offering a range of essential features:

- **OpenCameraSensor**: Activates the camera sensors to integrate virtual elements into the real scene visible to the user.
- **InitializeTracking**: Initializes the tracking of the user's movements and position, allowing intuitive interactions with the virtual scene.
- **ConnectToServer**: Establishes a connection with the Cloud Server to access the shared scene and synchronize the experience with other users.
- **UpdateScene**: Updates the virtual scene display based on user inputs and updates from the Cloud Server.

- **SendInputData**: Transmits input data to the Cloud Server, such as movements and selections.
- **MoveObject**: Allows users to change the position of objects within the virtual scene.
- **InitializeAudio**: Prepares the audio system for user voice communication.
- **SendAudio/ReceiveAudio**: Manages the sending and receiving of audio streams, facilitating real-time communication.
- **SpawnOtherAvatar**: Generates virtual representations of other users, enriching the shared scene with their virtual presence.

The **Cloud Server** is the heart of the synchronization logic and management of collaborative sessions, supporting crucial operations such as:

- **SyncScene**: Ensures all devices display an updated and synchronized scene version.
- **BroadcastUpdate**: Broadcasts changes made to the scene to all participants, ensuring consistency in the shared experience.
- **ReceiveInputData**: Processes input received from users to update the state of the scene accordingly.
- **DistributeAudio**: Handles the distribution of audio signals among participants, allowing voice interactions within the virtual environment.

Integrating the *Visor Component* and the *Cloud Server* enables a dynamic and interactive workflow, allowing users to interact with virtual objects and other participants in an augmented reality context. The other supporting components are **Global and Local Storage**, which manage the download of the various assets with the same logic as the previous architecture. A component called **Gesture Recognition Module** will be responsible for interpreting the gestures and associating them with a movement. Finally, the Edge Processing Unit has the task of preprocessing the data to send to the cloud Server. Let's now analyze the interactions between these components by analyzing a communication prototype using a sequence diagram 6.7.

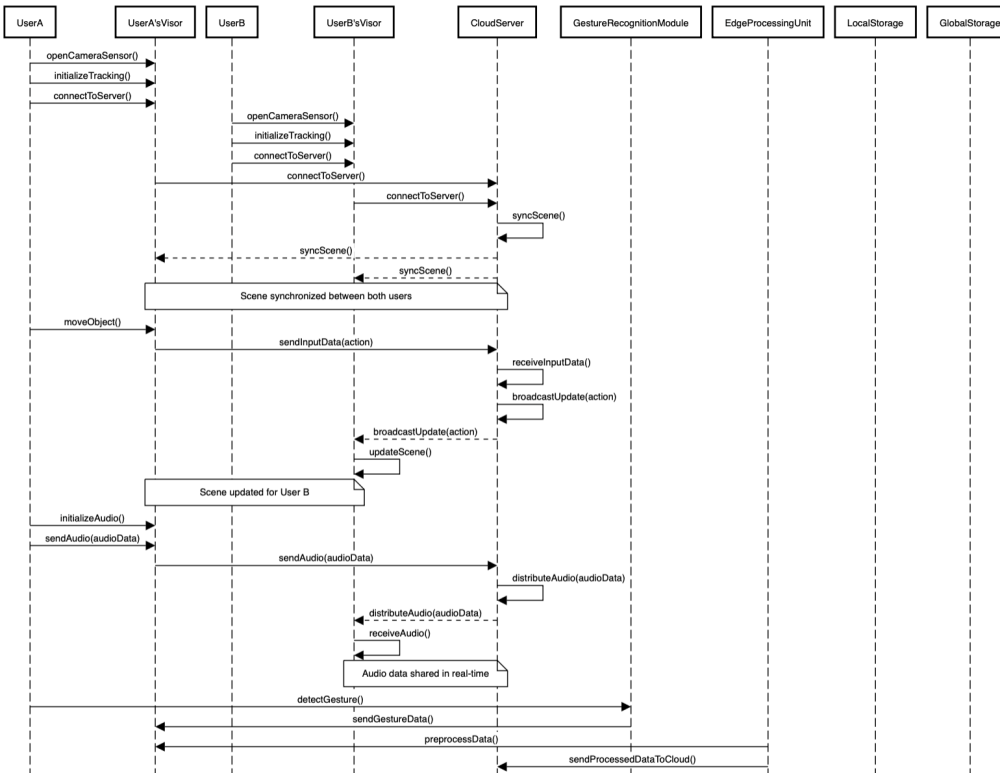


Figure 6.7: The sequence diagram shows the main operations for synchronizing a scene and sharing audio data between two users (UserA and UserB), using viewers, cloud servers, gesture recognition modules, and processing units at the edge. The flow includes opening camera sensors, synchronizing the scene, updating objects, and sharing real-time audio and gesture data.

This sequence reflects a typical workflow that allows users to share and interact within the same virtual scene superimposed on the real world, highlighting the collaborative capabilities of the application. At the start of the session, User A and User B activate their respective headsets, which begin tracking their movements and position through the *initializeTracking()* method that calls the **Gesture Recognition Module** component. This is essential to allow the system to interpret and respond to the movements and interactions of the users within the physical environment. Subsequently, the headsets connect with the Cloud Server via *connectToServer()*, a crucial step that allows users to access the same virtual scene and collaborate in real-time. Once the connection is established, the server proceeds with the initial synchronization of the scene for both users, sending the current state of the virtual environment via *syncScene()*, ensuring that User A and User B see a consistent and up-to-date representation of the shared scene. User A interacts with a virtual object during the session by moving it within the scene. His headset captures this action via *sendInput-Data(action)*, then communicates the update to the Cloud Server. The server processes the input received and, through *broadcastUpdates(action)*, propagates the changes to all connected users, including User B, which gets the update and sees the modified scene thanks to updateScene(). At

the same time, an audio communication dimension develops between users. User A sends audio data (such as words or sounds) that, passing through the server via *sendAudio(audioData)* and *distributeAudio(audioData)*, reaches User B, allowing him to listen to User A in real-time. When User B moves within the environment, his actions are transmitted to the server, which, in turn, updates User A on the new position of User B in the virtual scene, ensuring that the changes are visible to all participants. It is worth noting that at the same time as the user's movement tracking initialization process, a series of routines are started that involve all the sensors, including the camera, which is essential for the success of an augmented reality application. Finally, it is worth noting that a routine constantly called is *UpdateScene()*, which calls the camera update for each frame.

6.3 Implementation

After logically defining three reference architectures, this section will show the technologies used to implement them, mapping them for each component. Only then, before proceeding with simulations, different applications will be developed to demonstrate the correct design for each of the three architectures. Each application will be a concrete example of use, showing how each architecture behaves in real-life usage contexts.

6.3.1 Technologies Used

In this work, the main technologies used to implement the components described within the architectures are the following, as depicted in Fig. 6.8:

- **Python**: the language used to implement the motion capture system.
- **Unity and C#**: Unity was used as the main framework for the design of the three different architectures. Furthermore, specific scripts were created in C# for the more complex tasks.
- **OpenCV**: a fundamental Python library for implementing the first architecture, as it allows the use of pre-trained machine learning models [201].
- **Flusk**: a framework that allows the easy implementation of multiplayer software, particularly optimized for Unity.
- **Docker**: mainly used to create volumes capable of storing pre-loaded 3D assets [202].
- **Android Studio and Xcode**: used to convert applications created in Unity for the iOS and Android operating systems, respectively [203].

- **Firebase**: mainly used as online storage through the Firestore and Storage services [204].
- **Normcore**: a cloud service mainly used to share real-time data. After implementing the different architectures, each was validated through different case studies to verify their functioning and performance [205].

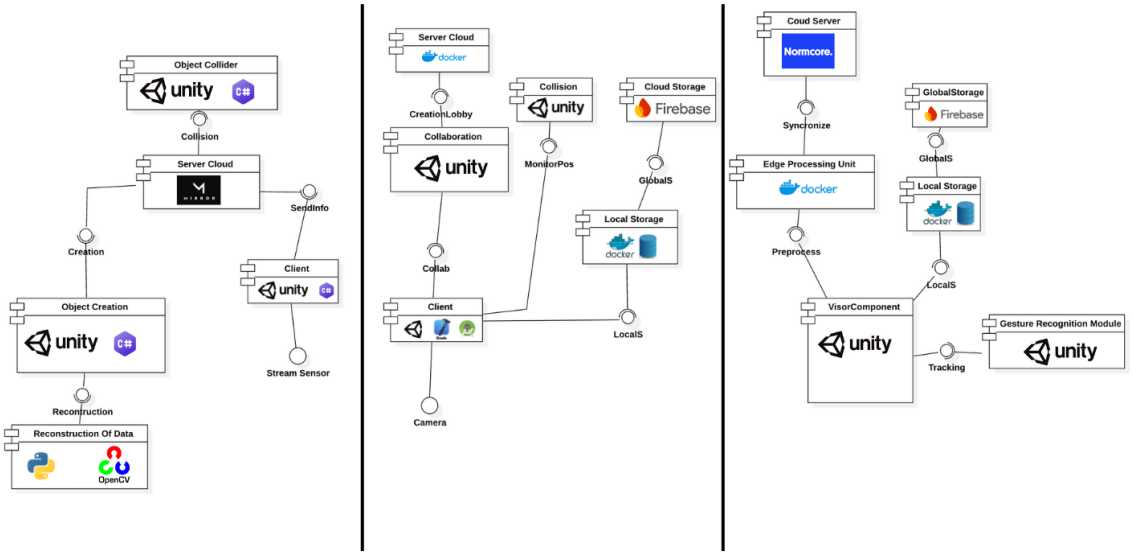
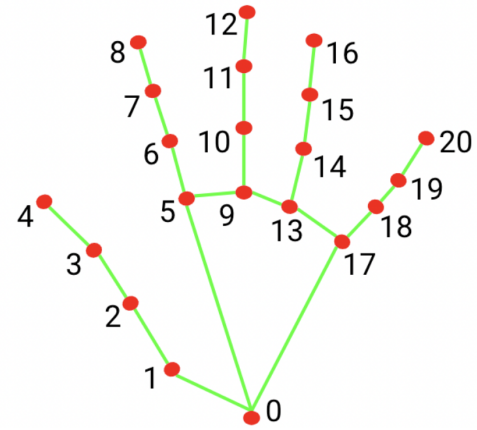


Figure 6.8: The figure shows the three reference architectures mapped with the most appropriate technologies to implement.

6.3.2 Case study 1: Collaborative Virtual Reality with Hand Recognition

For the first reference architecture mentioned in the previous section, an application was implemented to verify its proper functioning. It is a virtual space where different users can interact with each other or virtual objects. The user's hand represents the connection point between the real and virtual worlds. The operation is as follows: the user places his or her hand, left or right, in front of the camera. This captures the video stream and sends it in real-time to the Cloud. Here, the Object Creation component deals with virtual hand reconstruction using a rule-based approach, leveraging the Reconstruction of a Data component that identifies 21 different hand landmarks through a specific model called the Hand Landmarks Detection Model [206, 207, 208] Fig. 6.9.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 6.9: The figure conceptually illustrates the division of the hand into 21 points, allowing real-time tracking of the independent movements of each finger or the entire hand.

Depending on the number of connected users, the arrangement of individual hands within the virtual space changes, as described in the architecture. Collaboration was implemented through two different approaches: the possibility for each user to interact with virtual objects, which in this scenario are simple geometric figures embedded in the environment, and the possibility of interaction between multiple users (Fig. 6.10).

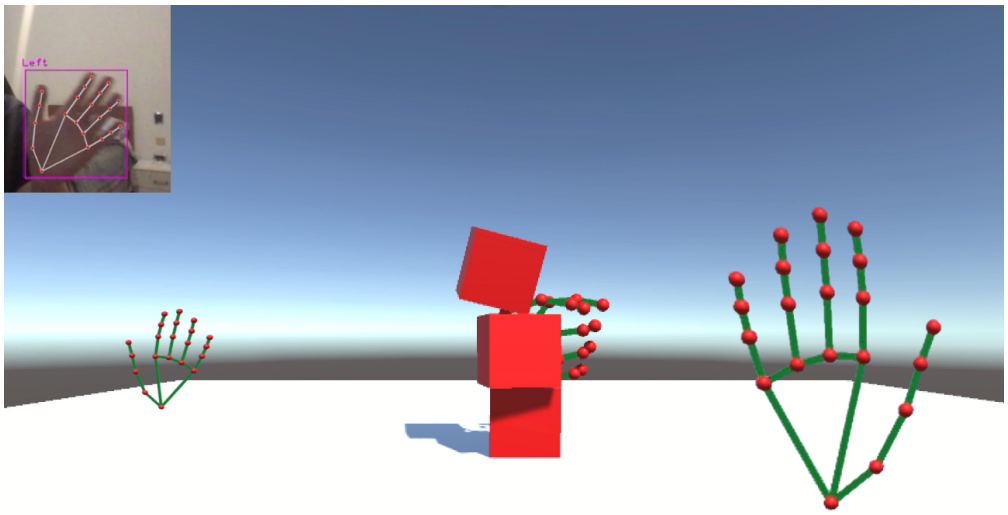


Figure 6.10: Screenshot of the application in operation, deployed on the first architecture, with three different users connected. Through its hand reconstruction, one of them interacts with the shared objects within the scene, specifically the cubes.

6.3.3 Case study 2: Collaborative Augmented Reality with the Use of Smartphone

For the second architecture, a case study based on smartphones was deployed, involving using three-dimensional objects collaboratively within a real environment. The creation of this application is also based on the use of Unity [209, 210]; unlike the previous case, the edge part manages the spawn of objects and interactions with them. For collaboration, the devices will use the local network to create a ”room”, thanks to which each client will communicate to all room members through P2P [211] messages the positions or movements of an object 6.11.

Figure 6.11: Photo taken during the WeMakeFuture exhibition held in Bologna at the MIC stand, in which different users, through the use of different devices, visualize and manipulate different objects collaboratively.

Once the user selects an object, the activated edge component is Local Storage. At this stage, an initial request is sent to the Local Storage, which searches its Docker volume [212]. The Local Storage contains the objects previously downloaded previously downloaded objects from the cloud and sends it them directly to the client for use if the object is already present. If, on the other hand, the object is not available locally, a new request is generated for cloud storage based on Firebase. The cloud storage system is structured through key-value logic, optimized to speed up the search and retrieval of needed items.

6.3.4 Case study 3: Collaborative Hybrid Virtual and Augmented Reality with use of Headset

This case study shifts most of the processing to the edge through visors, which handle real-time tracking of users' head and hand movements. The main objective is to overlay virtual elements on the incoming video stream, thus creating an augmented reality experience. For this specific project, the *MixedRealityToolkit* library was used [213]. This library offers an integrated framework supporting various inputs, such as gestures and spatial interactions. It enables the development of immersive experiences without implementing specific functionality from scratch. The scenario implemented 6.12 provides that, when the application is started, each user, via the viewer, views both the real environment, which differs for each of them, and an overlapping virtual layer, within which there are various interactive and collaboratively shareable objects. In addition, individual users can move within their real space, and their presence is visualized by other participants in the room through virtual avatars.

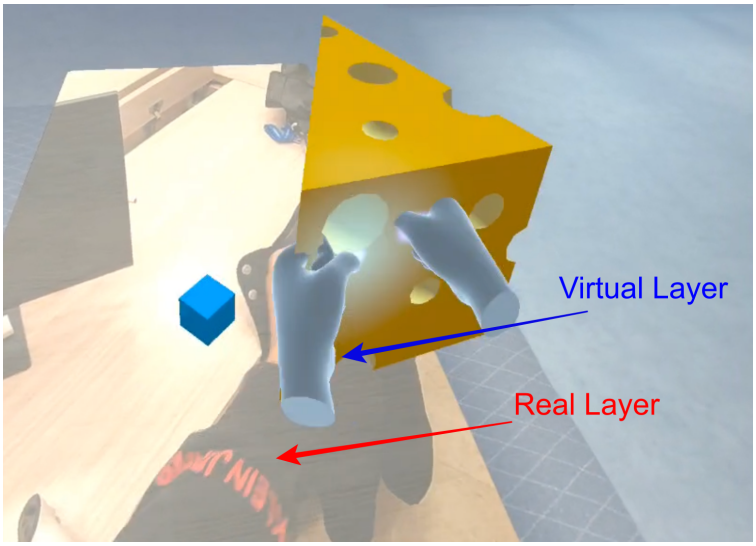


Figure 6.12: A frame captured while using the application shows how the real flow (the user's arm) is superimposed on a virtual flow (reconstructed hands and a shared usable object).

6.4 Testing and Validation

This section shows simulations have been performed and the results obtained are discussed. CloudAnalyst [214] was used for all simulations. It is a simulation tool based on CloudSim [215, 216], specifically designed to model and analyze the impact of the geographic distribution of cloud computing resources on performance. It provides an interface to characterize the real nodes and verify their correct functioning. With the help of this tool, various simulations were carried out, which

we will analyze in detail in this section. The Cloud Analyst tool was configured for the simulation, and the different nodes are represented in Fig 6.13. In detail, the characterizing nodes are the following: UB_1, UB_2, UB_3, UB_4 and UB_5 represent the various pulls of users that can vary from a value n to a value n'. In our specific case, four different simulations were carried out for each of the three architectures. The number of UB_S was set to 5, the peak user hours for each UB were set from 3:00 P.M. to 9:00 P.M., while the number of users varied in the following way. NUB_{min} is denominated as the minimum number of users for each UB, NUB_{max} is the number of connected users for each UB, N_{max} is the maximum number of total connected users, and N_{min} is the minimum number of connected users. The configurations considered are as follows:

1. $NUB_{\text{min}} = 0$ and $NUB_{\text{max}} = 1 \Rightarrow \text{N}_{\text{min}} = 0$ and $N_{\text{max}} = 5$.
2. $NUB_{\text{min}} = 1$ and $NUB_{\text{max}} = 10 \Rightarrow \text{N}_{\text{min}} = 5$ and $N_{\text{max}} = 50$.
3. $NUB_{\text{min}} = 10$ and $NUB_{\text{max}} = 100 \Rightarrow \text{N}_{\text{min}} = 50$ and $N_{\text{max}} = 500$.
4. $NUB_{\text{min}} = 100$ and $NUB_{\text{max}} = 1000 \Rightarrow \text{N}_{\text{min}} = 500$ and $N_{\text{max}} = 5000$.

As for the components in red, these represent the cloud and edge resources on which we have deployed the components described in the section dedicated to architectures. Since most of the services used came from America, we simulated the resources as if they were allocated by consistently setting the data transmission time with this design choice. As for the edge components, these were simulated by setting the latency time appropriately low to simulate a resource allocated close to the IoT devices and a reduced computational capacity based on the preliminary tests done in the implementation.

