UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN
INGENIERÍA DE REDES Y SERVICIOS TELEMÁTICOS

TRABAJO FIN DE MASTER

DESIGN AND DEVELOPMENT OF A SEMANTIC ETHICAL BLACK BOX FOR A SOCIAL ROBOT

AUDIT OF THE BIAS AND FAIRNESS OF TRAINING DATASETS

SAULO JOSÉ NUEZ ORTEGA

2024

TRABAJO DE FIN DE MASTER

Título:	Diseño Y Desarrollo de una Caja Negra Ética Semántica para Robot Social
Subtítulo:	Auditoría del Sesgo y la Equidad de los Conjuntos de Datos de Entrenamiento
Título (inglés):	Design and Development of a Semantic Ethical Black Box for a Social Robot
Subtítulo (inglés):	Audit of the Bias and Fairness of Training Datasets
Autor:	Saulo José Nuez Ortega
Tutor:	Carlos Ángel Iglesias Fernández
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos
Grupo de Sistemas Inteligentes

TRABAJO DE FIN DE MASTER

DESIGN AND DEVELOPMENT OF A SEMANTIC ETHICAL BLACK BOX FOR A SOCIAL ROBOT

Septiembre 2024

Resumen

En la actualidad, estamos expuestos a una gran cantidad de información que abarca medios de comunicación, noticias del día a día, programas de entretenimiento e incluso la publicidad, hasta las tan habituales redes sociales que cuentan con una poderoso alcance de difusión de contenidos, lo que desencadena en un bombardeo de información que resulta abrumador y dificulta la distinción de información veraz frente a información que ha sido manipulada con el fin de influir en el pensamiento y en la forma de actuar de los usuarios. Esto sumado a los enlaces de anzuelo, también conocidos como "clickbaits", que hacen usos de titulares llamativos o imágenes fuera de la realidad, para aumentar el tráfico web, desviando la atención de contenidos valiosos culturalmente hablando y fomentando el consumo de información superficial y poco útil.

Para combatir este tipo de situaciones, surge el proyecto Análisis de sentimiento Moral en datos textuales (AMOR), de la mano del Grupo de Sistemas Inteligentes (GSI) perteneciente a la Universidad Politécnica de Madrid (UPM). Este proyecto tiene como objetivo desarrollar el pensamiento crítico y la capacidad de gestionar las propias emociones cuando se leen medios y redes sociales, con el fin de luchar contra fenómenos como la desinformación, el lenguaje del odio y los enlaces anzuelo. Esto ayudaría a evaluar la credibilidad de las fuentes de información y a desarrollar una opinión que no se base en lenguajes de odio o en noticias influyentes.

Este proyecto, se divide en varios objetivos generales, donde nuestro propósito, se centra en usar robots sociales que interactúen con humanos y almacenar todas estas interacciones en lo que se denomina como una caja negra ética, de manera que podamos representar la información mediante grafos de conocimiento y acceder a esta información y auditarla en caso de que se observe algún comportamiento inusual. Este sistema permite llevar un registro a la hora de tratar con usuarios de diferente personalidad, cultura o creencia, con el fin de detectar y analizar el comportamiento de dichos usuarios y los robots sociales puedan adaptarse al tipo de persona con la que esté tratando.

Palabras clave: Caja Negra Ética, Robot social, Grafo de conocimiento, Modelos de aprendizaje automático.

Abstract

Today, we are exposed to much information, ranging from the media, daily news, entertainment programs, and even advertisements to widespread social networks with a powerful content dissemination scope. This leads to a bombardment of information that can be overwhelming and makes it challenging to distinguish truthful information from other types of information that have been manipulated to influence the thinking and actions of consumer users. This, in addition to bait links, also known as "clickbait", which make use of flashy headlines or extravagant images that are out of touch with reality, to increase web traffic, diverting attention from more culturally valuable content and encouraging the consumption of superficial and unhelpful information.

To combat this type of situation, the Moral sentiment analysis in textual data (AMOR) project arises from the hand of the Intelligent Systems Group (GSI) belonging to the Polytechnic University of Madrid (UPM). This project aims to develop critical thinking and the ability to manage emotions when reading media and social networks and fighting against disinformation, hate speech, and bait links. This would help assess the credibility of the information sources consulted and develop an opinion not based on hate speech or influential news.

This project is divided into different general objectives. Our purpose focuses on using social robots that interact with humans and being able to store all these interactions in what is called an ethical black box so that we can represent the information using Knowledge Graphs (KGs) and make use of different tools to access this information and audit it in the event of any behavior that is out of the ordinary. As mentioned earlier, this system associated with the problem allows us to keep a record when dealing with users with different personalities, cultures, or beliefs to detect and analyze the behavior of these users, and the social robots can adapt to the type of person they are dealing with.

Keywords: Ethical Black Box, Social Robot, Knowledge Graph, Machine Learning Models.

Agradecimientos

Me gustaría expresar mi agradecimiento a Carlos Ángel por el tiempo que me ha dedicado y por ofrecerme la oportunidad de trabajar en el Grupo de Sistemas Inteligentes.

A mi familia, gracias por ser mi pilar incondicional, por estar siempre a mi lado en cada paso de este camino. Su apoyo constante y confianza han sido esenciales para llegar hasta aquí.

Por último, quiero dar las gracias a mis amigos, tanto a los que ya estaban como a aquellos que surgieron durante esta etapa, por su compañía y respaldo a lo largo del camino.

Contents

List of Figures

Acronyms

AI	Artificial Intelligence
AMOR	Moral sentiment analysis in textual data
API	Application Programming Interface
CSV	Comma-separated values
CVR	Cockpit Voice Recorder
ECTS	European Credit Transfer System
GSI	Intelligent Systems Group
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
KG	Knowledge Graph
KGTK	Knowledge Graph Toolkit
ML	Machine Learning
MFT	Moral Foundations Theory
ORO	Open Robots Ontology
PyKEEN	Python Knowledge Embeddings
RDF	Resource Description Framework
ROS	Robot Operating System
SQL	Structured Query Language
SSH	Secure Shell
TSV	Tab-separated values

TTL	Terse RDF Triple Language
UPM	Polytechnic University of Madrid
URI	Uniform Resource Identifier

CHAPTER 1

Introduction

This chapter will introduce the context of the project, including a brief overview of all the different parts that will be discussed in the project. It will also break down a series of objectives that will be carried out during the implementation of the project. Moreover, it will introduce the document's structure with an overview of each chapter.

1.1 Context

Today, we are immersed in a constant stream of information led by the media and many social networks with a powerful reach. This information overload can make it difficult to distinguish between the truth and manipulated information designed to influence our opinions and behaviour. In addition, the phenomenon of "clickbait" [43], with its flashy and misleading headlines, diverts our attention from culturally valuable content, encouraging the consumption of superficial information.

To face this challenge, the AMOR [4] project was proposed by the GSI of the UPM. The primary mission of this project is to promote critical thinking and emotional management in the reading of different media, such as those mentioned above, to combat disinformation, hate speech, and click-baiting strategies. It aims to enable people to assess the credibility of information sources and form informed opinions free from negative influences. To achieve this, it seeks to use social robots that interact with humans with different personalities, cultures, and beliefs to provide as truthful information as possible and customise it to the specific user they are dealing with.