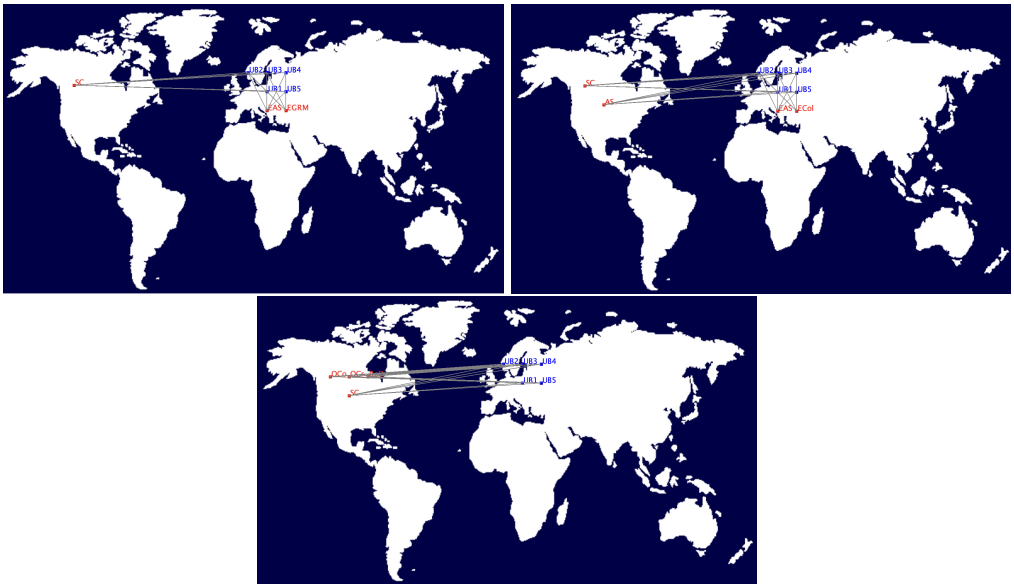


Figure 6.13: The figure shows graphically, for each architecture, the defined components and user groups, highlighting the interactions between the users and the various components.

Simulation	Average Response Time (ms)	Min (ms)	Max (ms)
1Arch0_1	297,66	44,12	621,06
1Arch1_10	329,92	43,76	673,30
1Arch10_100	800,68	73,04	1819,52
1Arch100_1000	5222,49	218,75	17223,80
2Arch0_1	48,00	36,72	62,64
2Arch1_10	76,27	36,70	269,22
2Arch10_100	409,71	34,53	2131,46
2Arch100_1000	4051,75	39,34	27742,37
3Arch0_1	47,61	35,30	64,73
3Arch1_10	74,46	33,75	269,02
3Arch10_100	353,02	37,72	1886,20
3Arch100_1000	3423,89	37,59	23583,14

Table 6.1: Simulation results: average, minimum and maximum response times of the three architectures in the different designated scenarios.

We draw relevant conclusions by analyzing the data we obtained from the simulations. As expected, the average response times increase as the number of connected users grows. In particular, in the first architecture, the response time increases drastically from 1 user to 1000 users (from 297_{ms} to over 5000_{ms}). This factor highlights a second important consideration. Architectures two and three, which are those with the application of edge devices, deliver better performance.

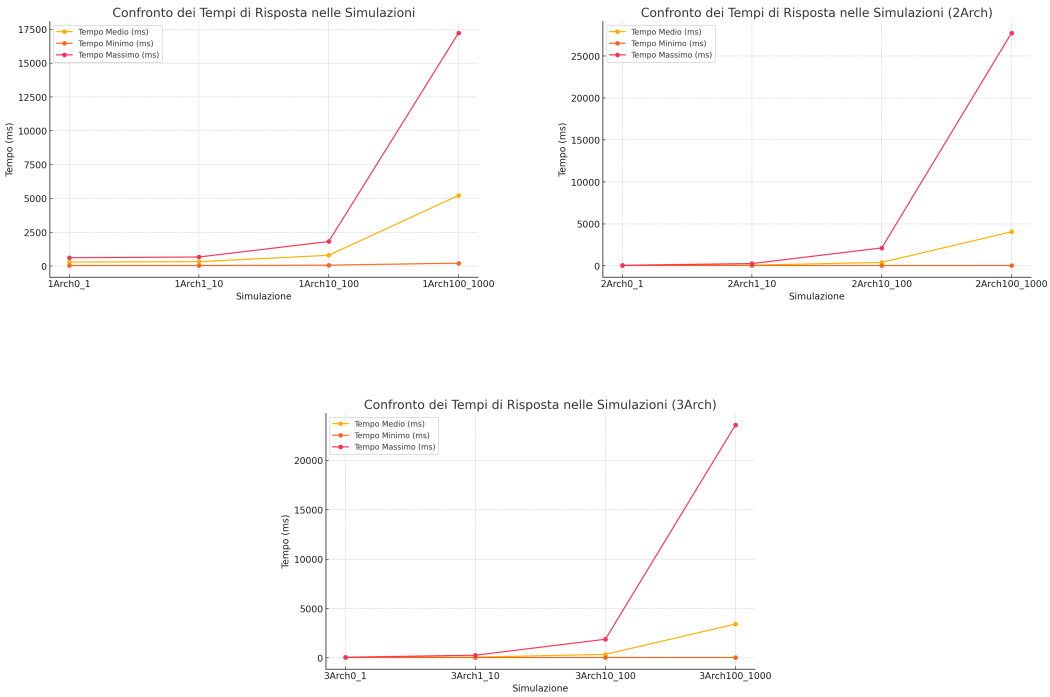


Figure 6.14: Graphical representation of performance as the number of users changes for three different architectures, with the figures sorted according to the order of the architectures presented. For each architecture, the orange line indicates the minimum time, the red line the maximum time, and the yellow line the average time.

Figure 6.14 shows that the resource distribution and optimized configuration of architectures two and three make them more scalable and efficient. Moreover, among all architectures, the minimum response times are relatively stable and low, especially for Architectures two and three, indicating that systems can handle isolated requests more efficiently, especially with low user load.

6.5 Concluding Remarks and Future Work

This work explored the deployment and performance of collaborative Virtual and Augmented Reality systems in a Cloud Continuum context, focusing on integrating and comparing cloud-centric and edge-centric hybrid architectures. Several key insights and conclusions emerged through the definition and implementation of these architectures and their subsequent validation through simulations. With the increase in the number of users, the cloud-centric architecture showed a substantial degradation in performance, highlighting the limits of an utterly centralized approach. Instead, hybrid architectures have balanced performance and scalability by distributing computational loads between cloud and edge resources. Among the future developments, indeed of considerable importance is to consider the privacy and security aspect by exploring methods of secure data trans-

mission and storage adapted to these distributed architectures [217, 218]. Different scheduling algorithms could also be applied to evaluate the various approaches to determine which maximizes the number of concurrent users. Furthermore, using other case studies to include more different application scenarios such as: healthcare [219], military [220] and educational fields, could provide further insights into the versatility and limitations of the proposed architectures. All of these advances could lead to the definition of a unified reference architecture for the AR/VR scenario, capable of adapting to all possible application contexts and being deployed modularly, depending on available resources and the specific needs of the case study.

Chapter 7

Result and Future Directions

Numerous topics have been explored throughout this doctoral program, all converging toward the goal of defining, implementing and testing different reference architectures in a cloud continuum scenario. The main topics covered include augmented reality, virtual reality and artificial intelligence. In particular, machine learning, deep and federated learning, natural language processing and expert systems techniques were addressed. The central pillar of this work was the cloud continuum, from core infrastructure to edge devices. Edge computing, in particular, played a crucial role in the contexts analyzed, bringing computing resources closer to the data source. The architectures designed and implemented in this area demonstrated the benefits of decentralized computing, which are particularly useful in scenarios such as the Internet of Things (IoT) and smart grids. All the architectures developed were classified into three macro areas: semantics, learning approaches, and augmented and virtual reality. Reference architectures implemented in the semantic field include: COVID-19 Expert System: an expert system to identify potential COVID patients and suggest treatments, using an ontology base. CR8: a storytelling tool that allowed manual and automatic creation of stories, with capabilities to manage authentication and simultaneous file access from multiple users, as well as public and private cloud storage integration. In machine, deep and federated learning, four architectures were developed in the medical field: three to test algorithms on real data from diabetic patients in local, global and federated scenarios, and one to study air quality by identifying environmental patterns based on the actions of people in their homes. Another architecture was implemented in collaboration with Enea to monitor social networks and collect information on conferences and events regarding energy communities. Finally, three different reference architectures were implemented to manage case studies in AR and VR. These papers have contributed to advancing the state of the art, offering several reference architectures for relevant case studies, and are a resource for anyone who wants to approach the world of the cloud continuum, which is the central pivot of this thesis. The case studies cover a variety of areas, predominantly in the medical field, but also in educational, military, and social settings. General

future goals, in addition to those mentioned at the end of each section, include the optimization of the aforementioned architectures also through the use of offloading techniques and 5G. On this topic, a work, published at the SWIMS 2024 conference, is reported in the appendix C. In addition, we can merge all the architectures defined the creation of a single final modular architecture that can be invoked and adapted based on the case study. Through a framework, it would be possible to select the case study, visualize the components suggested by the system, and simplify application development and deployment. This framework allows for the dynamic selection and configuration of architectural components based on the constraints of each use case, optimizing performance and scalability. Moreover, the architectures developed can be progressively integrated into the other reference architectures cited in chapter 2, enriching it with specialized modules that address complex scenarios such as real-time processing for AR/VR, federated training on sensitive data, and secure resource management in distributed environments. This integration helps to extend the current architectures with more flexible and adaptable solutions, enhancing interoperability between cloud providers and optimizing the utilization of edge resources.

List of Figures

List of Tables

List of Algorithms

Appendices

Appendix A

Towards an Intelligence system to support Diseases' Diagnoses and improve Health treatments

E-health is becoming more and more a reality, thanks to the increase of IoT health devices and to the availability of new technologies, algorithms and software, such as Semantics, Machine and Deep Learning. Systems are being developed to support doctors in making diagnoses, starting from data collected from sensors directly connected to the patients. This paper describes the preliminary steps towards the realisation of a platform for specialised doctors, capable of supporting diagnosis, by analysing the patients' symptoms, seen as an input to the system, and to recognise the possible diseases. In particular, we start by analysing semantic technologies as a means to describe sensors' outputs and to define rules that determine the diagnosis. To this end it is necessary to create or identify a support ontology used to describe the patients' symptoms, and then to define rules to identify the correct disease, in order to build a rule based expert system. In particular, this paper focuses on the identification and eventual modifications applied to existing disease ontologies.

A.1 Introduction

Today, in the medical field, the use of machine learning is becoming more and more a trend. The idea of being able to support doctors with automated algorithms, that can rapidly determine a diagnosis by analysing all of the patients' symptoms, could save more lives in the future. While this would be a very welcome outcome, we are still far from realising such a support system.. However, the availability of data coming from IoT sensors, the ever increasing computational power offered by smart devices and the availability of new learning algorithms are surely pushing

towards such a goal. The semantic based research that has been carried out and described in this paper arise precisely from the idea of creating a system that, thanks to a series of inputs provided either automatically by sensors, or manually by the doctor herself, manages to identify which of the infinite diseases could correspond to specific symptoms. While this would certainly be supportive for the practitioner, such a system would not simply substitute the doctor, but it will represent a real support that can confirm a specific diagnosis or provide an alternative "opinion". In order to recognise diseases, the idea would be to refer to two different approaches: one based on semantics and inference rules that, through an Expert System, would provide an exact outcome for each input; one based on Machine Learning which, through the observation of data, provides an estimation of the outcome with relative confidence levels. Subsequently, the idea would be to compare the two approaches and try to understand which is the best and, above all, for what reason. The first step described in this paper is to identify a medical domain ontology that is appropriate for the purpose of building the Expert System. To this end, the main ontologies of the medical field have been analysed and an attempt has been made to select and extend the most suitable, that is the Disease Ontology (DO) [221]. The remainder of this paper is organized as follows: section A.2 describes the overall methodology; section A.3 reports the State of the Art; section A.4 presents the Disease Ontology that has been studied here; section A.5 describes the extensions that have been made to the original ontology; section A.6 presents the methods used to populate the ontology with new individuals; section A.7 closes the paper with final remarks.

A.2 General Description of the Methodology

As stated in the Introduction, the work described in this paper is a preliminary step towards the creation of a more complex system. Indeed, the final objective is to compare Semantic approaches with Machine Learning techniques, to understand if they can be used alternatively or need to be integrated. Figure A.1 describes the steps that will be followed in the definition of the intelligent system.
On the left, the main steps describing the creation of a Rule based Expert System. It is defined as a four step procedure, in which it is necessary to:

1. Identify and evaluate existing ontology in order to choose the best one among those already produced by the Research community.

2. Extend the selected ontology.

3. Create the Rules that will be used to produce the output of the System, that is the Disease recognition.

4. Evaluate the overall system with data provided as an input from external Sensors or Practitioners.

While the left side focuses on the Semantic Aspects, the right one focuses on Machine Learning approaches. As before, four steps are taken in consideration

1. Identification of a set of potential Machine Learning algorithms to apply.
2. Test of the algorithms on Data. The first two steps can be repeated until a satisfactory algorithm or set of algorithms has been identified.
3. Algorithm tuning, needed to optimise the outcome of the learning.
4. Evaluation of the Machine Learning approach.

Both the two pipelines converge into a Comparison step, which is necessary to understand if one of the two approaches can be considered better than the other one, or if an integration is needed. This paper focuses on the first two steps of the left pipeline, thus concerning Semantics and Ontology identification and augmentation in particular.

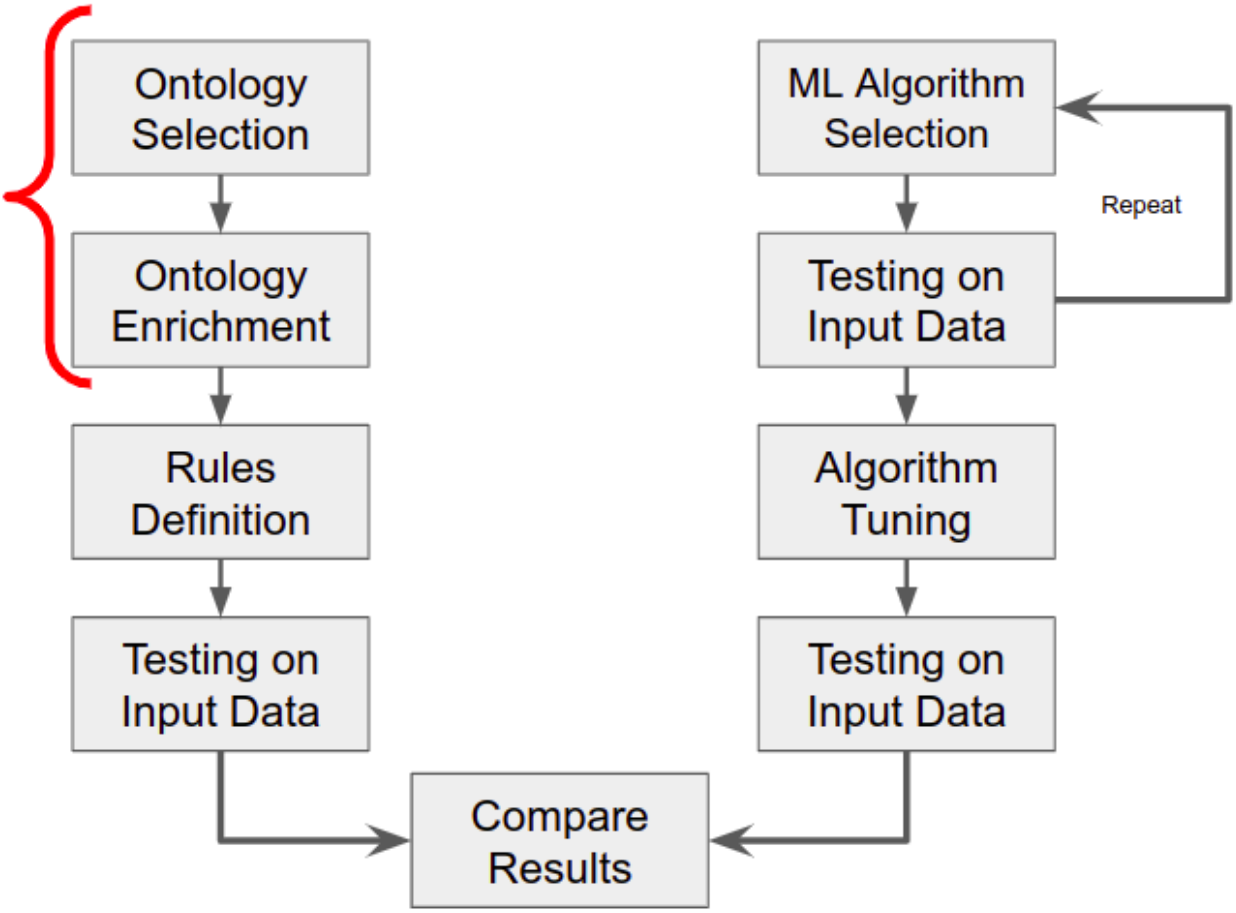


Figure A.1: Description of the Methodology

A.3 State of the Art

When groups of agents need to work together, be they people or software systems, you need to ensure that they understand the requests and information they receive. The interaction among agents essentially depends on the adoption of a conceptualisation layer, that is a formal representation of the reality of a specific situation as perceived and organised by an agent, together with a common language [222]. Through an ontological characterisation of the information, concepts can be found, isolated, organised and integrated according to what matters most, that is their meaning [223].

In computer science an ontology, used especially in studies on artificial intelligence and in the classification of data, is the attempt to formulate a classification of concepts, hence a hierarchy, within a domain, by also providing complex relationships and correlations among the described concepts. An ontology is therefore an attempt to formulate an exhaustive and rigorous conceptual

scheme within a given domain, providing more complex relationships than simple taxonomies, and building up models of complex systems with related logical axioms, passing through forms of intermediate classification.

The application of Ontologies in the Computer Science field has provided significant results in the past, and therefore could represent a good starting point for future developments. In the field of Cloud Computing, ontologies have been successfully adopted to reduce interoperability issues among Cloud Service [224]. In [225] semantic technologies have been integrated with a Big Data pipeline and Machine Learning techniques to build a complex system supporting the identification of anomalous activities in the Italian Juridical System, demonstrating that Semantics and Machine Learning can be integrated to provide better results.

In the same way, in the medical field the construction of ontologies combined with artificial intelligence techniques could lead to more efficient treatments for the patient.

Many ontologies have already been implemented in this field, and some of them have been analysed and reported here.