With this in mind, it is crucial to consider that social bots use ML algorithms that may be trained on biased data, which could result in unusual or unethical behaviour. In addition, some of these algorithms are intended to detect emotions in users' faces, allowing the bot to interpret their feelings. Therefore, to detect and correct these behaviours, it is essential to store all this information in an accessible place. This would allow the necessary analyses and audits to be carried out to correct the models that have caused any problems.

1.2 Project goals

In response to the challenge posed by one of the general objectives of the AMOR project, which is to store and record all the information obtained from the interactions between robots and humans, the main objective of this master thesis is to develop an Ethical Black Box system capable of storing the interactions between robots and other users so that specific patterns can be detected and, when necessary, audits can be performed to fix such behaviors. This system will allow storing of all the relevant information of the interactions and, from it, create KGs that will facilitate deep analysis of both the behaviors induced by the robots and the reactions and behaviors observed in humans or other robots with which it interacts.

The project is structured around several specific objectives, which are detailed below. First, work will be done on semantic data processing, ensuring that the information collected is processed adequately for further analysis. This includes standardizing and structuring the data to ensure its consistency and usefulness in later phases. Next, KGs will be obtained, visually representing the information collected from these interactions and the robot's machine learning algorithms, allowing a more precise understanding and more effective management of the information collected.

Finally, an intuitive and efficient web page will allow users to analyze and visualize the system quickly. This interface will provide visual and navigation tools that simplify access to information and allow us to investigate and obtain relevant information, thanks to the information collected and the knowledge graphs obtained, to perform audits more effectively.

To achieve these objectives, various technologies and tools will be used to ensure the integration and cohesion of all system elements. This includes advanced technologies for data processing, the treatment of KGs, the use of Application Programming Interfaces (APIs), the management of machine learning models, and the implementation of web interfaces.

1.3 Structure of this document

The remainder of this master thesis is structured as follows.

Chapter 2 covers research conducted about other existing work that uses monitoring systems, which examines different moral foundations and where the importance of auditing in Artificial Intelligence (AI) algorithms is described.

Chapter 3 outlines the various technologies and tools employed in developing this project, emphasizing their key features and the benefits of using them in this study.

Chapter 4 describes the project's architecture, including the design phase and implementation details. A general picture of the prototype architecture and the project in which this master thesis is framed. Then, all the modules developed in the prototype are widely detailed.

Chapter 5 describes a selected use case within a specific scenario. This chapter explains how the whole system works when an interaction between a robot and a human occurs, from obtaining the logs to processing the information and displaying it on the web interface.

Chapter 6 discusses the conclusions drawn from this project and the problems faced in its development. Finally, we focus on the possible next step to be done for future work.

CHAPTER 2

State of Art

In this chapter, the concept of ethical black box in robotics will be introduced, examining its importance for transparency and safety in decision making. In addition, the moral foundations guiding human-robot interaction will be discussed, as well as the challenges related to biases in AI algorithms and the auditing practices needed to ensure their fairness and trustworthiness

2.1 Introduction

Ensuring these technologies operate ethically and fairly is crucial in a society increasingly driven by AI and automation. Automated systems and social robots are taking on increasingly significant roles in our daily lives, from autonomous driving to direct interaction with people in sensitive contexts, such as healthcare or education. However, this advancement also presents ethical and technical challenges that require a detailed and careful analysis to ensure that these technologies are efficient and morally acceptable.

This paper explores how implementing monitoring systems, such as the ethical black box in Sect. 2.2, can offer a solution to document and analyze the decisions made by automated systems and social robots. In addition, the moral foundations in human-robot interactions will be examined in Sect.2.3, as understanding how different cultures and people perceive machines' decisions is crucial to evaluating their ethical behavior. Finally, the importance of auditing biases in AI algorithms in Sect. 2.4, an essential process to ensure that these systems work fairly and equitably, will be addressed. Through this analysis, we seek to provide a comprehensive view of how to address the ethical and technical challenges that arise with the increasing autonomy of machines in our society.

2.2 Ethical black box

The concept of black boxes was introduced in 1958 in the aviation world to store data from large aircrafts, also known as flight data recorders [57]. Since then, these systems have been increasing the data they record, saving from the position on the surface where they are navigating, data from internal and external environment sensors, also data from the autopilot configuration as the selected destination, altitude, and speed, until finally including the Cockpit Voice Recorder (CVR) [57] which was an innovative integration that allowed to save the conversations that pilots had. In case of an accident, these recordings, along with other collected data, helped to interpret the circumstances of the state of the flight. They could discover the failures or situations that caused the incident and support continuous improvement in aviation.

The efficiency and usefulness of black-box systems have led to their adoption in other industries, including the automotive sector [44]. Similarly to their use in aviation, these systems in the automotive sector can record and analyze the behavior of drivers and the vehicle itself. These data can be used to study the causes of common car accidents and develop preventive measures, thereby improving safety and efficiency in-vehicle use.

The increasing integration of black-box systems is closely linked to the growing dependence on automated systems and robots, which are expected to make our work easier by delegating more tasks and responsibilities to them. This will lead to more autonomous behavior on the part of these systems, which means an increase in unpredictable events that can lead to dangerous situations. Therefore, analyzing and studying the factors that can cause these situations is vital to a correct and safe operation.

As automated systems and robots become more integrated into our lives and delegate tasks and responsibilities to them, the autonomous behavior of these systems increases, which could lead to unpredictable and potentially dangerous events. Therefore, it is essential to analyze and study the factors that can trigger these situations to ensure these automated systems' safe and correct operation.

With the exponential growth of AI technologies, the adoption of automated systems has spread beyond aviation and automotive to sectors such as social robotics. These social robots, designed to interact and communicate with humans and other robots [3], also use advanced technologies such as sensors, microphones, and cameras to capture and respond to environmental stimuli [56]. Thus, we need to record and analyze the behavior of these robots to ensure their proper functioning and assess the ethical implications of their decisions.

The use of social robots has expanded in a variety of fields, from medical assistance and rehabilitation therapies to the hospitality industry and education. These robots, which often help the elderly or people with health problems, are also used to study human behavior and social interaction, thanks to AI. However, unusual or unfair decisions affecting different social groups can arise, as can be the case with other automated systems.

To address these challenges, creating an "ethical black" in field robotics, similar to the black box used in aviation, has been proposed [58]. This ethical black box would store all information related to the actions and behaviors of social robots, allowing detailed analysis in case of incidents or controversial decisions. As in aviation, this analysis could reveal the underlying causes of errors or unexpected behaviors, providing valuable data to improve the safety and functionality of these systems.

Therefore, by storing and analyzing the information captured by social robots in these ethical black boxes, we can verify whether unfair decisions or incidents are related to the data on which the AI has been trained, allowing us to correct possible biases and continuously improve human-robot interaction in an ethical and safe framework.

2.3 Moral foundations

Interactions between humans and social robots, especially those equipped with AI, present critical ethical issues. The increasing ability of robots to make autonomous decisions and, in turn, establish complex relationships with humans requires continuous evaluation to ensure that these systems act ethically and fairly. This need underscores the importance of creating an ethical black box that records all interactions between robots and humans, allowing for subsequent review and analysis to detect and address unusual or unethical behavior.