The first was PATO (Phenotypic Quality Ontology) [226] which can be used together with other ontologies such as GO (Gene Ontology) [227] or anatomical ontologies to refer to phenotype [228]. While interesting from an academical point of views, such and ontology is very far from our objective. Subsequently, the VO (Vaccine Ontology) was analysed [229, 230]. Again, such and ontology resulted to be more suitable for ensuring a consistent representation of vaccine knowledge and supporting automated reasoning about vaccines. Ontology for general medical science was subsequently analysed (OGMS). OGMS [231] includes very general terms used in all medical disciplines, including: ”disease”, ”disorder”, ”course of disease”, ”diagnosis”, ”patient” and ”health worker”. However, this ontology is more linked to the technical aspects of medicine related to information technology. Finally, the focus was on “Disease Ontology” (DO) [221] which appears to be more suitable for our purposes. This is because it does not classify diseases only, but it also refers to symptoms and, in general, covers many topics in the medical field. This ensures greater possibilities for future implementations. Section A.4 is completely dedicated to such an ontology.

A.4 The Disease Ontology

Disease Ontology (DO) is an ontology developed by biomedical researchers coordinated by the University of Maryland School of Medicine, Institute for Genome Sciences. The goal of this ontology is to provide the biomedical community with consistent and reusable descriptions of human diseases and of the characteristics of the phenotype in general.

It contains a consistent number of classes, subclasses and properties relating such elements. Among

classes, we have:

1. The Anatomy class, which in general is dedicated to the exact cataloguing of all the anatomical details of the human being. The word "Anatomy" itself comes from the Greek anatom and means dissection, which in ancient times was the main methodology for knowing the internal structure of a living being. In general, the class is divided into three main subclasses which are respectively:
   (a) Immaterial Anatomical Entity: most of the anatomical spaces are present. To name a few we can refer to the "epidural space" which is the space between the two layers of the dura mater (the outermost layer that covers the brain and spinal cord). Or another example could be the "retro-peritoneal space".
   (b) In the Material Anatomical Entity section, on the other hand, there are most of the parts of the body divided into categories.
   (c) Finally, there is the "Anatomical Cluster" class in which there are entire regions of the body. An example would be the 'hard palate', which is the anterior region of the palate.
2. The Cell class contains the different types of cells which are the following:
   (a) Abnormal cell: cells that have abnormal conditions, that are malformed or malfunctioning in some way. A subclass is represented by malignant cells.
   (b) Neural cell: that is all the cells of the nervous system among which we can see the neurons or the cells of the pituitary gland.
   (c) Native cell: where the eukaryotes cell is present, which is one of the most evolved cells.
   (d) Neuron: this class describes again a specific Neural cell, and could be considered as a repetition of the "Neural cell" class.
3. The ChEBI (Chemical Entities of Biological Interest) class, on the other hand, is divided into two sub-classes, which are respectively:
   (a) Chemical entity: where there are some subclasses concerning chemistry, including the atom or chemical substances.
   (b) Role: a section within which, after a series of subclasses, drugs are included.
4. Disease is one of the most complete sections of the ontology, where almost all human diseases are present, divided into various categories. The class is one of the most complete in ontology, for this reason it would be almost impossible and, above all, superfluous to analyse all the subclasses in detail. We can limit ourselves by saying that all diseases are ordered according to the areas of interest.

5. FoodMaterial: This section which generally concerns food is divided into 4 parties that relate to:
   (a) Foods derived from plants
   (b) Foods derived from the sea
   (c) Foods derived from crustaceans
   (d) Foods derived from vertebrate animals
6. Inheritance pattern: in this category there are various inheritance patterns and the subclasses are as follows:
   (a) inherited chromosomal characteristics
   (b) mitochondrial inheritances
   (c) monogenic inheritances
   (d) multifactorial diversity
7. Symptom: within this class there are all the categories related to symptoms. Also in this case, being one of the main classes, the list of all the subclasses would make the reading too heavy.

The ontology contains about 70 Object Properties, representing the correlations among all the classes and individuals in the ontology. More properties will be added as we extend the ontology to accomodate new Classes and related relationships.

A.5 Extending the Disease Ontology

While very complex and detailed, some important elements were deemed missing and were consequently added. In particular, some Classes and Object Properties have been added, together with necessary Data Properties that were completely missing in the original ontology. All the additions made are described in separate subsections.

A.5.1 Extension of Classes

In general, the concepts of Anatomy is divided into two sub-fields:

1. Zootomy: which deals with animal anatomy
2. Phytotomy: which deals with plant anatomy

The original ontology only considered Animal anatomy, while neglecting Plant, so the 'Phytotomy' class was added as a sub-class of Anatomy.
In the Anatomy class, in particular within the 'Immaterial Anatomical entity', the following subclasses have been added:

1. Retropharyngeal Space and Canine Space: they are two spaces of the head and neck respectively.

2. Cystohepatic triangle: an anatomical space bounded below by the cystic duct and above by the liver.

After analysing the entire ontology, it is possible to notice that, within the "Food material" section, a section dedicated to invertebrate animals is missing. Considering that within the missing class there should be considered arthropods, which are the largest group in the animal kingdom with more than 875,000 known species, it was deemed necessary to add a dedicate class and related subclasses. Therefore, the following additions were made:

1. Annelids

2. Arthropods

3. Clam

A.5.2 Extension of the Object properties

While Classes are necessary to express important concepts, Object Properties are fundamental to determine the relationships existing among such concepts.
As a start, a transitive Object Property has been added, as shown in the Protegè screenshot in Figure A.2, to represent the transitive relation between Respiratory Failures and Chronic Diseases.

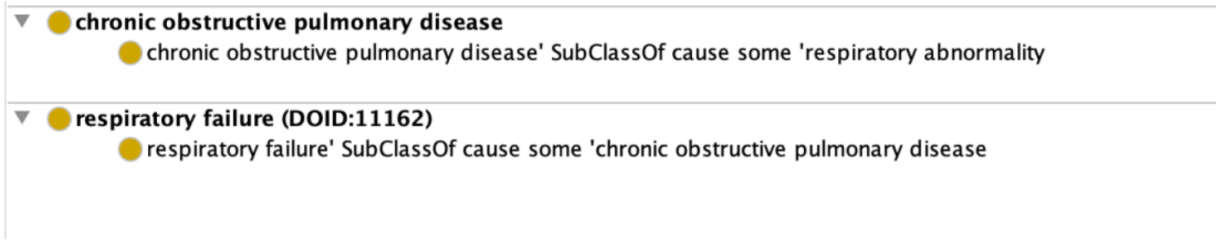


Figure A.2: A transitive Object Property added to the DO ontology

In this case the transitive Object Property involves 3 different classes:

1. chronic obstructive pulmonary disease

2. respiratory abnormality

3. respiratory failure

A Functional Object Property and its inverse have been added between Cells and Hepatocytes. The Functional Property **is_formed_from**, described in Figure A.3 determines that a Liver can only be constituted by Hepatocytes. Since the viceversa is also always true (Hepatocytes can be only found in a Liver, unless a sever pathology occurs) and Inverse Functional Object Property **forms** was created.

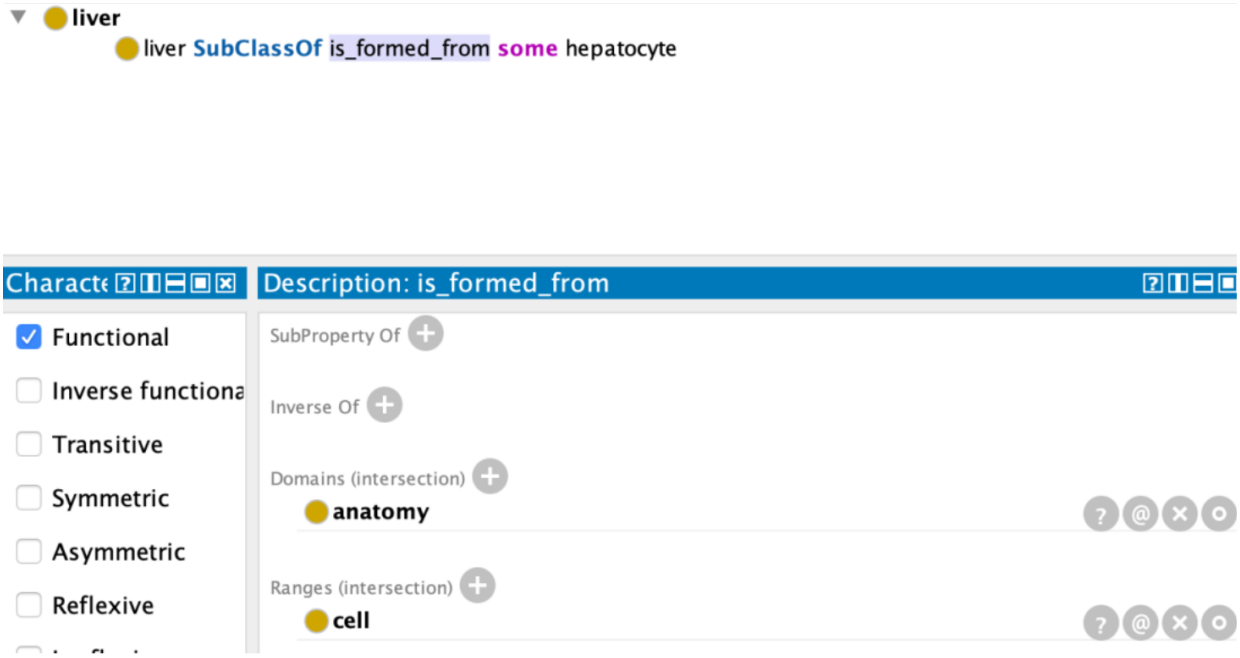


Figure A.3: The Functional Property **is_formed_from**

Similarly, the Inverse Functional Object Property forms, as shown in Figure A.4:

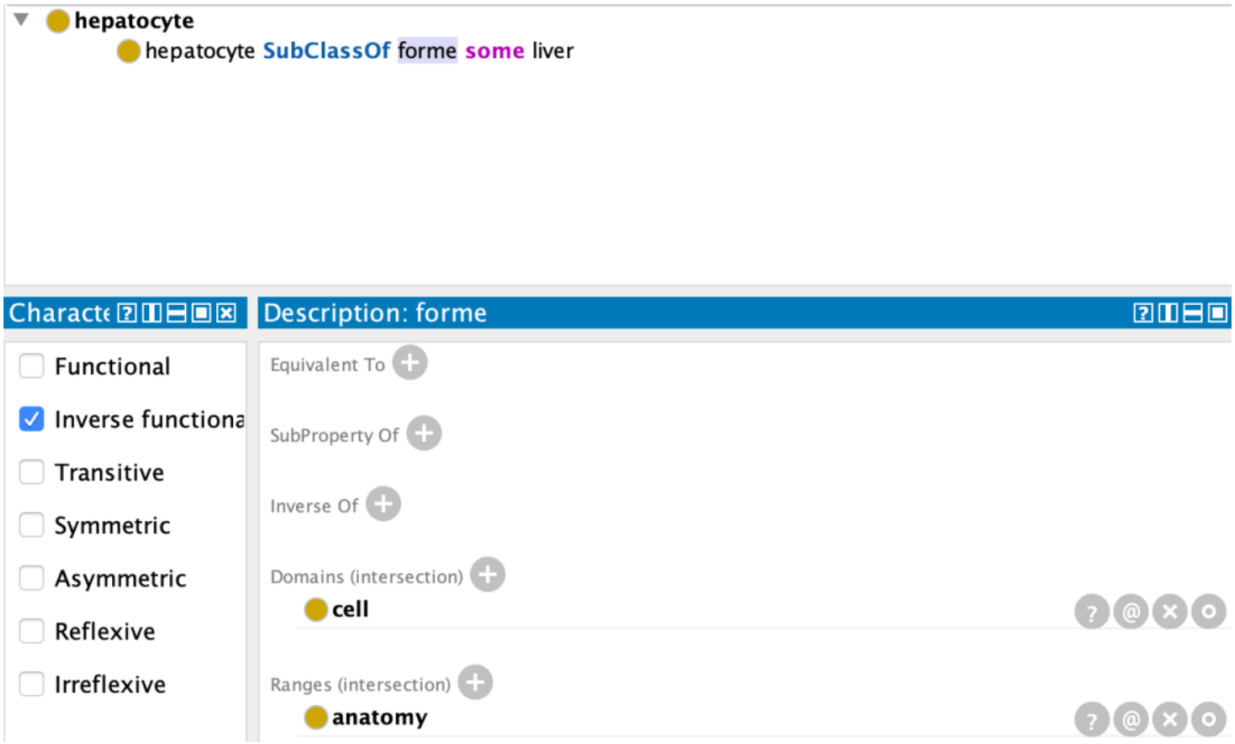


Figure A.4: The Inverse Functional Property **forms**

Domain and Ranges of these two properties are the same, but they are simply inverted.

A.5.3 Extension of the Data Properties

The Disease Ontology was completely missing Data Properties, that are necessary to express values to attach to measurements, or to provide a scale to measure some kinds of symptoms. For this reason, several Data Properties have been added:

1. Transmission rate (linked to the transmission process class), that is the speed with which a disease can be transmitted.

2. Pain, generally an expression of a symptom, the range of which has been expressed in low, medium or high.

3. Manifestation time, that is the time it takes for a symptom to manifest itself. This is expressed as slow, medium, fast.

4. Visibility represents how much a symptom manifests itself from an aesthetic point of view. For example, fever is an invisible symptom, but redness of the skin is.

5. Mortality: given a disease, this Data Property will give the percentage of the number of deaths caused by it.

6. Age affected: Given a disease, this Data Property will give the average age of patients affected by that disease.

7. Average life is a Data Property that allows to know the life expectancy. In particular, this property has as its domain in the kingdom of animals.

8. contagiousness: given a disease, this Data Property expressed through a Boolean value if the disease is contagious or not.

9. pharmacological effect: this Property specifically refers to drugs. Given a drug, it will give a parameter relating to the intensity of the pharmacological effect.

10. Sex affected: given a disease, this Data Property will provide information on the affected Sex.

11. Visibility: a Boolean parameter that can take two values, true or false depending on whether the disease has visible symptoms or not.

12. Plant origin: always referring to the "drugs" class. Given an individual it allows know the plant of origin of the active principle.

A.6 Population of the Ontology

Individuals are necessary to express rules and define an Expert System: without them, no inference would be actually possible. Several Individuals have been added, starting from the ChEBI class and relative subclasses, as enumerated in the following:

1. Gingko: a drug derived from gingko biloba leaves

2. Valerian: a drug derived from the leaves of "officinalis Valerian"

3. Ginseng: a drug derived from the roots of "Pana ginseng"

4. Opium: a drug derived from latex that comes out of "papaver somniferum" opium capsules

5. Digitalis: a drug derived from the leaves of "Digitalis Purpurea"

These Individuals have been also described with suitable Data Properties.
The Disease Class, which represents the largest part of the Ontology, has been populated with several Individuals, starting form different points of its hierarchical tree.
First of all, we started by adding COVID-19, in particular including five different variants:

1. B.1.1.7, UK variant, extremely contagious but no more lethal than the first variant of COVID-19;

2. 501-V2, South Africa variant, more contagious and deadlier than the first variant of COVID-19;

3. D614G, a dangerous mutant with about 13 times greater contagiousness than the first variant of COVID-19;

4. Cluster-5: Denmark variant spread among minks, which were culled to avoid contagion.

5. N501T, an Italian variant discovered by Arnaldo Caruso of the University of Brescia and Massimo Ciccozzi of the Biomedical Campus of Rome. Its characteristic is that it is extremely slow in healing, in fact a patient who is affected can be positive for months, although asymptomatic or in any case with few symptoms.

The importance of the Data Properties introduced in Section A.5 is clear: without them, these Individuals would just be names without any further information (apart for their belonging to the COVID-19 class). Figure A.5 shows one of the COVID Individuals, the 501-V2, with its instantiated Data Properties.

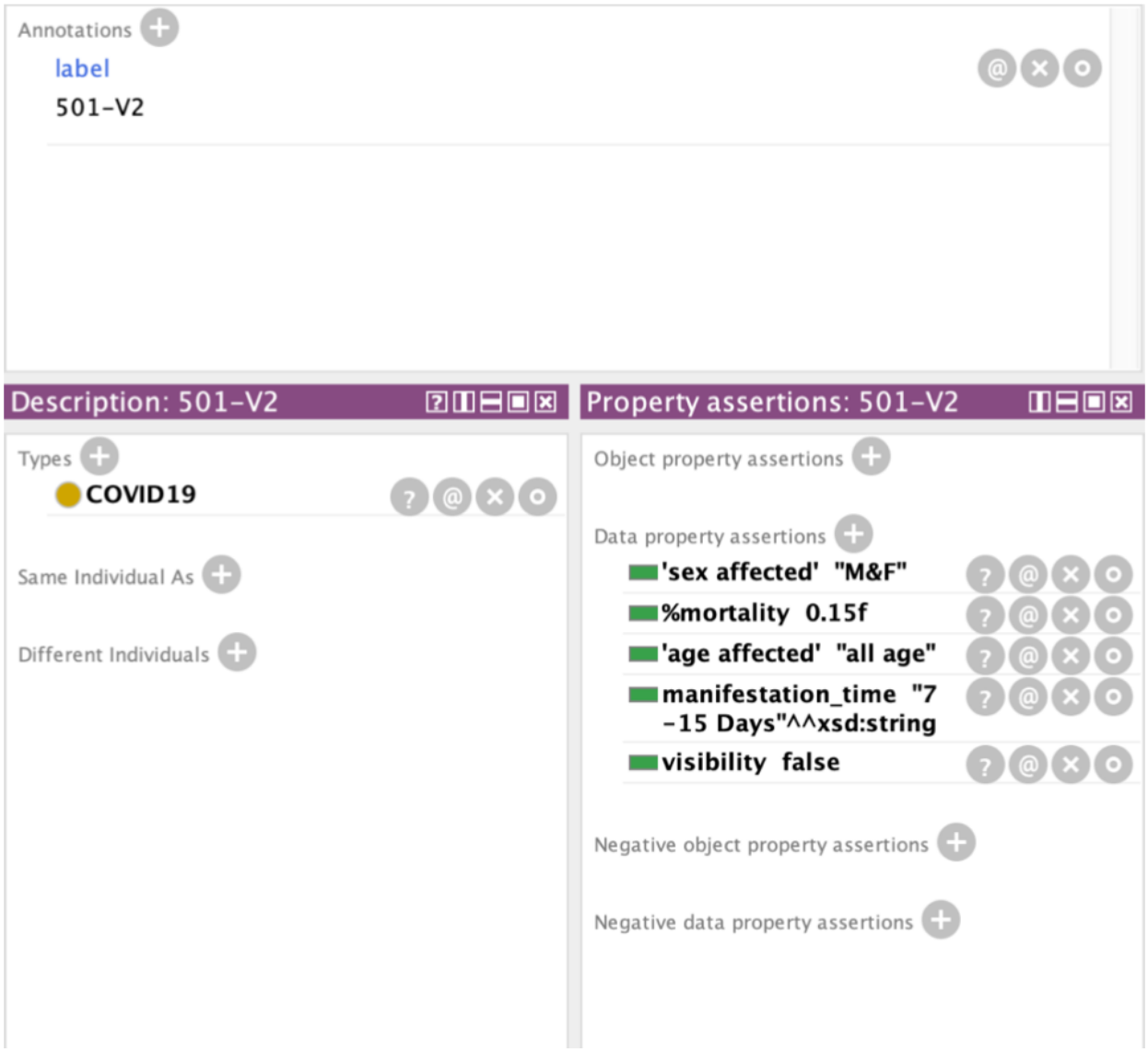


Figure A.5: The COVID 501-V2 Individual with its instantiated Data Properties.