However, ethical decisions or behaviors are not universal. Perceptions of what is just or unjust, right or wrong, can vary significantly among individuals, partly because of each person's different moral bases. These bases, such as care, fairness, loyalty, authority, purity, and freedom [54], are fundamental in determining whether a procedure or use of an AI and its outcome is ethical or not. As social robots interact in critical contexts, monitoring and evaluating these interactions is essential to ensure that they remain within the moral parameters accepted by society. All of this is taking into account that to develop these systems, the ethical issues of AI will have to be evaluated based on three dimensions [54]: (1) the organization's use of AI (i.e., the purposes for which it is used), (2) the data used to create and maintain the AI, and (3) the decisions the AI makes, as illustrated in Fig. 2.1 .

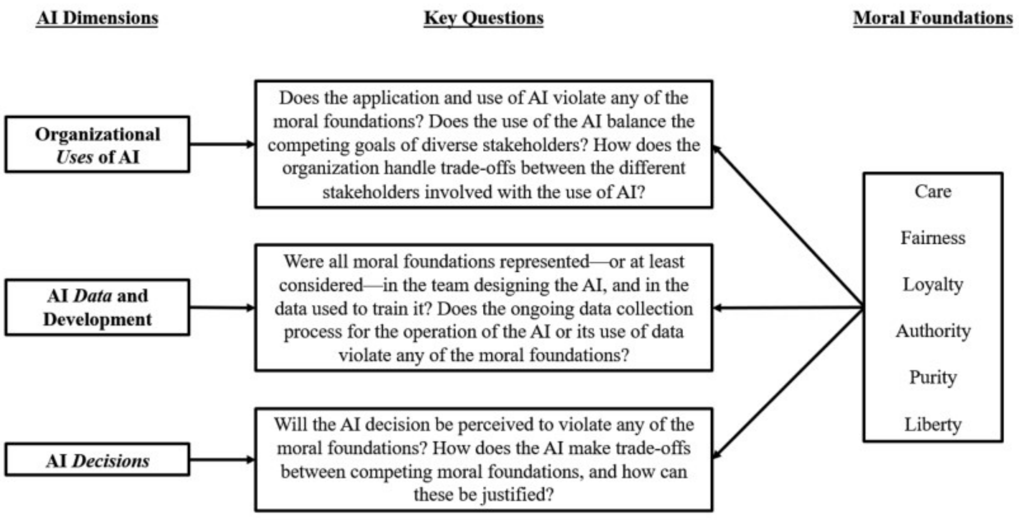

AI Dimensions
Key Questions
Moral Foundations
Organizational Uses of AI
Does the application and use of AI violate any of the moral foundations? Does the use of the AI balance the competing goals of diverse stakeholders? How does the organization handle trade-offs between the different stakeholders involved with the use of AI?
AI Data and Development
Were all moral foundations represented—or at least considered—in the team designing the AI, and in the data used to train it? Does the ongoing data collection process for the operation of the AI or its use of data violate any of the moral foundations?
AI Decisions
Will the AI decision be perceived to violate any of the moral foundations? How does the AI make trade-offs between competing moral foundations, and how can these be justified?
Care
Fairness
Loyalty
Authority
Purity
Liberty

Figure 2.1: Ethical AI dimensions and moral foundations [54]

An ethical black box would serve as a detailed record of all interactions, capturing data on the robot's decisions and the context in which they were made. This tool would be crucial for the transparency, detection of anomalies, and continuous improvement of AI systems. By analyzing the collected data, developers can identify unusual or potentially problematic behavior patterns and adjust algorithms accordingly.

The need for such monitoring becomes even more evident when we consider the complexity of human morality. Moral Foundations Theory (MFT), developed by Jonathan Haidt and Jesse Graham, helps to understand why ideas about right and wrong vary so much between cultures but remain crucial to human behaviour [5]. This theory says morality is not simple; it comprises various principles influenced by our evolutionary history, culture, and personal experiences.

These moral principles, such as care, fairness, loyalty, authority, and purity [19], influence how people view robot behavior and their decisions. For example, if a robot favors one person over another, that decision may affect the person's morality or culture, evaluating it differently. The ethical black box records these decisions and their principles, which helps us better understand why robots act in specific ways and whether their choices meet people's moral expectations.

Furthermore, if unethical behavior is detected according to these moral principles, the black box would allow for a thorough investigation. It would provide evidence on how and why a specific decision was made, either due to biases or errors in the training data or the development of the model. Access to the original model and the data used to train it helps us better understand the causes of the detected problems, allowing us to correct them more accurately and effectively.

2.4 Bias and auditing

When working with AI technologies and in the development and implementation of ML algorithms, bias represents a significant challenge in ensuring that these systems operate in a fair, accurate, and ethical manner [26]. In this context, bias refers to the fact that the data we provide to our models do not adequately reflect reality, resulting in the generation of unfair or inaccurate patterns [18]. These biases often originate in the data used during the training phase and in the methodologies and processes that guide the development of these ML algorithms. To address these problems, auditing algorithms and data quality management are essential steps in creating reliable, fair, and ethical systems.

Algorithm auditing has become crucial in identifying and mitigating biases within MLs. This audit process involves thoroughly examining how the algorithm works in specific environments to correct any biases or behaviors that can compromise the safety, reliability, or ethical standards of the system [26]. The goal is to improve the transparency of these algorithms [60], making it possible to understand and address the factors that influence their decision-making. We can follow an evaluation of the algorithm performance during the development stage based on five main phases [26]:

1. **Data and Task Setup.** This stage involves organizing the information we need and preparing it for use.

2. **Feature Pre-processing.** Here, the focus is on selecting the most relevant information and transforming it to be more understandable for our system.

3. **Model selection.** In this phase, different methods are tested to solve the given problem, followed by comparing and selecting the most effective one.

4. **Post-processing.** At this stage, results are presented in a way that makes them easier to understand and explain.

5. **Productionizing and Deploying.** The system is put into operation after undergoing several review processes.

A vital component of an audit is understanding the role that datasets play in the development of AI models. In this context, a ML dataset represents a data collection that can be processed as a single entity, aimed at analytical and predictive purposes by a computer [53]. The quality and representability of the data used to train these models will determine the performance and decision-making of these models [36], so having an unbalanced, incomplete, or biased data set can lead to models that reinforce existing inequalities, such as those related to gender, race, or social class [50]. Therefore, the initial phase of developing an AI model involves collecting the necessary data and rigorously processing it to ensure that it is free of errors and biases. This careful data management is essential for building models that can make fair and unbiased decisions.

Data bias can manifest itself in various ways [17], each with the potential to distort the results of an algorithm and lead to unfair or discriminatory decisions. For example, reporting bias can arise from documenting unusual situations, leading to unreliable results. Automation bias reflects the tendency to place excessive reliance on decisions made by automated systems, overlooking possible flaws that would be evident with human review. Implicit bias arises from assumptions based on personal experiences that may not be universally applicable. In addition, selection bias occurs when the sample used to train the model is not chosen appropriately or adequately, leading to biased and unrepresentative results.

Recognizing and addressing these biases is a critical step in the audit process. By thoroughly analyzing the data and the algorithm's behavior, auditors can identify where and how these biases affect the model's decisions. This understanding allows strategies to be developed to correct these problems [26], ensuring that the algorithm operates within ethical and legal standards. In addition, ongoing audits throughout the model's lifecycle allow for constant monitoring and improvement, making it possible to adapt the system as new biases or challenges arise.