Subsequently, the Osteoporosis class was populated. We can distinguish two individuals consisting in two types of osteoporosis, which are the following:

1. Postmenopausal osteoporosis: a disease that occurs in women within the first 20 years from the onset of menopause.

2. Senile osteoporosis: it can affect both sexes, but it mainly affects elderly people, especially people after 70 years of age

Again, Data Properties become relevant to distinguish the two Individual and to support a Rule based system. Figure A.6 shows the description of the Senile Osteoporosis Individual. It is clear

that, apart from the already described characteristics of the Disease, also the facts that it does not present visible symptoms and that is is not contagious is represented through the Data Properties.

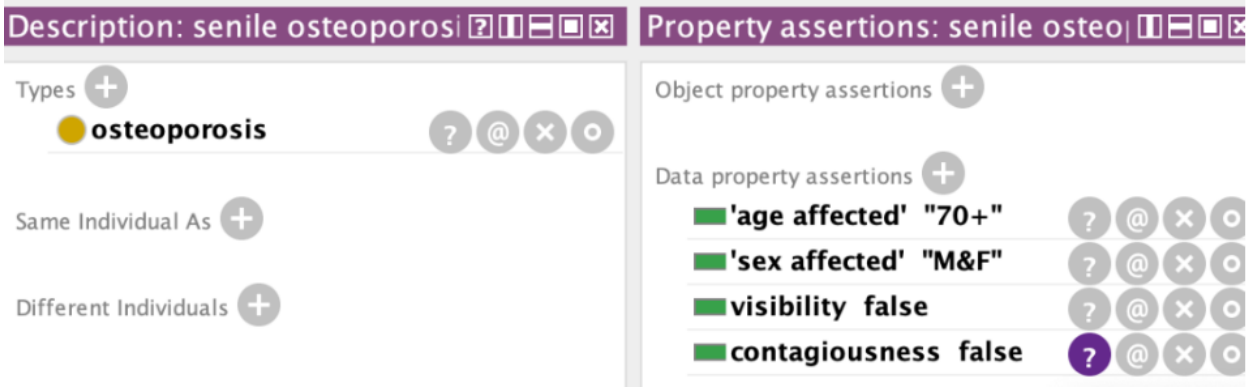


Figure A.6: Data Properties associated to the Senile Osteoporosis Individual

The last section of the disease we are going to populate are Gliomas, which account for about 40 percent of all primary brain Cancers. They include some sub types which are often named after the type of cell affected. They are given below here:

1. Diffuse astrocytomas: representing approximately 15 percent of all gliomas. The average age of Patients affected is 30-40 years.

2. Pilocytic astrocytomas: very rare form of cancer, accounting for about 5 percent of gliomas. However, they are the most common in children. Unlike other gliomas, they only rarely turn into a more aggressive cancer.

Since the class of Invertebrate Animals has been added, it was only natural to also populate it with some individuals. In particular, fours subclasses were taken in consideration:

1. Annelids, populated with the Individuals Red Earthworms and Polychaete annelids

2. Arthropods, to which the Individuals Crab and Butterfly were added.

3. Clam, where the Individuals Octopus and Cuttlefish were added.

A.7 Conclusion and Future Works

This paper describes the first step towards the creation of an Intelligent System able to support Doctors in their diagnosis. In order to compare different kind of approaches, such as Semantic based or Machine and Deep Learning oriented, an analysis of existing ontologies in the field of Health has been carried out, and the Disease Ontology has been selected as the most promising

one to realise a Rule based Expert System. Therefore, the Disease Ontology has been further examined, and augmented with new Classes, Object Properties, Data Properties and Individuals. In particular, the original absence of Data Properties has been considered an important missing piece of the Disease Ontology, as they can be used to provide useful information on a Disease, such as its infectivity, the main population affected, its mortality and lethality and so on.
The next step will consist in defining a set of Rules to build up the Expert System, which will then be tested and compared to Machine Learning approaches. Such a comparison will determine which of the two approaches will provide better results, or if an integration of both will be needed.

Appendix B

Automated Storytelling Technologies for cultural heritage

The awareness of the importance of communicating cultural heritage through innovative and automated means is steadily increasing. This awareness is fueled by the need to promote cultural heritage knowledge in accessible and engaging ways for an increasingly digital audience. This article aims to examine in detail the various methodologies used in the creation of automated storytelling for historical cultural heritage communication. In particular, attention is given to models based on large language models, such as those derived from advanced neural networks, which demonstrate significant potential in generating rich and engaging narratives. Another key aspect of the analysis concerns the use of chatbots in the context of automated storytelling. The possibilities offered by interactive conversation managed by virtual agents are explored, which can enhance the user experience through personalized dialogues and contextualized information. Additionally, the article focuses on various tools dedicated to automatic storytelling writing, evaluating their effectiveness and versatility in the specific context of historical cultural heritage. These tools are essential to facilitate the creative process and ensure narrative coherence. A significant part of the analysis addresses automated storytelling models based on processes, exploring how the sequence of events and concepts can be managed automatically to construct meaningful and relevant stories from a historical perspective. Furthermore, applications of ontologies and other tools are examined with the aim of improving the structure and understanding of narratives, ensuring effective linkage between different elements of historical cultural heritage. The overall goal of the article is to provide a comprehensive view of the current landscape of automated storytelling techniques, highlighting the still-open challenges and potential future development directions in this field.

B.1 Introduction

The introduction to automated storytelling technologies for cultural heritage unfolds as a captivating journey into the synergistic integration of various innovations, all aimed at enhancing and sharing the stories that enrich our historical and cultural heritage. Large Language Models (LLM) stand as fundamental pillars, capable of generating engaging and meticulously contextualized narratives through their advanced understanding of language. Concurrently, models based on semantics add layers of meaning and detail, enriching the narrative plot with a deeper comprehension of content. The ecosystem of automatic storytelling technologies also embraces the power of chatbots, intrinsically linked to advanced natural language models. These chatbots not only facilitate direct interaction with the audience, answering questions and providing detailed information, but they can also adapt to the context of cultural heritage, creating engaging and personalized conversations. In parallel, tools for automatic composition and writing emerge as allies in crafting unique narratives, allowing targeted customization and the formulation of stories that intricately weave with the fabric of cultural heritage. This convergence of technologies opens new perspectives, transforming automated storytelling into an art form that not only preserves and transmits our cultural heritage but does so in an accessible, engaging manner intrinsically linked to the expectations and preferences of the audience. Within this amalgamation of advanced models, semantics, chatbots, and composition tools, emerges a landscape of automated storytelling that harmoniously blends with the complexity of cultural narratives, promoting interactivity and active participation in this captivating exploration of our shared past.

B.2 Models

B.2.1 Models based on Large Language Models

The complexity of LLM has revolutionized automated storytelling, particularly in the context of cultural heritage communication and valorization. Models rooted in LLM play a crucial role in advancing automated storytelling for cultural heritage. Underpinning technologies like Natural Language Processing (NLP) empower models to comprehend implicit meanings within texts, extracting pertinent information from cultural heritage to enhance narratives with nuanced details and historical contexts. Simultaneously, Machine Learning (ML) optimizes models, enhancing their adaptability to cater dynamically to user preferences and specific contexts. The convergence of these technologies provides a robust platform for deploying advanced solutions, enabling the generation of storytelling that go beyond mere historical facts, creating an immersive experience through engaging and personalized stories. Additionally, LLM-based models integrate au-

tomatic translation functionalities, facilitating global appreciation of cultural heritage. Various transformer-based models, such as GPT, BERT, XLNet, and RoBERTa, play a crucial role in automated storytelling and cultural heritage enhancement, offering unique features for diverse applications. These models contribute to improved user experiences and the preservation of historical and cultural richness. To compare their effectiveness in cultural heritage storytelling, considerations include creativity, historical context understanding, computational efficiency, and adaptability to cultural sector needs. **GPT (Generative Pre-trained Transformer)** [232], a pre-trained transformer model, excels in creative text generation, capturing nuances in natural language. Despite requiring significant computational resources, its pre-training simplifies cultural heritage application development, providing adaptability across tasks [233]. **BERT (Bidirectional Encoder Representations from Transformers)** [234], known for bidirectional context understanding, is efficient for natural language processing and context-dependent tasks. It is more focused on context understanding than creative generation, making it suitable for creating stories based on historical information with fewer resource requirements. **XLNet (Transformer-XL Network)** [235], integrating bidirectional and long-term dependency capabilities, excels in understanding longer sequences, making it suitable for more detailed narratives. Despite demanding computational resources, its advantage lies in managing complex stories. **RoBERTa (Robustly optimized BERT approach)** [236], an optimized BERT implementation, maintains language understanding capabilities with a focus on efficiency. Like BERT, it is suitable for context understanding and story generation based on existing data, optimized for resource usage. In conclusion, the optimal model selection for automated storytelling in cultural heritage depends on specific application requirements. GPT showcase creativity and adaptability, while BERT and RoBERTa excel in historical context precision. XLNet's capacity for longer sequences suits comprehensive narratives. The decision should carefully balance creativity and historical accuracy, considering available computational resources and project requirements.

B.2.2 Chatbot

Interactive storytelling, in cultural heritage settings, is strongly supported by the use of chatbot technology. A chatbot is a conversational agent that, trained appropriately, provides the user with virtual guidance toward a structured, engaging, and coherent narrative in cultural contexts. Their strong development is certainly due to their clear ability to adapt the proposed content based on the conversations held with users. Chatbots based on **Google Dialogflow** or **RASA** demonstrate how, using NLP techniques, it is possible to create compelling narratives with valid and reliable content. Typically, chatbots rely on a series of intents, stories, and actions. An intent corresponds to a precise user request, and each intent, in the chatbot's knowledge base, corresponds to a set

of predetermined sentences (and their variants). Intent recognition is a key component, enabling the chatbot to understand the user's goals or requests. For instance, if a user shows interest in a particular artifact, the chatbot identifies the intent and adjusts the narrative accordingly. In the context of chatbots and stories, we are referring to pre-defined narrative sequences that the chatbot follows based on the user's messages, typically a series of recognized intents. These stories are crafted to lead the user through a meaningful and purposeful conversation [237]. With the ability to define paths that deviate from the main story, conversational agents can modify the narrative based on choices made by users, providing a personalized and interactive experience. Finally, actions represent operations that the conversational agent can perform as a result of the user's recognition of intent that involves more than simply sending a message. Actions can include executing code, calling APIs, interacting with third-party components, etc. Consider a conversational agent developed for guiding users through a museum tour. When a user expresses interest in a specific artwork, the chatbot can initiate a detailed narrative, covering aspects like the author, material, and artistic influences. During the interaction, using actions, the conversational agent can prompt other system components to play relevant audio or video files, enhancing the user's understanding of the artwork. Even with their great potential, the use of chatbots also presents challenges. Several weaknesses of this technology can range from potential limitations in user comprehension of input to the possible lack of context if the chatbot is not perfectly trained to the current technological limitation that mandates the use of dyadic conversations only [238]. Despite this, several advances in NLP and ML are underway, offering promising avenues for overcoming these limitations. As an example, the integration of traditional chatbots with emerging technologies such as LLM can contribute to the creation of more sophisticated and robust conversational agents capable of responding consistently even to user requests that perhaps do not sufficiently match any of the chatbot's intent, improving the overall user experience.

B.2.3 Semantic and Process-based Models

Several models of storytelling representation based on semantics are available in the literature. In particular, with the use of ontologies it is possible to provide better logical structuring of the story and greater interoperability of information, enabling better management and generation of content for storytelling. The study [239] introduces a storytelling ontology developed in Ontology Web Language (OWL), drawing inspiration from existing models like the ***Rhetorical Structure Theory Ontology*** [240] and the ***Common Narrative Model Ontology*** [241]. Furthermore, [239] presents an overview of utilizing process-based models to depict a story. Other works, such as [242, 243], contribute by presenting an ontology for narrative representation, accompanied by tools for constructing narratives based on the ontology and querying stories through a SPARQL

querying system. Meanwhile, [244, 245] outline a methodology that combines the process-based and semantic approaches, enabling the semantic annotation of structural elements in a process model (e.g., scenes, events, or actions) with domain concepts from an OWL ontology.

B.3 Tools for Automated Storytelling

B.3.1 Story Composition Tools

To develop automated storytelling systems, it is frequently essential to establish a structured, cohesive, and coherent representation of the narrative. Utilizing story composition tools can significantly aid scriptwriters in the creation process of storytelling. This section offers an overview of the cutting-edge story composition tools designed to assist scriptwriters, with a particular focus on those that provide a user-friendly graphical interface for crafting stories. The work [246] presents an web-based tool named **Egon.io**[1], which is a lightweigh tool that allows one to create models of domain storytelling to represent through graphical notation the functional requirements of a software system. Domain Storytelling is a collaborative, visual, and agile way to build domain-driven software. Egon.io is an open-source online tool for visualizing domain stories, that is an extension version of *BPMN.io*[2] that provide the possibility through a similar formalism to represent not only processes, but any graphical element. In fact, the tool allows the user to compose a story using the elements in a graphical palette, which can be customized according to the user's preferences. Upon completion of creation, the tool also offers a very simple story "run" mode. Figure B.1 shows an example of a domain story modeled with the Egon.io tool. As can be seen from Figure B.1, Egon.io allows to explicitly model scenes and scene changes, which usually occur when a change of location or time occurs, or when new characters enter the scene. The work [247] describes an open-source tool for telling interactive and nonlinear stories named **Twine**[3]. Twine enables users to build projects by incorporating pages or scenes that are interconnected through links, dictating the storyline's progression. Each scene is customizable with text, images, videos, or various multimedia elements. Users have the flexibility to insert decision points within scenes, altering the story's direction based on the choices made by the audience. Twine then generates an HTML file, which can be shared or published online, facilitating easy dissemination of the project to a broad audience.Twine is supported by a large community and the "*IFTF (Interactive Fiction Technology Foundation)*[4]", a nonprofit organization that provides infrastructure support for the interactive fiction community. Figure B.2 shows an example of a story, consisting of three scenes,

[1]https://egon.io/
[2]https://bpmn.io/
[3]https://twinery.org/
[4]https://iftechfoundation.org/

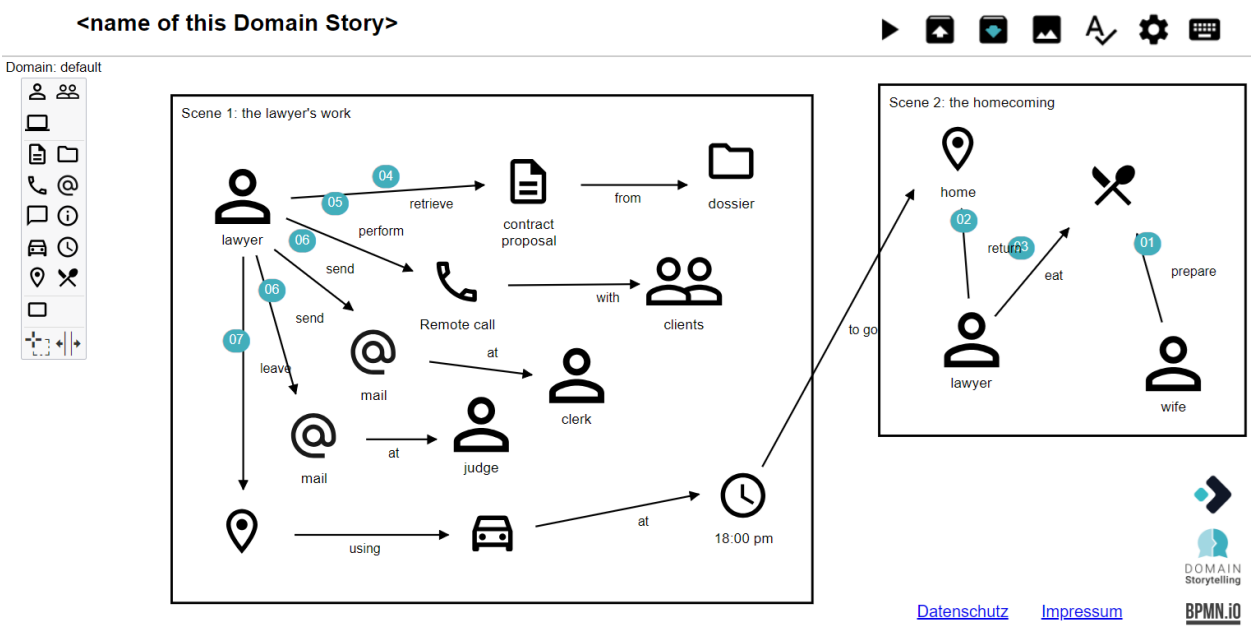


Figure B.1: Domain story modeled with the Egon.io Tool.

modeled using the Twine tool. Through the use of tags, Twine enables the definition of custom types of story elements. Several tools are employed in game design to facilitate story modeling, and among them, **Arcweave**[5] tool is one of the most widely used. As described by work [248], Arcweave is a cloud-based application designed for creating and prototyping games directly within a web browser. The game's storyline and logic are visualized through diagrams across multiple boards, connecting nodes and conditions. Subsequently, users can test the game prototype in play mode. Arcweave seeks to close the gap between writers and developers by implementing logic while maintaining a user-friendly interface for writers. Additionally, it facilitates real-time collaboration for teams of two or more. Figure B.3 shows an example of a story modeled using the Arcweave tool. One of the most commonly used tools for narrative design of a video game is **Articy**[6], described in work [249]. Articy is a versatile software application tailored for creating and organizing diverse narrative content, including screenplays, video game stories, drama scripts, and novel writing. Its intuitive user interface empowers creators to structure their narrative seamlessly, managing plot, character development, and environment creation efficiently. Notably, Articy automates the generation of reference and technical documentation for development teams, facilitating clear communication. Fostering real-time collaboration, Articy ensures easy synchronization of projects among creatives. It allows users to define custom palettes for actors, locations, and scene objects, providing a high degree of personalization. The tool also supports the creation of custom maps for locations, enabling users to specify the positions of scene objects and actors throughout

[5]https://arcweave.com/
[6]https://www.articy.com/en/

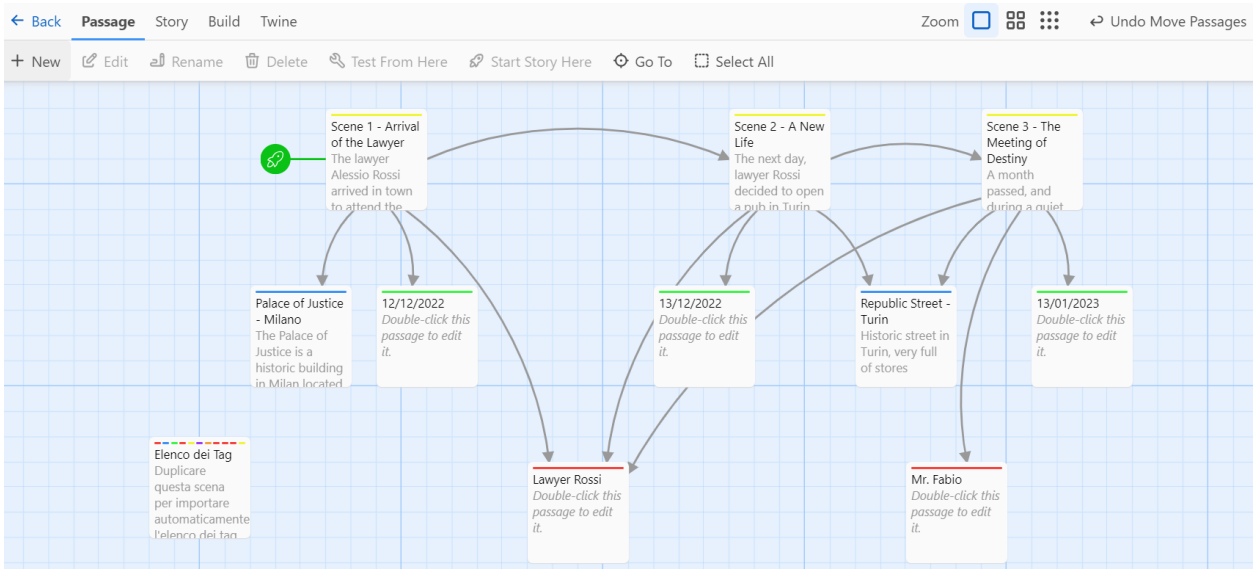


Figure B.2: Story modeled with the Twine Tool.

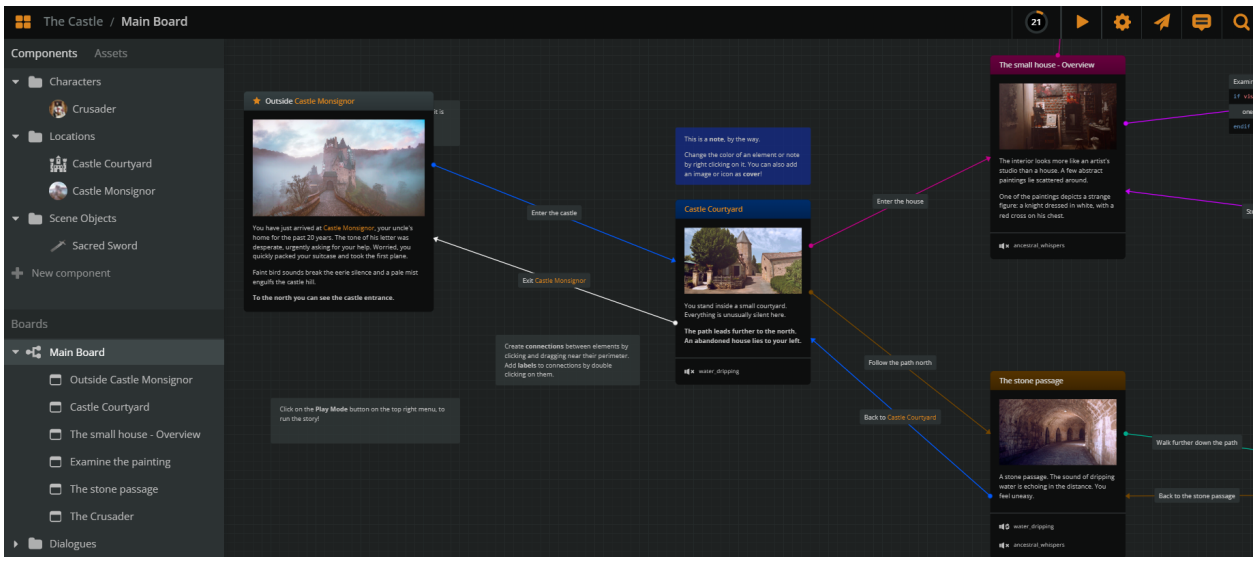


Figure B.3: Story modeled with the ArcWeave Tool.

the story. Figure B.4 shows an example of a story modeled using Articy. Table B.1 presents a

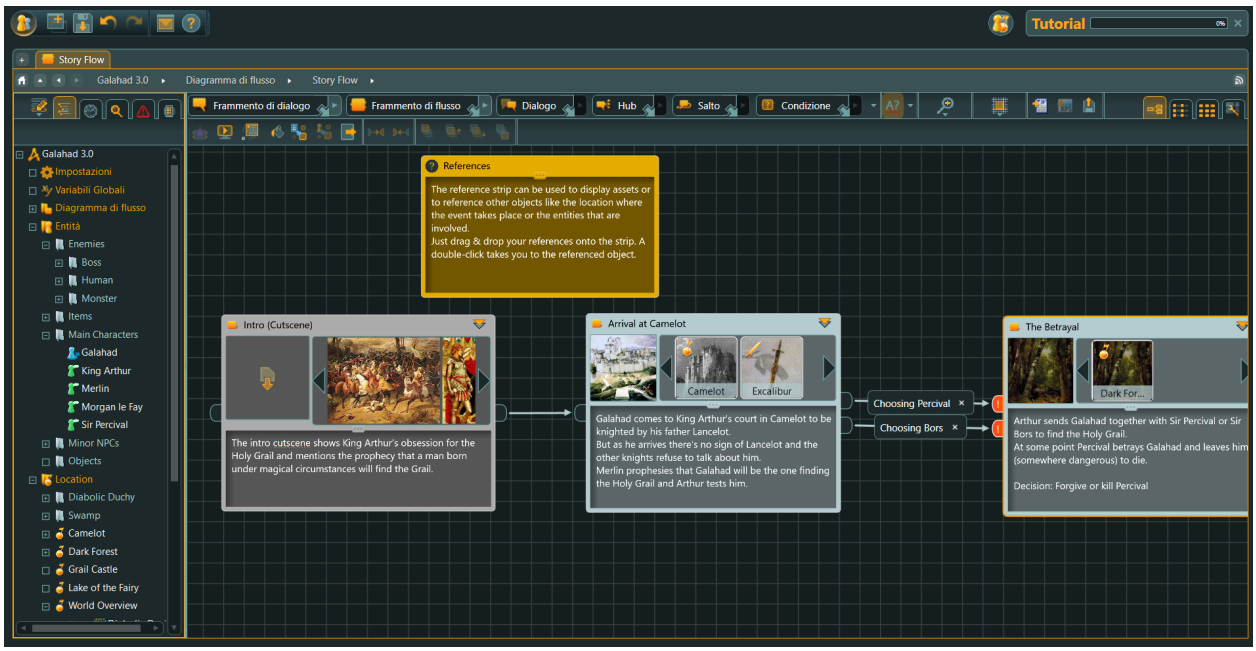


Figure B.4: Story modeled with the Articy Tool.

comparison of the features offered by the previously described tools.

B.3.2 Automatic Writing Tools

The strong demand for this type of tool has led to numerous efforts for the development of such tools both in the academic field and in industry. Among the main factors that support this process we must certainly mention: efficiency, as these tools allow the production of even large quantities of written content in a relatively short time. Furthermore, they are very versatile technologies capable of adapting to any context, from technical and formal writing to more creative and confidential writing. Finally, a final factor that has allowed the spread of this technology is accessibility: these tools are increasingly integrated into user-friendly platforms and consequently are easily accessible even to users who do not have a technical-scientific training. However, despite these advances, the development of such tools still faces numerous challenges, not only in the technical field but also from an ethical point of view. Certainly, among the most obvious problems is the lack of precision or coherence, especially in specialized contexts. Furthermore, there are also quite a few concerns regarding bias and privacy [250] and, last but not least, it should be emphasized that the use of such tools can reduce creativity and critical thinking in the long term. Currently, the market has many tools, some open-source, others with advanced functions accessible only for a fee. In order to illustrate the characteristics of some of the most used tools, a comparative analysis summarized in Table B.2 was carried out.

Table B.1: Comparison of Story Composition Tools

Feature	**Egon**	**Twine**	**Arcweave**	**Articy**
Creating Custom Story Elements		X	X	X
Modelling Story Flow	X	X	X	X
Representation of Times	X	X	X	X
Representation of Locations	X	X	X	X
Representation of Scene Objects	X	X	X	X
Play Mode	X	X	X	X
Creating Palettes of Actors and Locations			X	X
Real-time collaboration for teams			X	X
Define World's Map and the Pathways of History				X
Mapping Actors and Scene Objects to Locations during Story				X

Capability	Smodin	ChatGPT	Grammarly	DeepL	Bard AI	QuillBot
Question Answering	X	X			X	
Text Generation	X	X	X		X	X
Translation	X	X	X	X	X	X
Text Re-elaboration	X	X	X	X	X	X
Grammatical Correction	X	X	X	X	X	X
Plagiarism Check	X		X			X
Text Summarization	X	X			X	X
Stylistic Suggestions	X	X	X	X	X	X

Table B.2: Comparison of capabilities among various Automatic Writing Tools.

The comparison between the different writing platforms highlights a constant trend which is the continuous increase in the set of functions available for each tool. However, this homogeneity in the provision of these functions does not make the tools equivalent since factors such as accuracy, precision or recall stand out as critical in evaluating the effectiveness of these tools. While all platforms adhere to a high standard of service, there are notable distinctions and specialized areas

of expertise in managing tasks.

B.4 Conclusion and Future Works

In conclusion, the importance of communicating cultural heritage through innovative and automated means has gained significant attention due to the increasing demand for accessible and engaging ways to promote cultural heritage knowledge for a digital audience. This article has explored various methodologies used in the creation of automated storytelling for historical cultural heritage communication, with a focus on models based on LLM and chatbots, automatic storytelling writing tools, semantic-based and process-based models for storytelling and story composition tools. As technology continues to advance, it is imperative that we continue to explore new and innovative ways to promote cultural heritage knowledge, ensuring that it remains accessible and engaging for future generations.

Appendix C

Artificial Intelligence Techniques for Dynamic Offloading in Cloud Continuum Environment: A Review

In this work we are going to present the most recent progress on the topic of dynamic offloading guided by artificial intelligence (AI) techniques. The growing demands in terms of performance on edge devices often require very high computational capabilities. In spite of the progress made in hardware power, some applications, such as collaborative augmented reality (AR) and virtual reality (VR) or particular needs in medical or military sectors, require rapid response times which put a strain on the performance of the devices. In these contexts, the use of dynamic offloading in a fog and edge computing environment emerge an effective solution to increase the performance of the architectures. In the first part of this review we aim to analyze the enabling technologies, objectives, strategies and opportunities of dynamic offloading. After, based on this investigation and with the analysis of different algorithms developed, we would like to show how artificial intelligence can play a crucial role in this context, in particular with the use of machine learning (ML), deep learning (DL) or reinforcement learning (RL) models also with the use of hybrid solutions.

C.1 Introduction

In today's world, the internet influences almost every aspect of our daily lives and the utilization of IoT devices has become indispensable in many contexts. Despite the increase in their computational capacity, mobile devices still have limitations compared to fixed hardware resources. Big tech companies (such as Amazon, Google, IBM and others) have constantly invested in research to develop ever better cloud solutions. However, in some sectors such as the military or healthcare

ones, many of the applications (e.g. virtual or augmented reality, remote surgery, military operations) require rapid response times which challenge to traditional cloud center solutions. In fact, the transfer of large volumes of data can represent a bottleneck, even in an excellent network context. In this context, edge and fog computing technology represents a key point since, despite having reduced computational capabilities, it allows operations to be carried out faster as it is closer to the user rather than the server. This technology is gaining more and more ground and, in this field, dynamic offloading represents a great challenge as it aims to shift the computation load depending on the context for different needs which will be explained in detail later in the document. Also in recent years, AI technology, which thanks to its versatility is able to solve problems in numerous contexts, is advancing its state of the art also in the context of dynamic offloading. It is proving to be an alternative to classic approaches to rules. In this document we therefore ask and attempt to answer the following research questions:

- **RQ1**: What are the enabling technologies of dynamic offloading?
- **RQ2**: What are the strategies and metrics of dynamic offloading?
- **RQ3**: What is the state of the art of current advances in the field of artificial intelligence applied to dynamic offloading?

To accomplish this, the paper is structured as follows: in the next chapter, we will discuss about the enabling technologies for dynamic offloading, analysing why they are needed. Then, it will be discussed what are the metrics and the strategies of dynamic offloading in a cloud fog edge scenario. Finally, in the last chapter, an overview of algorithms using dynamic offloading with artificial intelligence techniques will be given and conclusions for further work will be drawn, providing important insights for the scientific community.

C.2 Enabling technologies

In general, there are numerous enabling technologies for dynamic offloading. Among these, the one that certainly plays a key role is cloud and edge computing: this technology provides remote platforms capable of hosting computation. While cloud computing typically moves all the processing to a central server, edge computing hosts the computation at the edge of the network. So, it guarantees the speed of response because it does not have to refer to a central server and also the security of the data since information is not shared directly with central server but it remains on the device. An example where this distributed approach turns out to be a correct choice is federated learning [16]. On the other hand, the traditional cloud computing takes care of transferring all the data to the central server. Most machine learning and deep learning algorithms still use this

approach. Another enabling technology is certainly the communication network. The power and type of the network represent a crucial factor for the communication aspect of the various tasks to be performed, in particular with the advent of 5G [115] [251] which guarantees low latency and is an essential point for dynamic offloading. In many cases, even if the computational power and architecture are high-performance, the network can represent a bottleneck for system performance. Furthermore, for the subdivision of the various tasks containerization and virtualization, more precisely, technologies such as Docker and Kubernetes which package and distribute container applications and virtual machines are fundamental in this context [252] as they facilitate the execution of downloaded activities on different platforms, solving the problem of not having to worry about dependencies arising from the execution environment. In addition, mention should also be made of communication protocols and APIs. There are fundamental for the exchange of information as they provide rules and specifications necessary for the exchange of data between the various tasks; examples include REST (Representational State Transfer) for simple yet efficient web APIs, MQTT (Message Queuing Telemetry Transport) for IoT (Internet of Things) device communication, and gRPC for high-performance communications between microservices. Finally, last but not least, there are the distributed processing platforms (e.g. Hadoop and Spark) which allow to distribute the workload across multiple devices.

C.3 Overview of Offloading Strategies

Offloading is a process whereby a smartphone, a mobile device or even more generally any IoT device delegates computationally intensive operations to a more powerful server or to another device. The first distinction to be made about offloading is the difference between dynamic and static. This difference depends mainly on the timing of the offloading decision. While static offloading methods use models to estimate performance and employ a client-server approach, dynamic offloading policies divide the architecture into 3 cloud-fog-edge layers Fig C.1 and models are continuously updated based on progressive profiling on individual devices.

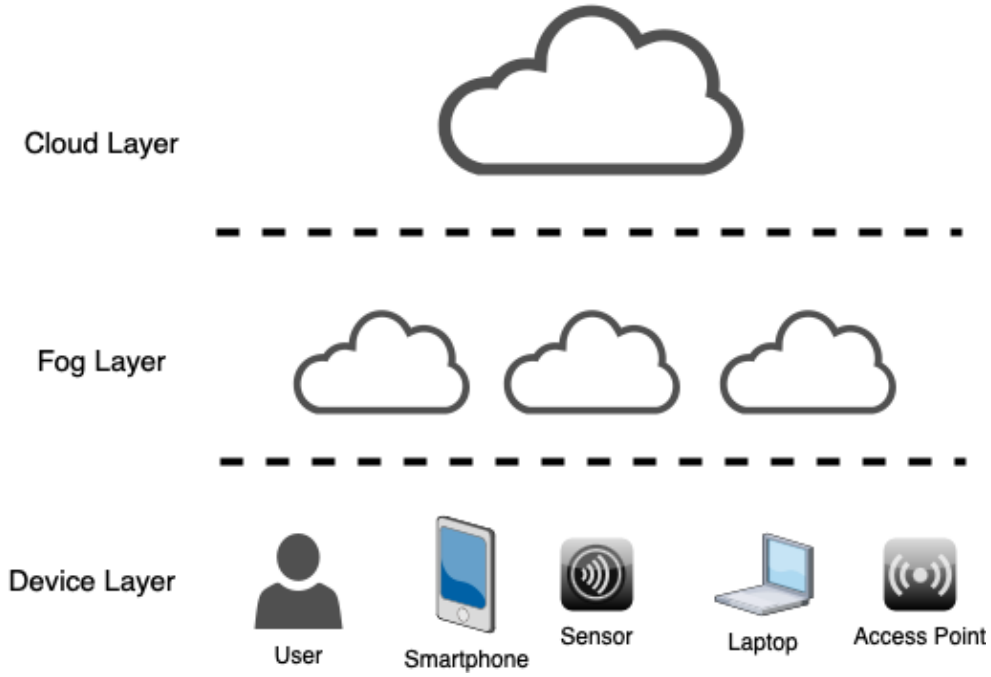


Figure C.1: Cloud-Fog-Edge Architecture

In a context of this type, offloading can occur in different ways:

- **Device to Device**: The first case is when offloading takes place on the same layer. Here, the device for various reasons (either because of limited resources or because it is busy doing something else) transfers the task to be performed to another device which at that moment has the possibility of performing these operations.
- **Device to Fog / Fog to Device**: The second case occurs when the device, unable to perform an operation for various reasons, decides to send the task to be performed directly to an edge server located in the fog computing layer, which tends to have more resources than the individual device. Obviously, the edge server, although more performing than the device, always has limited resources compared to the Cloud Server.
- **Fog to Server / Server to Fog**: The last case occurs when the power or storage capacity of the edge located in the fog are not sufficient. Therefore, it sends the task directly to the cloud which potentially enjoys privileges, such as scalability and adaptability. So, although the edge servers are capable of handling many tasks when needed, these operations can be further offloaded to the cloud.