Effective bias audit improves the fairness and reliability of AI systems and is crucial in building trust among users and stakeholders. In a world where AI is increasingly integrated into various aspects of society, from healthcare care to criminal justice, ensuring that these systems are free of bias is essential for their acceptance and success. By engaging in rigorous audit practices and addressing bias at every stage of the development process, we can create AI systems that are technically advanced and aligned with the ethical and moral standards of the communities they serve.

CHAPTER 3

Enabling Technologies

This chapter briefly reviews the main technologies that have made this project possible and some of the related published works.

3.1 Programming and Web Technologies

3.1.1 Python

In this project, we chose Python [12] as the programming language for server-side software development and the creation of web interfaces. Python, an interpreted and dynamically typed language, simplifies aspects such as defining variable data types and avoids compilation [25]. Its simplicity, along with its popularity in the field of data science, makes it an ideal choice for this work [7].

Python stands out for its ease of programming and readability compared to other languages, partly because it allows complex functions to be executed with just a few lines of code. In addition, it is open source and freely available, making it easy for the community to contribute to its continuous improvement. Its extensive standard library also offers various predefined modules and functions, which speed up development [46]. It has a comprehensive set of packages and frameworks created by third parties that cover many areas and industries. In data analysis, libraries such as Pandas and NumPy stand out, while for web development, this task is simplified with frameworks such as Django and Flask. Similarly, tools such as TensorFlow, PyTorch, and Scikit-Learn can be used in machine learning.

On the other hand, Python has some disadvantages, such as its execution speed, which is often slower than compiled languages such as C or C++ [7]. As an interpreted language, better choices for applications require high real-time performance. In addition, the intensive use of memory can be a drawback in systems with limited resources.

In conclusion, Python's constant evolution, versatility, readability, and many available libraries and frameworks make it one of the most popular programming languages. It is ideal for various applications, from web development to data science. Its ability to run on different platforms and the support of its community continue to attract developers and companies, making it an exciting technology to work with.

3.1.2 Streamlit

Streamlit [13] is an open-source application framework designed to create interactive web applications using Python quickly. This framework has gained popularity, especially in the fields of data science and machine learning, thanks to its ability to seamlessly integrate with popular Python libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn, among many others [9].

One of Streamlit's main advantages is its focus on simplicity and speed. Its intuitive syntax allows users with little web development experience to prototype applications without complex configurations quickly. This results in a more agile development process, allowing professional results to be obtained efficiently.

In addition, Streamlit excels in data visualization, facilitating the creation of interactive elements that allow data to be explored and presented dynamically. This approach is instrumental in data science projects, where the ability to manipulate and visualize data is crucial.

Another significant advantage of Streamlit is that it is open source and free, and its community is active and collaborative. This not only facilitates access to resources and documentation but also ensures that the tool is constantly evolving and improving, driven by the contributions of its users.

For all these reasons, Streamlit is considered an essential tool for developing data-centric web applications quickly and easily. Its ability to integrate with popular libraries, simplicity, and open-source nature make it ideal for data science and machine learning projects [52].

3.1.3 Flask

Flask [11] is a Python micro-framework designed to simplify the rapid creation of web applications by focusing on providing only basic functionality. This gives developers the flexibility to customize and expand their applications according to the specific needs of each project. As a lightweight framework, Flask can be used for both back-end and front-end development and offers essential tools such as an interactive debugger, a routing system to manage application routes, and Hypertext Transfer Protocol (HTTP) utilities to handle cookies and caches [37].

Unlike other more complex frameworks, Flask does not include a database abstraction layer or built-in validation or security features, allowing developers to decide which tools and libraries to implement. However, Flask has extensions that simplify adding functionality, such as server support, database management, migrations, asynchronous tasks, form validation, and request rate control [14]. In addition, it is fully compatible with Python 3 and newer versions.

In summary, Flask is an excellent technology that offers simplicity and flexibility, allowing developers to customize their web applications with only the necessary functionality. Being a lightweight microframework, it is ideal for projects that require a quick creation of applications without imposing a rigid structure, with a set of extensions that extend its functionality, adapting to various web development needs.

3.2 Machine Learning Technologies

3.2.1 MLflow

MLflow [33] is an open-source platform developed by Databricks that is designed to manage the lifecycle of ML projects [29]. This tool solves common challenges in developing ML solutions, such as recording, organizing, and tracking of experiments and models. MLflow enables development and data science teams to collaborate efficiently, facilitating experimentation, model tracking, and deployment. Its versatility makes it adaptable to various machine learning scenarios, optimizing the workflow from experimentation to production.

One of the most outstanding features of MLflow is MLflow Tracking [35], which allows you to record the parameters and metrics of ML experiments. This functionality maintains a detailed test history, facilitating model comparison and data-driven decision-making.

Another key feature is the MLflow Model Registry [34], which provides a robust system for registering and versioning trained models and their metadata and parameters. This facilitates model reuse and ensures that versions can be tracked over time. With model registration, teams can select the best-performing models and monitor their performance in real-world applications. This is critical to maintaining the quality and accuracy of deployed ML solutions.

MLflow stands out for its monitoring and logging functionalities and ability to integrate with famous ML libraries such as TensorFlow, PyTorch, and sci-kit-learn. This integration allows developers to work with popular tools in this sector while using MLflow to manage their experiments and models. In addition, the platform supports multiple programming languages, making it flexible and suitable for diverse development environments.

MLflow is, therefore, a feature-rich and flexible tool that addresses many of the challenges in the lifecycle of ML projects. Its functionalities, such as MLflow Tracking and MLflow Model Registry, improve the organization and tracking of experiments and models and foster collaboration in data science teams, favoring optimization in ML development processes.

3.2.2 Pandas

Pandas [38] is an open-source Python library that is an essential tool in data science, analysis, and manipulation. Initially developed in 2008 and opened to the public in 2009, it stands out for its ability to simplify and optimize work with data. It is crucial in an environment where accuracy and reliability are critical to project success.

Pandas main advantages are their ease of use and ability to handle data in various ways. This library enables complex operations such as data cleansing and transformation, including managing missing values and integrating different data sets [31]. These functionalities are essential to ensure that the data used in any analysis is accurate and reliable. In addition, Pandas integrates efficiently with other popular tools and libraries in the Python ecosystem, extending its capabilities and facilitating its use in data science projects.

Pandas is highly recommended regardless of the size of the data set. Its intuitive design and excellent capabilities make it a versatile and adaptable tool for various needs. Whether for small projects or for handling large volumes of information, it allows you to perform data analysis and transformations quickly and efficiently, making it an essential part of any data science toolkit.

3.2.3 Plotly

Plotly [42] is a Python data visualization library that allows you to create high-quality interactive plots. It is widely used in data science, financial analysis, and other areas where effective data visualization is critical. Plotly offers two main modules, Plotly Express and Plotly Graph Objects, which facilitate the creation of graphs of different complexity levels.

On the one hand, Plotly Express [41] is a high-level module designed to simplify graph creation. It is ideal for users looking to quickly generate visualizations from data sets without complicated configurations, generating scatter plots, histograms, and line graphs with a few lines of code, among other things.

On the other hand, Plotly Graph Objects [40] is the low-level module that offers more detailed and customized control over graphs. This module is ideal for creating complex or more customized visualizations, as it allows you to adjust every graph aspect.

In summary, Plotly is a versatile tool that adapts to quick and straightforward needs through its two main modules and projects requiring detailed and specific visualization control. This makes it an exciting tool for projects with a large amount of data and projects with a manageable amount of data.

3.3 Knowledge Graph Technologies

The development of AI based applications and the management of large KGs has been facilitated by advanced tools such as KGTK, Python Knowledge Embeddings (PyKEEN), and KG4Py. These technologies allow for the manipulation, preservation, and analysis of extensive KGs like Wikidata and DBpedia, simplifying their use in various environments.

KGTK consists of a Python library that allows different actions to be easily performed on KGs [24]. Its goal is to make it easier for developers to create and utilise KGs in applications [20], allowing them to work with these graphs in environments such as Jupyter notebooks, much like data scientists build data-driven models using Scikit-learn [39]. KGTK provides a specific file format for representing KGs, as well as tools for validating, cleaning, and manipulating data, modules for importing and exporting information in different formats, and techniques for performing queries and graph analysis [20].

On the other hand, PyKeen is a Python package specifically designed to train and evaluate KG embedding models [30]. It offers various models and functions to calculate model error and tools to measure its correct functioning, providing great flexibility in different situations [1]. Although PyKeen is easy to use and well-documented, its primary focus is training and evaluating KG embeddings.

Finally, KG4Py is another Python library that allows the generation of KGs and the performance of semantic searches using natural language [28]. It combines a pre-trained ML model with an unsupervised model to achieve more accurate and relevant semantic searches. This toolkit can also be used to find similar code fragments in enterprise codebases by understanding the semantics of function annotations, thus reducing work and costs in the development environment [28].

After comparing these technologies, we have decided to opt for KGTK mainly due to its extensive documentation rather than KG4Py. KGTK has a paper describing the idea proposed by this tool and on which we have already commented. It also has a web page with documentation to help users better understand its features and implementation. Finally, it presents a repository on GitHub with some tutorial notebooks to better understand its operation. It should be noted that PyKEEN also has fairly complete documentation and simple use. On the other hand, this tool is more focused on training and evaluation of KGEs, while KGTK is a more general tool. Hence, it fits more with what we are looking for when it comes to representing our information, making queries about it, and finally extracting it if necessary; as a negative part, we must consider that this technology is somewhat more complex.

Framework	**Main Functionality**	**Advantages**
KGTK	Manipulation and analysis of KGs	Extensive documentation, tutorials and versatile tools
PyKeen	Training and evaluation of KG embeddings	Easy to use, well-documented
KG4Py	Generation of KGs and semantic searches	Accuracy in semantic searches

Table 3.1: Comparison of knowledge graph technologies

CHAPTER 4

Architecture and Methodology

This chapter shows the system structure we have designed and each module's functioning, explaining the connection and interaction with the different elements that integrate our architecture.

4.1 Overview

The black box allows auditing of the interactions of robots with people and the AI algorithms designed for robots. A KG using the Open Robots Ontology (ORO) ontology to model this information. In this system, the robots use ML models and interact with each other and with people, as described in more detail in Sect. 4.2. To create this KG, the black box imports and transforms two data sources: a) information from two data sources, the logs of the robots' interactions and reasoning, and b) the configuration of machine learning models defined using an ML Tracking API.

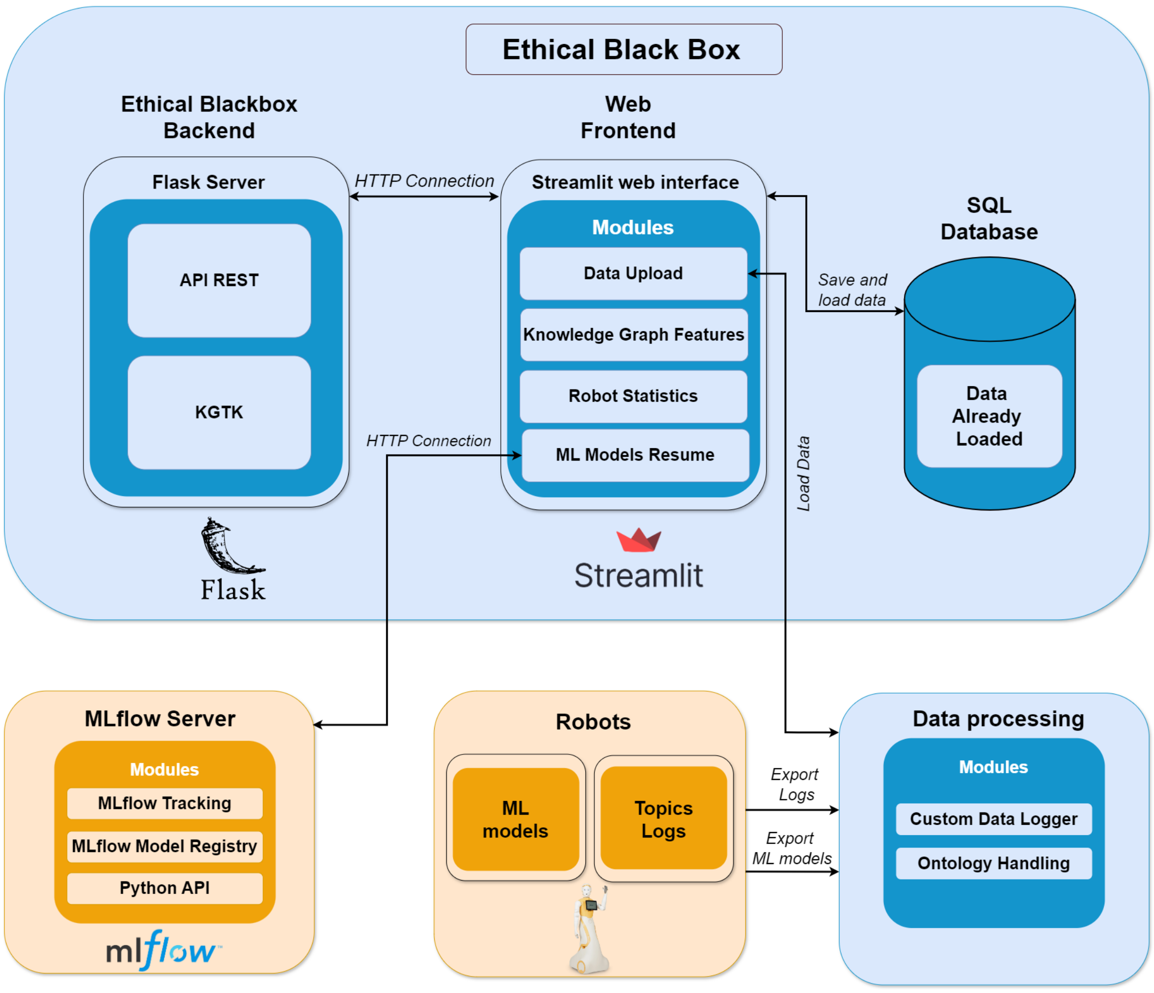


Figure 4.1: Design of our ethical black-box system architecture

The first data source depends on the type of robot used, which determines its capabilities and format. In our case, we have used ARI robots from PAL Robotics, and their reasoning (expressed in Resource Description Framework (RDF)) has been included in the logs, and the interactions carried out, as detailed in Sect. 2.2. Given the popularity of the second data source, we have used the MLflow API. However, other alternatives, such as MLSpec, aim to standardize it, and convergence in these solutions is expected. To perform semantic modeling, an ORO ontology has been used and adapted to model the main tracking elements. This information is valuable for correcting unethical behaviors that do not adapt to the specific context. To visualize all the information we have obtained from the robot's interactions with its environment, a web page with Streamlit has been developed that allows us to observe this information more dynamically and visually. At the same time, it is of great help to access the information we have already analyzed previously to compare results or observe such a case, which is why this system has an Structured Query Language (SQL) database that allows us to obtain these previously studied cases.