In general, there is no single objective for which offloading techniques are used, but they certainly all have the common goal of reducing the load on resource-constrained devices. This results in an

improvement of the overall experience and overall system efficiency. Going into more detail, the metrics aimed at improving dynamic offloading are:

- **Bandwidth**: Effective bandwidth management is essential to optimize offloading performance. For large data transfers, high bandwidth is required to avoid bottlenecks. [253]
- **Energy**: it is needed in the offloading process for sending, executing, and returning results. This means that it is essential to maximize battery life especially in mobile and IoT devices.[254]
- **Reliability**: it is mainly influenced by system processing times and latency. It represents the entire duration of both sending and processing action [255].
- **Latency**: this in offloading involves the time spent transmitting tasks to edge servers or other clients, executing them, and returning the results. To minimize it, different types of delays need to be managed. [256]
- **Security**: communications, sending tasks and data from one machine to another can cause security problems. In these circumstances it is therefore necessary to guarantee the protection of information, of any nature, during its sending [257].
- **Cost**: it is influenced by parametrics such as: the transmission and processing of the tasks, the location of the task or the response time.

In addition, an important distinction which must be made is the difference between applications that use artificial intelligence techniques, such as ML (Machine Learning)[258], DL (Deep Learning) [77], RL(Reinforcement Learning) and those that do not. In the former case, we talk about applications that use learning algorithms to improve the decisions to be made at various stages of offloading. Artificial intelligent models, with the analyses of datas, are able to identify contextual behavioral patterns in order to predict the need how to optimize the offloading process. As we have seen previously, there are different objectives and, depending on what you want to optimize, the model is trained to be able to respond to specific needs. A rule-based approach, on the other hand, is based on techniques that are more predetermined by algorithms for choosing the decisions to be made(e.g. fixed rules or scheduling algorithms).

C.4 AI-Assisted Dynamic Offloading Algorithm

Over the years dynamic offloading in the context of distributed elaboration has also gained more and more ground with the spread of edge computing. At the same time, artificial intelligence has

also gained ground due to its versatility in uses that allows it to be applied in almost all fields with good results. The following table presents an overview of the most recent developments in the field of algorithms of dynamic offloading supported by artificial intelligence models sorted by type of learning. These algorithms, although with different objectives and requirements that will be summarised under in the table C.1(such as response time, latency optimisation, energy saving), have been developed to optimise offloading choices.

Table C.1: Overview of AI Algorithms and Objectives of Dynamic Offloading

Algorithm Name	**AI Type**	**Algorithm**	**Objective**
S-OAMC (Sampling-based Optimization for Application Migration in Cloudlets) and G-OAMC (Greedy-based Optimization for Application Migration in Cloudlets)	Machine Learning	Matrix Completion	Ensure that applications can run with the least turnaround time for the purpose of improving user experience [259]
MALMOS (Machine Learning-based Mobile Offloading Schedule)	Machine Learning	Classifier	Optimize the offloading process for mobile applications to improve scheduling performance over static policies based on thresholds and linear equations

Continued on next page

Table C.1 – continued from previous page

Algorithm Name	**AI Type**	**Algorithm**	**Objective**
CSOS (Context-Sensitive Offloading System)	Machine Learning	JRIP, IBK and Naive Bayes	The machine learning classifiers is used to ensure high accuracy in offloading decisions specifically by determining whether the execution of a workload is more efficient locally or on a remote server based on context [260]
DSLO (Deep Supervised Learning-based Offloading)	Deep Learning	CNN (Convolutional Neural Network), DNN (Dense Neural Network)	The goal is to make near-optimal offloading decisions on the one hand by maximizing system utility and on the other hand by minimizing CPU execution latency in MEC networks [261]
IRAO (Intelligent Radio Access Optimization)	Deep Learning	DNN (Dense Neural Network) With SynDNN and AsyDNN	It aims to improve the computational performance of IoT (Internet of Things) systems, especially those constrained by limited resources such as industrial sensors [262]

Continued on next page

Table C.1 – continued from previous page

Algorithm Name	**AI Type**	**Algorithm**	**Objective**
SecOFF-FCIoT (Secure Computation Offloading Scheme in Fog-Cloud-IoT Environment)	Deep Learning	Neuro-Fuzzy model	By means of the layered architecture for the offloading scheme, secure security is put first while balancing latency and power consumption [263]
C-GCN-DDPG-Agent (Graph Convolutional Network-Deep Deterministic Policy Gradient)	Hybrid	GCN (Graph Convolutional Network), DDPG (Deep Deterministic Policy Gradient)	Minimize the long-term weighted average cost, which includes both task completion time and energy consumption on mobile devices, taking into account the dynamism of the task and the environment.[264]
SAGIN (Space-Air-Ground Integrated Networks)	Hybrid	Double Q-learning and DSRPM	The goal is to reduce packet loss, increase network throughput and consequently improve delivery time[265]

Continued on next page

Table C.1 – continued from previous page

Algorithm Name	**AI Type**	**Algorithm**	**Objective**
CO and RP for EC (Computation Offloading and Resource Provisioning for Edge-Cloud Computing Environment)	Hybrid	Long Short-Term Memory (LSTM), Q-learning, Learning Automata (LA)	Addressing the continuous demands due to the proliferation of technologies such as: AR, VR, IVC and AAL. In this context, a proposed solution is to take into account the edge server provisioning problem based on the dynamic nature of mobile devices and the unpredictable edge computing ecosystem [266]
DMRO (Distributed Meta Reinforcement Offloading)	Hybrid	DNNs and Q-learning	Improving task offloading decisions in collaborative edge-cloud environments by reducing costs and increasing portability, the ability to quickly learn the environment and identify optimal offloading solutions with a reduced number of learning steps [267]

The table shows a general picture of the latest developments in offloading algorithms supported by AI mechanisms, underlining the diversity of both approaches and objectives. The algorithms are sorted by the type of learning: machine learning, deep learning, reinforcement learning or hybrid approaches. Each algorithm uses optimal techniques and approaches for different contexts: DSLO

and IRAO are particularly effective in reducing latency and increasing computational performance, others, such as C-GCN-DDPG-Agent, focus on reducing energy consumption while maintaining optimal task completion time; S-OAMC and G-OAMC aim to optimize execution times, while SecOFF-FCIoT has security as its main objective. Overall, the different goals of the different algorithms summarized in this chapter show the wide range of offloading challenges. What is certain is that the integration of these algorithms in the real world can improve resource management and consequently improve the user experience.

C.5 Conclusion

The work done in this document was to provide a general overview of the main enabling technologies, metrics, strategies and an recap of the AI techniques used to drive dynamic offloading algorithms. The aim is to understand if, in this context, artificial intelligence can be considered a correct choice. From the analysis carried out, it is clear how the trend in the use of these methodologies, especially in recent years, have led to valid alternatives compared to the classic rule-based approaches. However, it should be underlined that approaches of this type are not to be considered as a better or worse choice. What determines it is not the methodology but the context on which it is applied. This document could be useful for anyone who intends to approach to the topic of dynamic offloading supported by AI techniques

Bibliography

[1] S. Moreschini, F. Pecorelli, X. Li, S. Naz, D. Hästbacka, and D. Taibi, "Cloud continuum: the definition," *IEEE Access*, vol. 10, pp. 131 876–131 886, 2022.

[2] D. Aguiari, A. Ferlini, J. Cao, S. Guo, and G. Pau, "C-continuum: Edge-to-cloud computing for distributed ai," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 1053–1054.

[3] A. Ali, Y. Ming, S. Chakraborty, and S. Iram, "A comprehensive survey on real-time applications of wsn," *Future internet*, vol. 9, no. 4, p. 77, 2017.

[4] C. Bachhuber, A. S. Martinez, R. Pries, S. Eger, and E. Steinbach, "Edge cloud-based augmented reality," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2019, pp. 1–6.

[5] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[6] B. U. Demirel, I. A. Bayoumy, and M. A. Al Faruque, "Energy-efficient real-time heart monitoring on edge–fog–cloud internet of medical things," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 472–12 481, 2021.

[7] E. L. Wisotzky, J.-C. Rosenthal, P. Eisert, A. Hilsmann, F. Schmid, M. Bauer, A. Schneider, and F. C. Uecker, "Interactive and multimodal-based augmented reality for remote assistance using a digital surgical microscope," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2019, pp. 1477–1484.

[8] T. K. Dang, X. Lan, J. Weng, and M. Feng, "Federated learning for electronic health records," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 5, pp. 1–17, 2022.

[9] E. Bauer, "Improving operational efficiency of applications via cloud computing," *IEEE Cloud Computing*, vol. 5, no. 1, pp. 12–19, 2018.

[10] K. Hu, Y. Li, M. Xia, J. Wu, M. Lu, S. Zhang, and L. Weng, "Federated learning: a distributed shared machine learning method," *Complexity*, vol. 2021, no. 1, p. 8261663, 2021.

[11] S. Heiler, "Semantic interoperability," *ACM Computing Surveys (CSUR)*, vol. 27, no. 2, pp. 271–273, 1995.

[12] H. Gupta, S. B. Nath, S. Chakraborty, and S. K. Ghosh, "Sdfog: A software defined computing architecture for qos aware service orchestration over edge devices," *arXiv preprint arXiv:1609.01190*, 2016.

[13] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of things journal*, vol. 3, no. 6, pp. 854–864, 2016.

[14] K. T. Kim, C. Joe-Wong, and M. Chiang, "Coded edge computing," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 237–246.

[15] A. Amato, B. Di Martino, and S. Venticinque, "A distributed cloud brokering service," *Informatica*, vol. 26, no. 1, pp. 1–15, 2015.

[16] G. J. Pezzullo, A. Esposito, and B. di Martino, "Federated learning of predictive models from real data on diabetic patients," in *International Conference on Advanced Information Networking and Applications*. Springer, 2023, pp. 80–89.

[17] M. R. KARTHIKEYAN, M. M. RAMKUMAR, and M. S. MOHANAPRIYA, "Incorporation of edge computing through cloud computing technology," *the International Research l Journal of Engineering and Technology*, vol. 7, no. 9, pp. 2395–0056, 2020.

[18] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE international conference on smart cloud (SmartCloud)*. IEEE, 2016, pp. 20–26.

[19] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2019.

[20] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[21] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[22] X. Lin, J. G. Andrews, A. Ghosh, and R. Ratasuk, "An overview of 3gpp device-to-device proximity services," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 40–48, 2014.

[23] T. M. Mitchell and T. M. Mitchell, *Machine learning*. McGraw-hill New York, 1997, vol. 1, no. 9.

[24] L. P. Zhuhadar and M. D. Lytras, "The application of automl techniques in diabetes diagnosis: current approaches, performance, and future directions," *Sustainability*, vol. 15, no. 18, p. 13484, 2023.

[25] A. L. Samuel, "Machine learning," *The Technology Review*, vol. 62, no. 1, pp. 42–45, 1959.

[26] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[27] J. M. Norman, "Mcculloch & pitts publish the first mathematical model of a neural network."

[28] L. N. Kanal, "Perceptron," in *Encyclopedia of Computer Science*, 2003, pp. 1383–1385.

[29] P. D. by Raytheon, S. S. ROCKET, and M.-H. See, "Abbreviation for control test vehicle. cutwater (navy) project to study nonacoustic means for detecting submarines, ie, ir, electromagnetic, and other approaches. cybertron," 1962.

[30] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman, "Autoclass: A bayesian classification system," in *Machine learning proceedings 1988*. Elsevier, 1988, pp. 54–64.

[31] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[32] M. W. Berry, A. Mohamed, and B. W. Yap, *Supervised and unsupervised learning for data science*. Springer, 2019.

[33] X. Zhu, T. Rogers, R. Qian, and C. Kalish, "Humans perform semi-supervised classification too," in *AAAI*, vol. 2007, 2007, pp. 864–870.

[34] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning: a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, 2006.

[35] B. Kolo, *Binary and multiclass classification*. Lulu. com, 2011.

[36] İ. Uysal and H. A. Güvenir, "An overview of regression techniques for knowledge discovery," *The Knowledge Engineering Review*, vol. 14, no. 4, pp. 319–340, 1999.

[37] G. W. Flake and S. Lawrence, "Efficient svm regression training with smo," *Machine learning*, vol. 46, pp. 271–290, 2002.

[38]

[39] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora, "Decision tree regression for soft classification of remote sensing data," *Remote Sensing of Environment*, vol. 97, no. 3, pp. 322–336, 2005.

[40] A. M. Ahmed, A. Rizaner, and A. H. Ulusoy, "A decision tree algorithm combined with linear regression for data classification," in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*. IEEE, 2018, pp. 1–5.

[41] T. Rahmat, A. Ismail, and S. Aliman, "Chest x-rays image classification in medical image analysis," *Applied Medical Informatics*, vol. 40, no. 3-4, pp. 63–73, 2018.

[42] M. R. Zare, W. C. Seng, and A. Mueen, "Automatic classification of medical x-ray images," *Malaysian Journal of Computer Science*, vol. 26, no. 1, pp. 9–22, 2013.

[43] M. Assam, H. Kanwal, U. Farooq, S. K. Shah, A. Mehmood, and G. S. Choi, "An efficient classification of mri brain images," *IEEE Access*, vol. 9, pp. 33 313–33 322, 2021.

[44] T. C. Phan *et al.*, "Tumor segmentation and classification using machine learning approaches," *International Journal of Data Informatics and Intelligent Computing*, vol. 3, no. 1, pp. 1–11, 2024.

[45] W. A. Awad and S. ELseuofi, "Machine learning methods for spam e-mail classification," *International Journal of Computer Science & Information Technology (IJCSIT)*, vol. 3, no. 1, pp. 173–184, 2011.

[46] E. G. Dada, J. S. Bassi, H. Chiroma, A. O. Adetunmbi, O. E. Ajibuwa *et al.*, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, 2019.

[47] H. Wang and H. Zhang, "Movie genre preference prediction using machine learning for customer-based information," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 110–116.

[48] C. Jin, L. De-Lin, and M. Fen-Xiang, "An improved id3 decision tree algorithm," in *2009 4th international conference on computer science & Education*. IEEE, 2009, pp. 127–130.

[49] S. Ruggieri, "Efficient c4. 5 [classification algorithm]," *IEEE transactions on knowledge and data engineering*, vol. 14, no. 2, pp. 438–444, 2002.

[50] D. G. Denison, B. K. Mallick, and A. F. Smith, "A bayesian cart algorithm," *Biometrika*, vol. 85, no. 2, pp. 363–377, 1998.

[51] V. Vapnik, "Statistical learning theory," *John Wiley & Sons google schola*, vol. 2, pp. 831–842, 1998.

[52] A. Mathur and G. M. Foody, "Multiclass and binary svm classification: Implications for training and classification users," *IEEE Geoscience and remote sensing letters*, vol. 5, no. 2, pp. 241–245, 2008.

[53] U. Khandelwal, A. Fan, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Nearest neighbor machine translation," *arXiv preprint arXiv:2010.00710*, 2020.

[54] C. Duyckaerts and G. Godefroy, "Voronoi tessellation to study the numerical density and the spatial distribution of neurones," *Journal of chemical neuroanatomy*, vol. 20, no. 1, pp. 83–92, 2000.

[55] V. Priyadarshni, A. Nayyar, A. Solanki, and A. Anuragi, "Human age classification system using k-nn classifier," in *Advanced Informatics for Computing Research: Third International Conference, ICAICR 2019, Shimla, India, June 15–16, 2019, Revised Selected Papers, Part I 3*. Springer, 2019, pp. 294–311.

[56] P. Langley, W. Iba, K. Thompson *et al.*, "An analysis of bayesian classifiers," in *Aaai*, vol. 90. Citeseer, 1992, pp. 223–228.

[57] D. Berrar, "Bayes' theorem and naive bayes classifier," 2019.

[58] K. Alghatani, N. Ammar, A. Rezgui, A. Shaban-Nejad *et al.*, "Predicting intensive care unit length of stay and mortality using patient vital signs: machine learning model development and validation," *JMIR medical informatics*, vol. 9, no. 5, p. e21347, 2021.

[59] E. D. Dan, O. Jude, and N. A. Chukwukailo, "Application of multivariate multiple linear regression model on vital signs and social characteristics of patients," *Int J Eng Sci*, vol. 2, pp. 40–56, 2013.

[60] V. Daliya, T. Ramesh, and S.-B. Ko, "An optimised multivariable regression model for predictive analysis of diabetic disease progression," *IEEE Access*, vol. 9, pp. 99 768–99 780, 2021.

[61] L. Perreault, Q. Pan, K. J. Mather, K. E. Watson, R. F. Hamman, and S. E. Kahn, "Effect of regression from prediabetes to normal glucose regulation on long-term reduction in diabetes risk: results from the diabetes prevention program outcomes study," *The Lancet*, vol. 379, no. 9833, pp. 2243–2251, 2012.

[62] Z. Gu, Y. Li, M. Zhang, and Y. Liu, "Modelling economic losses from earthquakes using regression forests: Application to parametric insurance," *Economic Modelling*, vol. 125, p. 106350, 2023.

[63] W. Liu, X. Li, Z. Chen, G. Zeng, T. León, J. Liang, G. Huang, Z. Gao, S. Jiao, X. He *et al.*, "Land use regression models coupled with meteorology to model spatial and temporal variability of no2 and pm10 in changsha, china," *Atmospheric Environment*, vol. 116, pp. 272–280, 2015.

[64] A. Paniagua-Tineo, S. Salcedo-Sanz, C. Casanova-Mateo, E. Ortiz-García, M. Cony, and E. Hernández-Martín, "Prediction of daily maximum temperature using a support vector regression algorithm," *Renewable Energy*, vol. 36, no. 11, pp. 3054–3060, 2011.

[65] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[66] J. R. Rodríguez Mejia, A. Woocay Prieto, A. Valles Chávez, R. Villela Varela, F. E. López Monteagudo, and C. Reyes Rivas, "Estimation of solar radiation in northwest mexico based on the angstrom model and polynomial regression," *Ingeniería Energética*, vol. 43, no. 1, pp. 35–47, 2022.

[67] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013.

[68] M. E. Celebi and K. Aydin, *Unsupervised learning algorithms*. Springer, 2016, vol. 9.

[69] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[70] S. Banerjee, A. Choudhary, and S. Pal, "Empirical evaluation of k-means, bisecting k-means, fuzzy c-means and genetic k-means clustering algorithms," in *2015 IEEE international WIE conference on electrical and computer engineering (WIECON-ECE)*. IEEE, 2015, pp. 168–172.