The architecture works as follows. First, the process starts with collecting the interactions between the robot and its environment. To achieve this, it is necessary to access the robot's operating system and develop specific code that allows the export of this information in a manageable and consistent format. This step is essential to ensure that the data obtained are easily processable and helpful for further analysis. Similarly, the ML models that make up the robot's AI are extracted. These models are stored on an MLflow server, which can be accessed to retrieve all associated features and metrics. This centralized storage allows for orderly management and facilitates access to the information needed for auditing and continuous improvement of the models. Once all relevant information has been collected, the next step is visualization and analysis. For this purpose, a web page has been developed where interaction records can be manually uploaded. This interface allows for a detailed observation of the behaviors recorded by the robot, which is essential for evaluating its performance. After loading these data, they are sent to a flask-developed back-end server, where the information is adapted to the established ontology. This ontology ensures that all data are integrated in a uniform format, which facilitates their handling and analysis. In one of the tabs of this website, the ML models previously stored in MLflow are accessible through an API, allowing these models to be retrieved and analyzed in an automated way consistent with the rest of the architecture. Finally, with the entire architecture in place, it is possible to perform an exhaustive analysis of the information collected. This analysis seeks to understand the robot's behavior patterns and detect possible anomalies or unusual behavior. If such irregularities are identified, a detailed audit could be performed to determine their cause and take corrective measures. This approach ensures that the system is functional, transparent, and capable of continuous improvement through data analysis and auditing feedback.

As illustrated in Fig. 4.1, this shows our core system, the ethical black box, which consists of several elements with their respective modules, all depicted in blue. This color differentiates it from other systems or valuable elements that we need to obtain or manage information and data that our primary system requires later. In the following, we will detail the elements that make up our architecture for easier understanding. We will describe these components from the bottom up, using Fig. 4.1 as a reference. We will start by explaining where the data that feed our core system, the Ethics Black Box, come from, related to the records of robot-human interaction. We will also discuss where social robots use the ML models are stored. We will then describe how this data is processed and managed to make it easily manageable and presentable in our system. Finally, with all this information, it will be easier to understand and explain the parts that make up our Ethical Black Box and how it works.

4.2 Robots

As mentioned above, the first step of our architecture consists of accessing the data that will feed our ethical black box system and which will be audited. This data comes from robots provided by the GSI of the UPM. Specifically, it is the ARI robot model manufactured by PAL Robotics, as mentioned above. The data extracted from these robots fall into two main categories: a) records of the robot's interactions, either with humans or with other robots, along with their reasoning processes, and b) the ML models that make up the robot's AI. This can be observed in the Fig. 4.2

The data source corresponding to the ML models has to do with algorithms that are either already included in the robot from the factory or have been developed later for a specific purpose. In our case, these robots arrived later than expected, so the factory models needed to be fully configured, and there was no time to develop any new ones. Therefore, to test our ethical black-box prototype, we have used generic models to simulate how they should appear when we access them and examine their properties. This is explained in more detail in Chap. 5.

On the other hand, the data source that has to do with the interactions of the robots with their environment has a more elaborate process of extraction on the one hand so that we can access our robot and export all the valuable data it is collecting, and on the other hand, the transformation of these data to convert them into a generic semantic model that allows us to maintain the cohesion of the information we are working with. This simplifies handling a large amount of data and the system's scalability. Logs of different topics are stored in the robots to achieve this goal. Among the logs we have used are those generated during conversations between humans and the robot, the responses generated by the robot, and the robot's facial expressions. These records are crucial for our ethical black box, providing a complete and detailed view of the robot's interactions with its environment. Analysis of these data allows us to identify patterns and behaviors, perform audits, and improve the models used. Furthermore, integrating these data into our KGs provides more efficient management and visualization.

The logs can be obtained in several ways, each with advantages and disadvantages. One of these ways is to access the WebCommander tool [48], a web page designed by PAL Robotics and hosted directly on the robot. This tool is accessible from a web browser on our computer. Provides information about the robot, including a section dedicated to viewing the generated logs, such as those mentioned above. The WebCommander interface allows users to intuitively visualize the logs, making it easy to access the relevant data. Another method of obtaining the logs is by accessing the robot's terminal. This is achieved through an Secure Shell (SSH) connection to the robot [2], allowing observation of the topic information in real-time from the command line. Despite the availability of these options, both present significant challenges in data structure, which can be difficult to handle and process. For this reason, we have chosen to develop a custom data logger [47]. This custom data logger allows us to collect and store the information on the above topics in a Comma-separated values (CSV) file format. This format is much more common and more accessible to manipulate, facilitating the analysis and integration of data into our system.

Since we are working with Robot Operating System (ROS), we must build and run a new ROS package to create our custom data logger. This package must be properly integrated into the ROS ecosystem to ensure proper functioning. The following detailed steps are necessary to establish an SSH connection with the robot. Once connected, we will proceed to:

1. Create a new workspace in ROS, where our package will be developed.
2. Configure the environment and dependencies necessary for the ROS package.
3. Write the scripts required to subscribe to the relevant topics and extract the desired data.
4. Format this data into CSV files and save them in an accessible location.
5. Test and validate the operation of the custom data logger to ensure that the data is collected and stored correctly.

This approach simplifies data management and ensures a structured and easily accessible record of the robot's interactions, essential for detailed analysis and efficient auditing of its behavior.

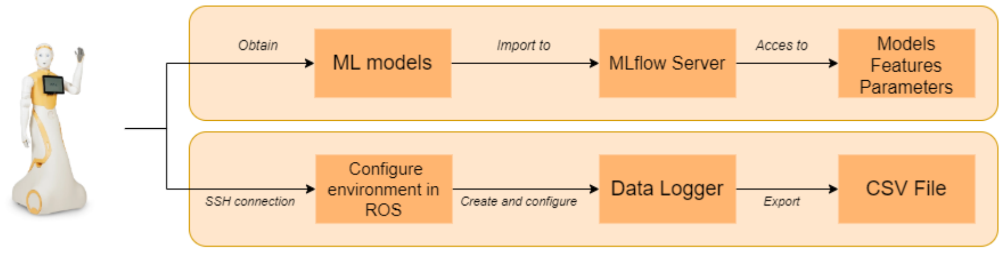


Figure 4.2: Robot information

4.3 MLflow Server

Another module that makes up our architecture consists of an MLflow server. This will allow us to store the models used by the robots and calculate the corresponding metrics, allowing us to have a central recorder to store and distribute the ML models used by our robot and to have them within reach if necessary. In our case, since we have not accessed the models used by the robots, we have trained some models that simulate the correct functioning of this system so that in the future, when we access these models, we can store them without significant problems. However, we will go into detail on this in the case studies.

When we access the MLflow server for the first time, we can see that the interface has a section on the left side where the experiments created for each stored model and their respective metrics appear. The first time the server is started, an experiment called "Default" appears in this sidebar, as shown in Fig. 4.3.

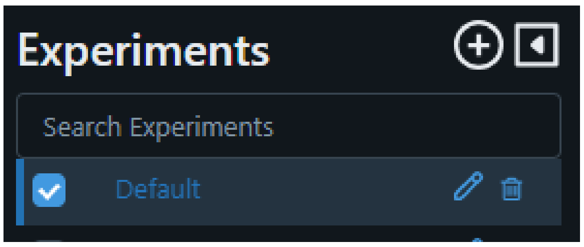


Figure 4.3: MlFlow Default Experiment

On the other hand, the central part of the interface displays the information related to the model to which the experiment refers and different options that allow us to configure the graphs and metrics we would like to visualize, as shown in Fig. 4.4.

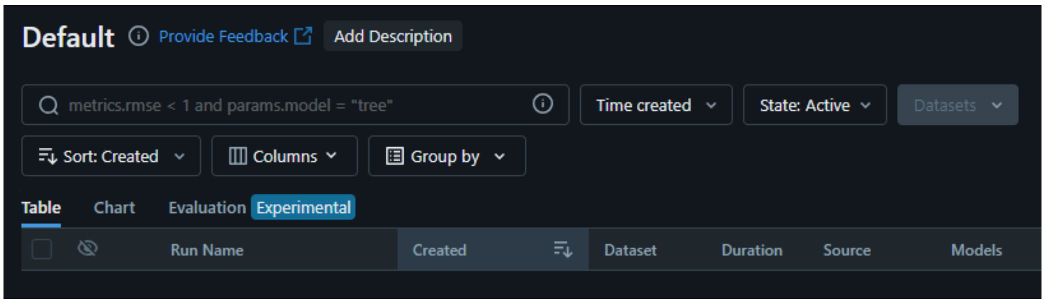


Figure 4.4: MlFlow Default Experiment Information

To create the experiments that record the models and the desired information, it is necessary to establish the Uniform Resource Identifier (URI) of the MLflow server where all the data in question will be stored and to define its name under which the executions will be recorded. In our case, the server is configured to run locally on http://127.0.0.1:5000.

```
remote_server_uri = "http://127.0.0.1:5000"
mlflow.set_tracking_uri(remote_server_uri)
mlflow.set_experiment("experiment_name")
```

The next step is to start a new run within the specified experiment, which logs the parameters, metrics, data sets, and other data from the experiment.

```
with mlflow.start_run(run_name="run_Example"):
mlflow.log_param("epochs", 10).
mlflow.log_metric("f1-score", f1)
mlflow.log_input(pd_dataset, context="training")
mlflow.keras.log_model(model, "model")
```

Once we verified that the server was working properly and that all information had been recorded correctly, we used the Python API provided by MLflow to access the stored information from our web front end.

4.4 Data Processing

In this section, we will discuss in detail one of the essential components of our architecture, the Data Processing box. This element plays a crucial role in how our system handles and transforms the data generated by the robot. To get a clearer idea, we will start by explaining how the logs of the interactions between robots and users are collected. These records, or logs, are the primary data source and capture the actions and responses over time during interactions. Then, once these logs are obtained, they go through a conversion process to an ontology, which consists of a structured and scalable format designed to be easily interpreted by our system. This approach ensures that the data are accessible, reusable, and seamlessly integrated into the operations of the Ethical Black Box.

4.4.1 Custom Data Logger

As mentioned, we have developed a custom data logger to obtain the robot logs of the topic we want in CSV format. This will make it much easier to deal with this type of data in later tasks related to adapting this information to the ontology we will use.

To do so, we have followed the documentation on the official PAL Robotics website and the official ROS website, which explains the steps to follow to achieve our goal. With this in mind, we explain these steps and what each does.

First, from our personal computer, we create a new Catkin workspace, which consists of a folder in which we can modify, build, and install Catkin packages:

```
mkdir -p ~/example_ws/src
```

This command creates a new workspace within our home directory containing a src folder. This workspace will allow us to organize and develop our code for ROS. The following command allows us to prepare our terminal to be able to use the commands and tools of the ROS Noetic version:

```
source /opt/ros/noetic/setup.bash
```

We then use the following commands to create a new catkin package within the previously defined workspace. We assign it a name and indicate that our package will use Python to develop nodes in ROS and the type of messages for communication.

Once the previous step is completed, we access the newly created package and create a new directory. We will add an empty script to fill the necessary configurations and give it execution permissions. For this, we execute the following commands:

```
mkdir scripts
```

```
touch scripts/hello_world.py
chmod +x scripts/hello_world.py
```

The next step is to edit the CMakeLists.txt file to automate the compilation process of the program we will run. The configuration of this file includes the following:

```
cmake_minimum_required(VERSION 2.8.3)
project(hello_world)
find_package(catkin REQUIRED COMPONENTS
 rospy
 )
 catkin_package()
 # Install Python scripts
install (PROGRAMS scripts/hello_world.py DESTINATION ${
    CATKIN_PACKAGE_BIN_DESTINATION})
```

In the same way, we implement the Python script we created in the previous steps. To do this, we have modified a code provided in the Pal Robotics documentation to create a custom data logger, configuring a ROS node that subscribes to the type of message our topic emits. We can obtain this information by using the command “rostopic type topicName”. With this in mind, the data of the received messages is extracted and saved in a CSV file.

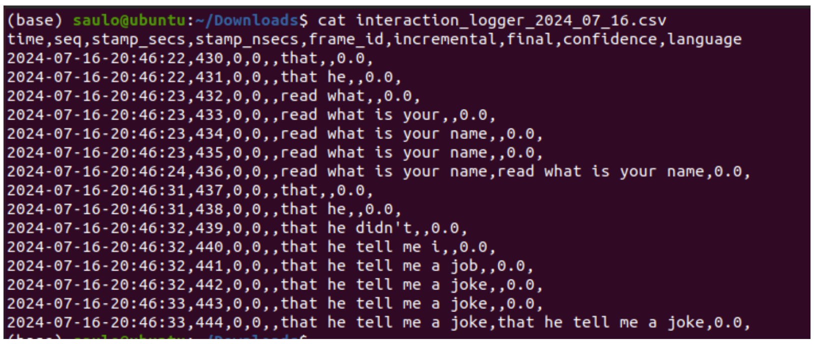
Next, we compile all the code inside our Catkin workspace created at the beginning and configured in the previous steps using the following command:

```
catkin_make
```

Finally, we run the following commands to finish configuring and preparing both the created workspace and ensuring that all packages are correctly installed and compiled. Finally, we run the ROS kernel and execute the script created:

```
cd build/
make install
cd
source example_ws/install/setup.bash
cd example_ws
source devel/setup.bash
roscore
rosrun hello_world hello_world.py
```

Finally, we would obtain the result shown in Fig. 4.5, where the different logs of the speech captured by the robot from the human are shown:



```
(base) saulo@ubuntu:~/Downloads$ cat interaction_logger_2024_07_16.csv
time,seq,stamp_secs,stamp_nsecs,frame_id,incremental,final,confidence,language
2024-07-16-20:46:22,430,0,0,,that,,0.0,
2024-07-16-20:46:22,431,0,0,,that he,,0.0,
2024-07-16-20:46:23,432,0,0,,read what,,0.0,
2024-07-16-20:46:23,433,0,0,,read what is your,,0.0,
2024-07-16-20:46:23,434,0,0,,read what is your name,,0.0,
2024-07-16-20:46:23,435,0,0,,read what is your name,,0.0,
2024-07-16-20:46:24,436,0,0,,read what is your name,read what is your name,0.0,
2024-07-16-20:46:31,437,0,0,,that,,0.0,
2024-07-16-20:46:31,438,0,0,,that he,,0.0,
2024-07-16-20:46:32,439,0,0,,that he didn't,,0.0,
2024-07-16-20:46:32,440,0,0,,that he tell me i,,0.0,
2024-07-16-20:46:32,441,0,0,,that he tell me a job,,0.0,
2024-07-16-20:46:32,442,0,0,,that he tell me a joke,,0.0,
2024-07-16-20:46:33,443,0,0,,that he tell me a joke,,0.0,
2024-07-16-20:46:33,444,0,0,,that he tell me a joke,that he tell me a joke,0.0,
```