[71] S. K. Uppada, "Centroid based clustering algorithms—a clarion study," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 6, pp. 7309–7313, 2014.

[72] L. Qian, C. Plant, and C. Böhm, "Density-based clustering for adaptive density variation," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 1282–1287.

[73] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[74] S. Kärnä, V.-M. Sorvala, and J.-M. Junnonen, "Classifying and clustering construction projects by customer satisfaction," *Facilities*, vol. 27, no. 9/10, pp. 387–398, 2009.

[75] S. Xie, "Defining geographical rating territories in auto insurance regulation by spatially constrained clustering," *Risks*, vol. 7, no. 2, p. 42, 2019.

[76] H. Wang and B. Raj, "On the origin of deep learning," *arXiv preprint arXiv:1702.07800*, 2017.

[77] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019.

[78] S. Fitz and P. Romero, "Neural networks and deep learning: A paradigm shift in information processing, machine learning, and artificial intelligence," *The Palgrave Handbook of Technological Finance*, pp. 589–654, 2021.

[79] J. Gupta, S. Pathak, and G. Kumar, "Deep learning (cnn) and transfer learning: a review," in *Journal of Physics: Conference Series*, vol. 2273, no. 1. IOP Publishing, 2022, p. 012029.

[80] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, "Exploiting image-trained cnn architectures for unconstrained video classification," *arXiv preprint arXiv:1503.04144*, 2015.

[81] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

[82] M. Radovic, O. Adarkwa, and Q. Wang, "Object recognition in aerial images using convolutional neural networks," *Journal of Imaging*, vol. 3, no. 2, p. 21, 2017.

[83] M. Coşkun, A. Uçar, Ö. Yildirim, and Y. Demir, "Face recognition based on convolutional neural network," in *2017 international conference on modern electrical and energy systems (MEES)*. IEEE, 2017, pp. 376–379.

[84] M. Aladem and S. A. Rawashdeh, "A single-stream segmentation and depth prediction cnn for autonomous driving," *IEEE Intelligent Systems*, vol. 36, no. 4, pp. 79–85, 2020.

[85] R. Chawla, S. M. Beram, C. R. Murthy, T. Thiruvenkadam, N. Bhavani, R. Saravanakumar, and P. Sathishkumar, "Brain tumor recognition using an integrated bat algorithm with a convolutional neural network approach," *Measurement: Sensors*, vol. 24, p. 100426, 2022.

[86] H. Shi, N. Zhang, X.-q. Wu, and Y.-D. Zhang, "Multimodal lung tumor image recognition algorithm based on integrated convolutional neural network," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 21, p. e4965, 2020.

[87] X. Chen, C. Cao, and J. Mai, "Network anomaly detection based on deep support vector data description," in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. IEEE, 2020, pp. 251–255.

[88] L. Medsker and L. C. Jain, *Recurrent neural networks: design and applications*. CRC press, 1999.

[89] K. Smagulova and A. P. James, "A survey on lstm memristive neural network architectures and applications," *The European Physical Journal Special Topics*, vol. 228, no. 10, pp. 2313–2324, 2019.

[90] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru," *arXiv preprint arXiv:2305.17473*, 2023.

[91] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[92] B. Di Martino, A. Esposito, G. J. Pezzullo, and T.-H. Weng, "Evaluating machine and deep learning techniques in predicting blood sugar levels within the e-health domain," *Connection Science*, vol. 35, no. 1, p. 2279900, 2023.

[93] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[94] K. I. Roumeliotis and N. D. Tselikas, "Chatgpt and open-ai models: A preliminary review," *Future Internet*, vol. 15, no. 6, p. 192, 2023.

[95] I. Ahmed and R. Islam, "Gemini-the most powerful llm: Myth or truth," *Authorea Preprints*, 2024.

[96] M. Shukla, I. Goyal, B. Gupta, and J. Sharma, "A comparative study of chatgpt, gemini, and perplexity," *International Journal of Innovative Research in Computer Science & Technology*, vol. 12, no. 4, pp. 10–15, 2024.

[97] L. Colucci Cante, B. Di Martino, M. Graziano, D. Branco, and G. J. Pezzullo, "Automated storytelling technologies for cultural heritage," in *International Conference on Emerging Internet, Data & Web Technologies*. Springer, 2024, pp. 597–606.

[98] T. Zhang, Y. Zhang, V. Vineet, N. Joshi, and X. Wang, "Controllable text-to-image generation with gpt-4," *arXiv preprint arXiv:2305.18583*, 2023.

[99] S. Zhang, Z. Chen, Y. Shen, M. Ding, J. B. Tenenbaum, and C. Gan, "Planning with large language models for code generation," *arXiv preprint arXiv:2303.05510*, 2023.

[100] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[101] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving data privacy via federated learning: Challenges and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, 2020.

[102] X. Yao, C. Huang, and L. Sun, "Two-stream federated learning: Reduce the communication costs," in *2018 IEEE Visual Communications and Image Processing (VCIP)*. IEEE, 2018, pp. 1–4.

[103] N. Bouacida and P. Mohapatra, "Vulnerabilities in federated learning," *IEEE Access*, vol. 9, pp. 63 229–63 249, 2021.

[104] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated learning in smart city sensing: Challenges and opportunities," *Sensors*, vol. 20, no. 21, p. 6230, 2020.

[105] E. J. Horvitz, J. S. Breese, and M. Henrion, "Decision theory in expert systems and artificial intelligence," *International journal of approximate reasoning*, vol. 2, no. 3, pp. 247–302, 1988.

[106] Y. Shanliang, F. Yuewen, Z. Peng, and H. Kedi, "Implementation of a rule-based expert system for application of weapon system of systems," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*. IEEE, 2013, pp. 2603–2606.

[107] S. S. Abu-Naser, H. El-Hissi, M. Abu-Rass, and N. El-Khozondar, "An expert system for endocrine diagnosis and treatments using jess," 2010.

[108] M. Omid, "Design of an expert system for sorting pistachio nuts through decision tree and fuzzy logic classifier," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4339–4347, 2011.

[109] L. Van der Gaag and P. Lucas, "An overview of expert system principles," *Expert Syst Model Autom Reason*, pp. 195–224, 1990.

[110] L. Muñoz-Saavedra, L. Miró-Amarante, and M. Domínguez-Morales, "Augmented and virtual reality evolution and future tendency," *Applied sciences*, vol. 10, no. 1, p. 322, 2020.

[111] J. Donally, *The immersive classroom: Create customized learning experiences with AR/VR*. International Society for Technology in Education, 2022.

[112] M. D. O. Carrasco and P.-H. Chen, "Application of mixed reality for improving architectural design comprehension effectiveness," *Automation in Construction*, vol. 126, p. 103677, 2021.

[113] M. Nabiyouni, S. Scerbo, D. A. Bowman, and T. Höllerer, "Relative effects of real-world and virtual-world latency on an augmented reality training task: an ar simulation experiment," *Frontiers in ICT*, vol. 3, p. 34, 2017.

[114] J.-M. Normand, M. Servières, and G. Moreau, "A new typology of augmented reality applications," in *proceedings of the 3rd augmented human international conference*, 2012, pp. 1–8.

[115] B. Di Martino, G. J. Pezzullo, W. Low, P. Ljungberg, and S. Saha, "Survey on reference architecture for cloud continuum and multi-access edge computing (mec) in 5g networks," in *International Conference on Advanced Information Networking and Applications*. Springer, 2024, pp. 141–150.

[116] N. P. A. Plan, "National institute of standards and technology (nist)," *Retrieved from*.

[117] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Barger, and D. Leaf, "Nist cloud computing reference architecture. nist sp 500-292," *Consultado el*, vol. 6, no. 08, p. 2013, 2011.

[118] M. Ciotti, M. Ciccozzi, A. Terrinoni, W.-C. Jiang, C.-B. Wang, and S. Bernardini, "The covid-19 pandemic," *Critical reviews in clinical laboratory sciences*, vol. 57, no. 6, pp. 365–388, 2020.

[119] Y. Kang, Z. Cai, C.-W. Tan, Q. Huang, and H. Liu, "Natural language processing (nlp) in management research: A literature review," *Journal of Management Analytics*, vol. 7, no. 2, pp. 139–172, 2020.

[120] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[121] B. Di Martino, G. J. Pezzullo, D. Branco, V. Bombace, and S. Ceglie, "Distributed collaborative ar on cloud continuum: A case study for cultural heritage," in *International Conference on Advanced Information Networking and Applications*. Springer, 2024, pp. 132–140.

[122] B. Di Martino, G. J. Pezzullo, and E. Grassia, "Support for automated story telling using natural language processing techniques aimed at recognizing narrative elements," in *International Conference on Emerging Internet, Data & Web Technologies*. Springer, 2024, pp. 607–616.

[123] G.-N. Ke, D. Grajfoner, S. Carter, N. DeLima, R. Khairudin, W.-Y. Lau, K. A. Kamal, and S. C. Lee, "Psychological wellbeing and employability of retrenched workforce during covid-19: a qualitative study exploring the mitigations for post pandemic recovery phase," *Frontiers in Public Health*, vol. 10, p. 907797, 2022.

[124] X. Fu, J. Zhang, and L. Wang, "Introduction to the special section: the impact of covid-19 and post-pandemic recovery: China and the world economy," *Journal of Chinese Economic and Business Studies*, vol. 18, no. 4, pp. 311–319, 2020.

[125] B. Dutta and M. DeBellis, "Codo: an ontology for collection and analysis of covid-19 data," *arXiv preprint arXiv:2009.01210*, 2020.

[126] J. V. Stone, "Bayes' rule: a tutorial introduction to bayesian analysis," 2013.

[127] V. Vassiliadis, J. Wielemaker, and C. Mungall, "Processing owl2 ontologies using thea: An application of logic programming." in *OWLED*, vol. 529. Citeseer, 2009.

[128] D. L. Gomes and T. H. B. Barros, "The bias in ontologies: An analysis of the foaf ontology," in *Knowledge Organization at the Interface*. Ergon-Verlag, 2020, pp. 236–244.

[129] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall *et al.*, "The obo foundry: coordinated evolution of ontologies to support biomedical data integration," *Nature biotechnology*, vol. 25, no. 11, pp. 1251–1255, 2007.

[130] S. S. Ibrahim and N. A. S. Abdullah, "Ontology design for patient medication record," in *2021 13th International Conference on Information & Communication Technology and System (ICTS)*. IEEE, 2021, pp. 84–89.

[131] O. Mohammed, R. Benlamri, and S. Fong, "Building a diseases symptoms ontology for medical diagnosis: an integrative approach," in *The First International Conference on Future Generation Communication Technologies*. IEEE, 2012, pp. 104–108.

[132] B. Di Martino, A. Esposito, and G. J. Pezzullo, "Towards an intelligence system to support diseases' diagnoses and improve health treatments," in *2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*. IEEE, 2022, pp. 1–6.

[133] A. Dhawan, R. Rao, A. Ambekar, A. Pusp, and R. Ray, "Treatment of substance use disorders through the government health facilities: Developments in the "drug de-addiction programme" of ministry of health and family welfare, government of india," *Indian journal of psychiatry*, vol. 59, no. 3, pp. 380–384, 2017.

[134] M. DeBellis and B. Dutta, "From ontology to knowledge graph with agile methods: the case of covid-19 codo knowledge graph," *International Journal of Web Information Systems*, vol. 18, no. 5/6, pp. 432–452, 2022.

[135] G. R. Babu, R. Sundaresan, S. Athreya, J. Akhtar, P. K. Pandey, P. S. Maroor, M. R. Padma, R. Lalitha, M. Shariff, L. Krishnappa *et al.*, "The burden of active infection and anti-sars-cov-2 igg antibodies in the general population: Results from a statewide sentinel-based population survey in karnataka, india," *International Journal of Infectious Diseases*, vol. 108, pp. 27–36, 2021.

[136] R. Sivakumar and P. Arivoli, "Ontology visualization protégé tools–a review," *International Journal of Advanced Information Technology (IJAIT) Vol*, vol. 1, 2011.

[137] I. K. Alshammari, E. Atwell, and M. A. Alsalka, "Automatic mapping of quranic ontologies using rml and cellfie plugin," in *International Conference on Applications of Natural Language to Information Systems*. Springer, 2022, pp. 307–314.

[138] V. Belleudi, M. Finocchietti, F. Fortinguerra, A. Di Filippo, F. Trotta, M. Davoli, and A. Addis, "Drug prescriptions in the outpatient management of covid-19: evidence-based recommendations versus real practice," *Frontiers in Pharmacology*, vol. 13, p. 825479, 2022.

[139] C. J. Hemmer, M. Löbermann, and E. Reisinger, "Covid-19: epidemiology and mutations: An update," *Der Radiologe*, pp. 1–7, 2021.

[140] Y. Araf, F. Akter, Y.-d. Tang, R. Fatemi, M. S. A. Parvez, C. Zheng, and M. G. Hossain, "Omicron variant of sars-cov-2: genomics, transmissibility, and responses to current covid-19 vaccines," *Journal of medical virology*, vol. 94, no. 5, pp. 1825–1832, 2022.

[141] R. Sharnagat, "Named entity recognition: A literature survey," *Center For Indian Language Technology*, pp. 1–27, 2014.

[142] M. Shutaywi and N. N. Kachouie, "Silhouette analysis for performance evaluation in machine learning with applications to clustering," *Entropy*, vol. 23, no. 6, p. 759, 2021.

[143] P. Lemenkova, "Processing oceanographic data by python libraries numpy, scipy and pandas," *Aquatic Research*, vol. 2, no. 2, pp. 73–91, 2019.

[144] E. Bisong and E. Bisong, "Introduction to scikit-learn," *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*, pp. 215–229, 2019.

[145] X. Schmitt, S. Kubler, J. Robert, M. Papadakis, and Y. LeTraon, "A replicable comparison study of ner software: Stanfordnlp, nltk, opennlp, spacy, gate," in *2019 sixth international conference on social networks analysis, management and security (SNAMS)*. IEEE, 2019, pp. 338–343.

[146] S. Raschka, "An overview of general performance metrics of binary classifier systems," *arXiv preprint arXiv:1410.5330*, 2014.

[147] T. R. Harris, J. D. Bransford, and S. P. Brophy, "Roles for learning sciences and learning technologies in biomedical engineering education: A review of recent advances," *Annual Review of Biomedical Engineering*, vol. 4, no. 1, pp. 29–48, 2002.

[148] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," *Journal of Big data*, vol. 2, pp. 1–32, 2015.

[149] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2032–2033, 2012.

[150] B. Di Martino, V. Bombace, L. Colucci Cante, A. Esposito, M. Graziano, G. J. Pezzullo, A. Tofani, and G. D'Agostino, "Machine learning, big data analytics and natural language processing techniques with application to social media analysis for energy communities," in *Computational Intelligence in Security for Information Systems Conference*. Springer, 2022, pp. 425–434.

[151] G. D'Agostino, A. Tofani, V. Bombace, L. Colucci Cante, A. Esposito, M. Graziano, G. J. Pezzullo, and B. Di Martino, "Eclistener: A platform for monitoring energy communities," in *Computational Intelligence in Security for Information Systems Conference*. Springer, 2022, pp. 498–507.

[152] B. Di Martino, G. J. Pezzullo, and C. Beggiato, "Towards optimized design and deployment of a military supply chain on federated cloud continuum supported by simulation-based performance evaluation," in *2023 IEEE International Workshop on Technologies for Defense and Security (TechDefense)*. IEEE, 2023, pp. 423–428.

[153] K. Chowdhary and K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[154] P. Zerbino, A. Stefanini, and D. Aloini, "Process science in action: A literature review on process mining in business management," *Technological Forecasting and Social Change*, vol. 172, p. 121021, 2021.

[155] S. Selva Birunda and R. Kanniga Devi, "A review on word embedding techniques for text classification," *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020*, pp. 267–281, 2021.

[156] J. A. Zachman, "Conceptual, logical, physical: It is simple," *Zachman International, Co*, vol. 36, 2000.

[157] Y. Guo, Z. Zhao, K. He, S. Lai, J. Xia, and L. Fan, "Efficient and flexible management for industrial internet of things: A federated learning approach," *Computer Networks*, vol. 192, p. 108122, 2021.

[158] J. Stubbs, W. Moreira, and R. Dooley, "Distributed systems of microservices using docker and serfnode," in *2015 7th International Workshop on Science Gateways*. IEEE, 2015, pp. 34–39.

[159] R. Matthias, A. Bihlmaier, and H. Wörn, "Robustness, scalability and flexibility: key-features in modular self-reconfigurable mobile robotics," in *2012 IEEE International Con-*

ference on Multisensor Fusion and Integration for Intelligent Systems (MFI). IEEE, 2012, pp. 457–463.

[160] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, no. 2011. Athens, Greece, 2011, pp. 1–7.

[161] Y. Li and T. Yang, "Word embedding for understanding natural language: a survey," *Guide to big data applications*, pp. 83–104, 2018.

[162] Y. Vasiliev, *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press, 2020.

[163] S. Bird, "Nltk: the natural language toolkit," in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, 2006, pp. 69–72.

[164] L. Li, S. Liu, Y. Peng, and Z. Sun, "Overview of principal component analysis algorithm," *Optik*, vol. 127, no. 9, pp. 3935–3944, 2016.

[165] B. H. Meyer, A. T. R. Pozo, and W. M. N. Zola, "Global and local structure preserving gpu t-sne methods for large-scale applications," *Expert Systems with Applications*, vol. 201, p. 116918, 2022.

[166] Channabasamma, Y. Suresh, and A. Manusha Reddy, "A contextual model for information extraction in resume analytics using nlp's spacy," in *Inventive Computation and Information Technologies: Proceedings of ICICIT 2020*. Springer, 2021, pp. 395–404.

[167] E. Howland, "Regular expressions for checking dates." *Markup Languages: Theory & Practice*, vol. 2, no. 2, 2000.

[168] C.-z. Liu, Y.-x. Sheng, Z.-q. Wei, and Y.-Q. Yang, "Research of text classification based on improved tf-idf algorithm," in *2018 IEEE international conference of intelligent robotic and control engineering (IRCE)*. IEEE, 2018, pp. 218–222.

[169] S. Loria *et al.*, "textblob documentation," *Release 0.15*, vol. 2, no. 8, p. 269, 2018.

[170] M. V. Koroteev, "Bert: a review of applications in natural language processing and understanding," *arXiv preprint arXiv:2103.11943*, 2021.

[171] B. Sagot, "A multilingual collection of conll-u-compatible morphological lexicons," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[172] H. Stadtler, "Supply chain management: An overview," *Supply chain management and advanced planning: Concepts, models, software, and case studies*, pp. 3–28, 2014.

[173] J. Mills, J. Schmitz, and G. Frizelle, "A strategic review of "supply networks"," *International Journal of Operations & Production Management*, vol. 24, no. 10, pp. 1012–1036, 2004.

[174] B. Debnath, S. Sengupta, and J. Li, "Flashstore: High throughput persistent key-value store," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1414–1425, 2010.

[175] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez, W. Wei, H. Song, A. Zomaya, and S. Khan, "System modelling and performance evaluation of a three-tier cloud of things," *Future Generation Computer Systems*, vol. 70, pp. 104–125, 2017.

[176] B. Di Martino, G. Cretella, and A. Esposito, "Semantic and agnostic representation of cloud patterns for cloud interoperability and portability," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2. IEEE, 2013, pp. 182–187.

[177] H. Zankadi, A. Idrissi, N. Daoudi, and I. Hilal, "Identifying learners' topical interests from social media content to enrich their course preferences in moocs using topic modeling and nlp techniques," *Education and Information Technologies*, vol. 28, no. 5, pp. 5567–5584, 2023.

[178] J. D. Hamilton, *Time series analysis*. Princeton university press, 2020.

[179] M. R. Berthold and F. Höppner, "On clustering time series using euclidean distance and pearson correlation," *arXiv preprint arXiv:1601.02213*, 2016.

[180] P. Senin, "Dynamic time warping algorithm review," *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, p. 40, 2008.

[181] Y. Yu, Y. Zhu, D. Wan, H. Liu, and Q. Zhao, "A novel symbolic aggregate approximation for time series," in *Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM) 2019 13*. Springer, 2019, pp. 805–822.

[182] K. N. Behara, A. Bhaskar, and E. Chung, "A novel approach for the structural comparison of origin-destination matrices: Levenshtein distance," *Transportation Research Part C: Emerging Technologies*, vol. 111, pp. 513–530, 2020.

[183] D. T. Anh and L. H. Thanh, "An efficient implementation of k-means clustering for time series data with dtw distance," *International Journal of Business Intelligence and Data Mining*, vol. 10, no. 3, pp. 213–232, 2015.

[184] L. Cui, X. Su, Y. Zhou, and L. Zhang, "Clustergrad: Adaptive gradient compression by clustering in federated learning," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–7.

[185] S. K. Lo, Q. Lu, L. Zhu, H.-Y. Paik, X. Xu, and C. Wang, "Architectural patterns for the design of federated learning systems," *Journal of Systems and Software*, vol. 191, p. 111357, 2022.

[186] N. Bhuyan, S. Moharir, and G. Joshi, "Multi-model federated learning with provable guarantees," in *EAI International Conference on Performance Evaluation Methodologies and Tools*. Springer, 2022, pp. 207–222.

[187] L. Reinfurt, U. Breitenbücher, M. Falkenthal, F. Leymann, and A. Riegg, "Internet of things patterns for communication and management," *Transactions on Pattern Languages of Programming IV*, pp. 139–182, 2019.

[188] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, H. Yalame *et al.*, "Safelearn: Secure aggregation for private federated learning," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 56–62.

[189] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated reconstruction: Partially local federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 220–11 232, 2021.

[190] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE, 2020, pp. 1–6.

[191] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," *arXiv preprint arXiv:1812.00984*, 2018.

[192] M. BLESSING, "Topic: Ai-driven canary deployments for seamless zero downtime," 2021.

[193] B. Pintaudi, I. Gironi, R. Nicosia, E. Meneghini, O. Disoteo, E. Mion, and F. Bertuzzi, "Minimed medtronic 780g optimizes glucose control in patients with type 1 diabetes mellitus," *Nutrition, Metabolism and Cardiovascular Diseases*, vol. 32, no. 7, pp. 1719–1724, 2022.

[194] N. Zhang, Y. Zhang, and H. Lu, "Seasonal autoregressive integrated moving average and support vector machine models: prediction of short-term traffic flow on freeways," *Transportation Research Record*, vol. 2215, no. 1, pp. 85–92, 2011.

[195] C. Barcelo-Vidal, L. Aguilar, and J. A. Martín-Fernández, "Compositional varima time series," *Compositional Data Analysis: Theory and Applications. John Wiley & Sons*, 2011.

[196] M. Garrich, N. Amaya, G. S. Zervas, P. Giaccone, and D. Simeonidou, "Architecture on demand: Synthesis and scalability," in *2012 16th International Conference on Optical Network Design and Modelling (ONDM)*. IEEE, 2012, pp. 1–6.

[197] A. Ometov, O. L. Molua, M. Komarov, and J. Nurmi, "A survey of security in cloud, edge, and fog computing," *Sensors*, vol. 22, no. 3, p. 927, 2022.

[198] S. Venticinque and A. Amato, "A methodology for deployment of iot application in fog," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, pp. 1955–1976, 2019.

[199] S. Venticinque and S. Nacchia, "Learning and prediction of e-car charging requirements for flexible loads shifting," in *Internet and Distributed Computing Systems: 12th International Conference, IDCS 2019, Naples, Italy, October 10–12, 2019, Proceedings 12*. Springer, 2019, pp. 284–293.

[200] T. Weigert, D. Garlan, J. Knapman, B. Møller-Pedersen, and B. Selic, "Modeling of architectures with uml: Panel," in *International Conference on the Unified Modeling Language*. Springer, 2000, pp. 556–569.

[201] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[202] S. Hardikar, P. Ahirwar, and S. Rajan, "Containerization: cloud computing based inspiration technology for adoption through docker and kubernetes," in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 2021, pp. 1996–2003.

[203] I. Sazonov, D. Magiluk, and A. Mikhailova, "Analysis of modern tools for a mobile apps development running on android and ios," , p. 269, 2019.

[204] P. Chougale, V. Yadav, A. Gaikwad, and B. Vidyapeeth, "Firebase-overview and usage," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 3, no. 12, pp. 1178–1183, 2021.

[205] A. Kalaitzidou, N. Senechal, P. Dimitriou, K. Chandran, and M. Mcginity, ""e-wafe"-a full body embodied social exergame," in *Extended Abstracts of the 2022 Annual Symposium on Computer-Human Interaction in Play*, 2022, pp. 286–290.

[206] T. Grzejszczak, M. Kawulok, and A. Galuszka, "Hand landmarks detection and localization in color images," *Multimedia Tools and Applications*, vol. 75, pp. 16 363–16 387, 2016.

[207] A. Sharma, J. Pathak, M. Prakash, and J. Singh, "Object detection using opencv and python," in *2021 3rd international conference on advances in computing, communication control and networking (ICAC3N)*. IEEE, 2021, pp. 501–505.

[208] G. Chandan, A. Jain, H. Jain *et al.*, "Real time object detection and tracking using deep learning and opencv," in *2018 International Conference on inventive research in computing applications (ICIRCA)*. IEEE, 2018, pp. 1305–1308.

[209] J. Linowes, *Augmented reality with unity AR foundation: a practical guide to cross-platform AR development with Unity 2020 and later versions*. Packt Publishing Ltd, 2021.

[210] T. Chaudhry, A. Juneja, and S. Rastogi, "Ar foundation for augmented reality in unity," *International Journal of Advances in Engineering and Management (IJAEM)*, vol. 3, no. 1, pp. 662–667, 2021.

[211] R. R. Suryono, B. Purwandari, and I. Budi, "Peer to peer (p2p) lending problems and potential solutions: A systematic literature review," *Procedia Computer Science*, vol. 161, pp. 204–214, 2019.

[212] A. M. Potdar, D. Narayan, S. Kengond, and M. M. Mulla, "Performance evaluation of docker container and virtual machine," *Procedia Computer Science*, vol. 171, pp. 1419–1428, 2020.

[213] P. D. Ritsos, J. Mearman, J. Jackson, and J. C. Roberts, "Synthetic visualizations in web-based mixed reality," in *Immersive Analytics: Exploring Future Visualization and Interaction Technologies for Data Analytics: Workshop, IEEE Conference on Visualization (VIS), Phoenix, Arizona, USA*, 2017.

[214] T. Ahmed and Y. Singh, "Analytic study of load balancing techniques using tool cloud analyst," *International Journal of Engineering Research and Applications*, vol. 2, no. 2, pp. 1027–1030, 2012.

[215] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modeling," *Procedia Engineering*, vol. 38, pp. 3566–3572, 2012.

[216] R. Kumar and G. Sahoo, "Cloud computing simulation using cloudsim," *arXiv preprint arXiv:1403.3253*, 2014.

[217] A. Di Mauro, A. Di Nardo, G. F. Santonastaso, and S. Venticinque, "An iot system for monitoring and data collection of residential water end-use consumption," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–6.

[218] S.-R. Oh and Y.-G. Kim, "Development of iot security component for interoperability," in *2017 13th International Computer Engineering Conference (ICENCO)*. IEEE, 2017, pp. 41–44.

[219] E. Zhu, A. Hadadgar, I. Masiello, and N. Zary, "Augmented reality in healthcare education: an integrative review," *PeerJ*, vol. 2, p. e469, 2014.

[220] S. Gurusubramani, M. Sureshanand, J. Jeganamarnath, D. Sathishkumar, and A. Sheela, "Augmented reality in military applications," *Int. J. Eng. Adv. Technol*, vol. 9, no. 1S, pp. 51–54, 2019.

[221] L. M. Schriml, E. Mitraka, J. Munro, B. Tauber, M. Schor, L. Nickle, V. Felix, L. Jeng, C. Bearer, R. Lichenstein *et al.*, "Human disease ontology 2018 update: classification, content and workflow expansion," *Nucleic acids research*, vol. 47, no. D1, pp. D955–D962, 2019.

[222] D. L. McGuinness, F. Van Harmelen *et al.*, "Owl web ontology language overview," *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.

[223] A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent systems*, vol. 16, no. 2, pp. 72–79, 2001.

[224] B. D. Martino, A. Esposito, and G. Cretella, "Semantic representation of cloud patterns and services with automated reasoning to support cloud application portability," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 765–779, 2017.

[225] B. Di Martino, S. D'Angelo, A. Esposito, and P. Lupi, "Anomalous witnesses and registrations detection in the italian justice system based on big data and machine learning techniques," in *International Conference on Advanced Information Networking and Applications*. Springer, 2022, pp. 183–192.

[226] G. V. Gkoutos, C. Mungall, S. Dolken, M. Ashburner, S. Lewis, J. Hancock, P. Schofield, S. Kohler, and P. N. Robinson, "Entity/quality-based logical definitions for the human skeletal phenome using pato," in *2009 annual international conference of the IEEE engineering in medicine and biology society*. IEEE, 2009, pp. 7069–7072.

[227] H. Mi, A. Muruganujan, D. Ebert, X. Huang, and P. D. Thomas, "Panther version 14: more genomes, a new panther go-slim and improvements in enrichment analysis tools," *Nucleic acids research*, vol. 47, no. D1, pp. D419–D426, 2019.

[228] S. Köhler, L. Carmody, N. Vasilevsky, J. O. B. Jacobsen, D. Danis, J.-P. Gourdine, M. Gargano, N. L. Harris, N. Matentzoglu, J. A. McMurry *et al.*, "Expansion of the human phenotype ontology (hpo) knowledge base and resources," *Nucleic acids research*, vol. 47, no. D1, pp. D1018–D1027, 2019.

[229] Y. He, L. Cowell, A. D. Diehl, H. Mobley, B. Peters, A. Ruttenberg, R. H. Scheuermann, R. R. Brinkman, M. Courtot, C. Mungall *et al.*, "Vo: vaccine ontology," in *The 1st International Conference on Biomedical Ontology (ICBO-2009): July*, 2009, pp. 24–26.

[230] A. Y. Lin, Y. Yamagata, W. D. Duncan, L. C. Carmody, T. Kushida, H. Masuya, J. Beverley, B. Dutta, M. DeBellis, Z. M. Pendlington *et al.*, "A community effort for covid-19 ontology harmonization." in *ICBO*, 2021, pp. 122–127.

[231] M. Jensen, A. P. Cox, N. Chaudhry, M. Ng, D. Sule, W. Duncan, P. Ray, B. Weinstock-Guttman, B. Smith, A. Ruttenberg *et al.*, "The neurological disease ontology," *Journal of biomedical semantics*, vol. 4, no. 1, pp. 1–10, 2013.

[232] G. Davis, M. Grierson *et al.*, "Investigating attitudes of professional writers to gpt text generation ai based creative support tools," 2022.

[233] X. Yang and I. Tiddi, "Creative storytelling with language models and knowledge graphs." in *CIKM (Workshops)*, 2020.

[234] D. Rothman, *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more*. Packt Publishing Ltd, 2021.

[235] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[236] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[237] A. Amato, R. Aversa, D. Branco, S. Venticinque, G. Renda, and S. Mataluna, "Porting of semantically annotated and geo-located images to an interoperability framework," *Lecture Notes in Networks and Systems*, vol. 497 LNNS, p. 508 – 516, 2022.

[238] A. Ambrisi, R. Aversa, D. Branco, M. Ficco, S. Venticinque, G. Renda, and S. Mataluna, "Intelligent agents for diffused cyber-physical museums," *Studies in Computational Intelligence*, vol. 1026, p. 285 – 295, 2022, cited by: 1.

[239] L. Colucci Cante, B. Di Martino, and M. Graziano, "A comparative analysis of formal storytelling representation models," in *Complex, Intelligent and Software Intensive Systems*, L. Barolli, Ed. Cham: Springer Nature Switzerland, 2023, pp. 327–336.

[240] A. Nakasone and M. Ishizuka, "Storytelling ontology model using rst," in *2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2006, pp. 163–169.

[241] E. Concepción, P. Gervás, and G. Méndez, "A common model for representing stories in automatic storytelling," in *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*, Madrid, Spain, 12/2017 2017.

[242] V. B. Lenzi, F. Marcelloni, D. C. Meghini, F. Niccolucci, D. M. Doerr, and M. Luise, "An ontology for narratives," 2017.

[243] C. Meghini, V. Bartalesi, and D. Metilli, "Representing narratives in digital libraries: The narrative ontology," *Semantic Web*, vol. 12, no. 2, pp. 241–264, 2021.

[244] B. Di Martino, L. Colucci Cante, A. Esposito, and M. Graziano, "A tool for the semantic annotation, validation and optimization of business process models," *Software: Practice and Experience*, 2023.

[245] B. Di Martino, M. Graziano, L. Colucci Cante, A. Esposito, and M. Epifania, "Application of business process semantic annotation techniques to perform pattern recognition activities applied to the generalized civic access," in *Complex, Intelligent and Software Intensive Systems*, L. Barolli, Ed. Cham: Springer International Publishing, 2022, pp. 404–413.

[246] S. Hofer and H. Schwentner, *Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software*. Addison-Wesley Professional, 2021.

[247] M. K. J. Carlon, D. E. Gonda, E. M. Andrews, J. M. Gayed, R. A. Olexa, and J. S. Cross, "Educational nonlinear stories with twine," in *Proceedings of the Ninth ACM Conference on Learning@ Scale*, 2022, pp. 248–251.

[248] L. Nowak, I. Grabska-Gradzińska, W. Palacz, E. Grabska, and M. Guzik, "Tool for game plot line visualization for designers, testers and players," in *Cooperative Design, Visualization, and Engineering*, Y. Luo, Ed. Cham: Springer Nature Switzerland, 2023, pp. 85–93.

[249] R. Horst, M. Lanvers, and R. Dörner, "Quest-centric authoring of stories, quests, and dialogues for computer game modifications," in *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Volume 2: HUCAPP, Online Streaming, February 6-8, 2022*, A. Paljic, M. Ziat, and K. Bouatouch, Eds. SCITEPRESS, 2022, pp. 217–224. [Online]. Available: https://doi.org/10.5220/0010895800003124

[250] G. S. Nelson, "Bias in artificial intelligence," *North Carolina medical journal*, vol. 80, no. 4, pp. 220–222, 2019.

[251] S. Patil, V. Patil, and P. Bhat, "A review on 5g technology," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 1, no. 1, pp. 26–30, 2012.

[252] G. J. Pezzullo, B. Di Martino, and M. Bubak, "Container-based platform for computational medicine," in *International Conference on Advanced Information Networking and Applications*. Springer, 2022, pp. 131–140.

[253] M. A. Khan, E. Baccour, Z. Chkirbene, A. Erbad, R. Hamila, M. Hamdi, and M. Gabbouj, "A survey on mobile edge computing for video streaming: Opportunities and challenges," *IEEE Access*, vol. 10, pp. 120 514–120 550, 2022.

[254] X. He, H. Xing, Y. Chen, and A. Nallanathan, "Energy-efficient mobile-edge computation offloading for applications with shared data," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–6.

[255] W. Fan, S. Li, J. Liu, Y. Su, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for accuracy-aware machine-learning-based iiot applications," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3305–3321, 2022.

[256] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 881–892, 2021.

[257] T. Meng, "Security and performance tradeoff analysis of offloading policies in mobile cloud computing," Ph.D. dissertation, 2017.

[258] S. Ray, "A quick review of machine learning algorithms," in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 2019, pp. 35–39.

[259] E. F. Maleki, L. Mashayekhy, and S. M. Nabavinejad, "Mobility-aware computation offloading in edge computing using machine learning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 328–340, 2021.

[260] W. Junior, E. Oliveira, A. Santos, and K. Dias, "A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment," *Future Generation Computer Systems*, vol. 90, pp. 503–520, 2019.

[261] S. Yang, G. Lee, and L. Huang, "Deep learning-based dynamic computation task offloading for mobile edge computing networks," *Sensors*, vol. 22, no. 11, p. 4088, 2022.

[262] X. Li, Y. Qin, H. Zhou, Y. Cheng, Z. Zhang, and Z. Ai, "Intelligent rapid adaptive offloading algorithm for computational services in dynamic internet of things system," *Sensors*, vol. 19, no. 15, p. 3423, 2019.

[263] A. A. Alli and M. M. Alam, "Secoff-fciot: Machine learning based secure offloading in fog-cloud of things for smart city applications," *Internet of Things*, vol. 7, p. 100070, 2019.

[264] J. Chen and Z. Wu, "Dynamic computation offloading with energy harvesting devices: A graph-based deep reinforcement learning approach," *IEEE Communications Letters*, vol. 25, no. 9, pp. 2968–2972, 2021.

[265] H. Eom, R. Figueiredo, H. Cai, Y. Zhang, and G. Huang, "Malmos: Machine learning-based mobile offloading scheduler with online training," in *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*. IEEE, 2015, pp. 51–60.

[266] A. Shahidinejad and M. Ghobaei-Arani, "Joint computation offloading and resource provisioning for e dge-cloud computing environment: A machine learning-based approach," *Software: Practice and Experience*, vol. 50, no. 12, pp. 2212–2230, 2020.

[267] G. Qu, H. Wu, R. Li, and P. Jiao, "Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.