Figure 4.5: Custom data logger logs

4.4.2 Ontology

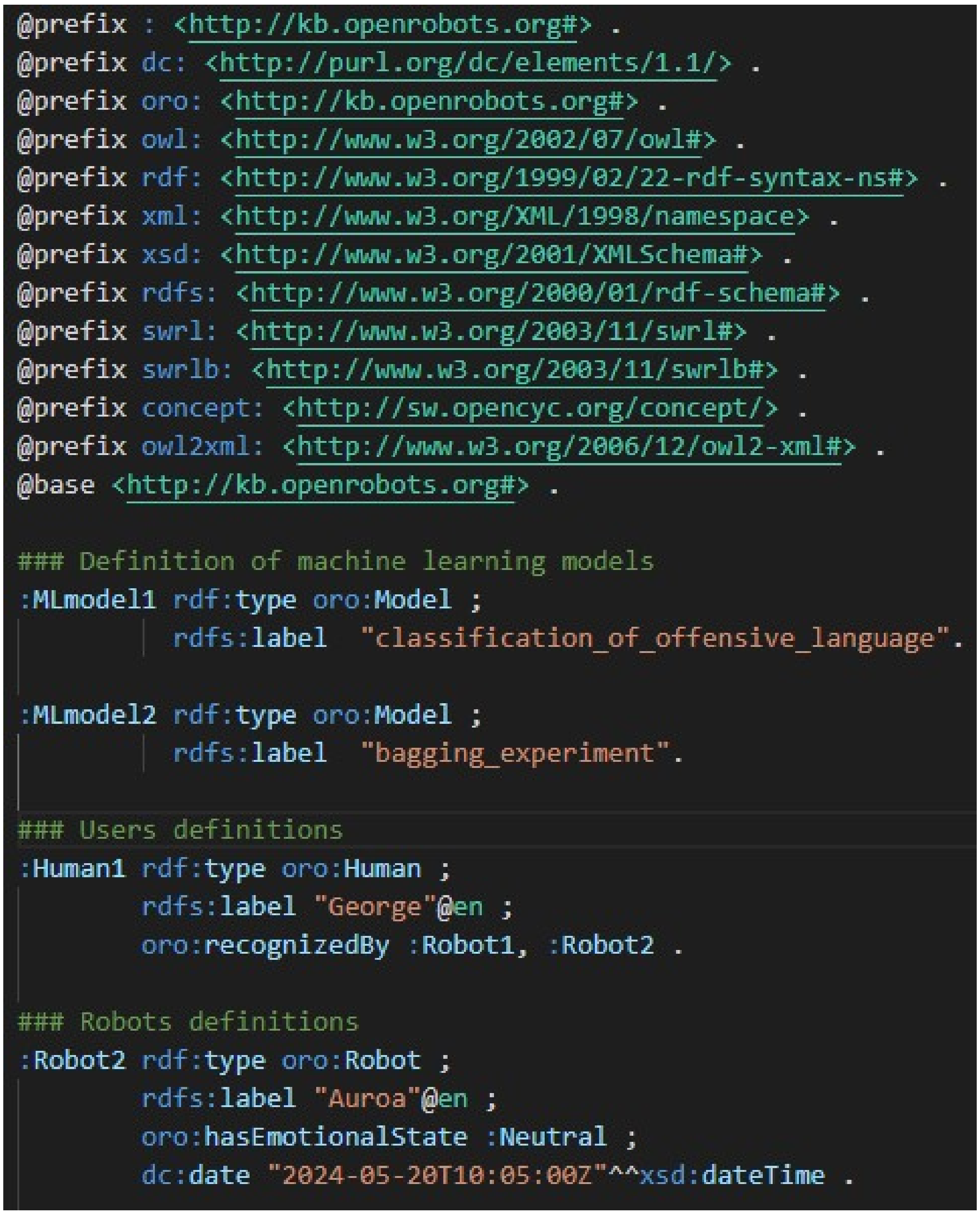
An ontology [16] is a formal and structured representation of a set of concepts and the relationships between them within a specific domain. It serves to organize and share knowledge clearly and coherently. In areas such as robotics and AI, ontologies enable machines to understand content in an explicit and actionable way, making it easier to provide more accurate and helpful answers [45]. Ontologies define entities, classes, properties, predicates, functions, and the relationships between these components [10]. Their principal value is that they standardize knowledge, allowing different systems to interact and share data and solving semantic interoperability problems. Machines must be able to reason in an advanced way and make decisions about their environment, making technologies such as AI and information systems work more efficiently and coherently.

ORO [27] is a knowledge management platform explicitly created for robots, designed to enable them to manage and process information effectively. It allows robots to understand the environment in which they find themselves, allowing them to associate concepts, actions and objects logically. This is made possible through an ontology that organizes knowledge into clear and coherent relationships between different elements of the environment, such as objects, the actions that can be performed with them, and the people with whom they interact. One of the key features of ORO is its ability to allow robots to make inferences, meaning they can deduce new information from fundamental knowledge. For example, if a robot knows that a cup is a type of liquid container and sees a cup on a table, it can intuit that it may contain water or some other liquid [27]. In addition, ORO focuses on physical objects and more abstract aspects corresponding to human behaviour. All these features make it easy for ORO to adapt to changing situations, learn about new objects and concepts, and make intelligent decisions in real time. This is crucial in complex environments where robots need to assist people autonomously. This makes the platform a powerful tool for advancing cognitive robotics and enabling robots to become more effective and valuable assistants in everyday life.

During the development of this thesis, we have worked in parallel on developing a specific ontology prototype for the AMOR project, which aims to define a formal semantic representation for modelling moral and ethical values. In this context, ORO will be used with this ontology, not to model robotic interactions but to incorporate a semantic structure to manage moral values in different scenarios. The integration of ORO with AMOR aims to unify and standardize the representation of these values, providing a coherent framework that facilitates their application in situations requiring ethical decisions. This ontology is currently under development and defines several relationships that allow the structuring of how a dataset can be composed of other characteristics and how each specific characteristic belongs to a given dataset. In addition, properties associated with this data, such as the name of the dataset, the name of the category or the name of a feature, are established, facilitating the description of these elements and providing a common language for naming and categorizing the entities involved. In terms of classes, an ontology defines several characteristics that are very important in ML, representing different types of data that can be analyzed or monitored within the system. These classes are essential to organize and manage how data can be structured and used in the models. In addition, the ontology allows for the modelling, auditing and monitoring of data quality, ensuring that, for example, the number of categories or the numerical range of values is consistent and correct. This is especially relevant in applications such as sentiment analysis, where it is crucial that categorical data, such as emotions, are accurately represented and audited.

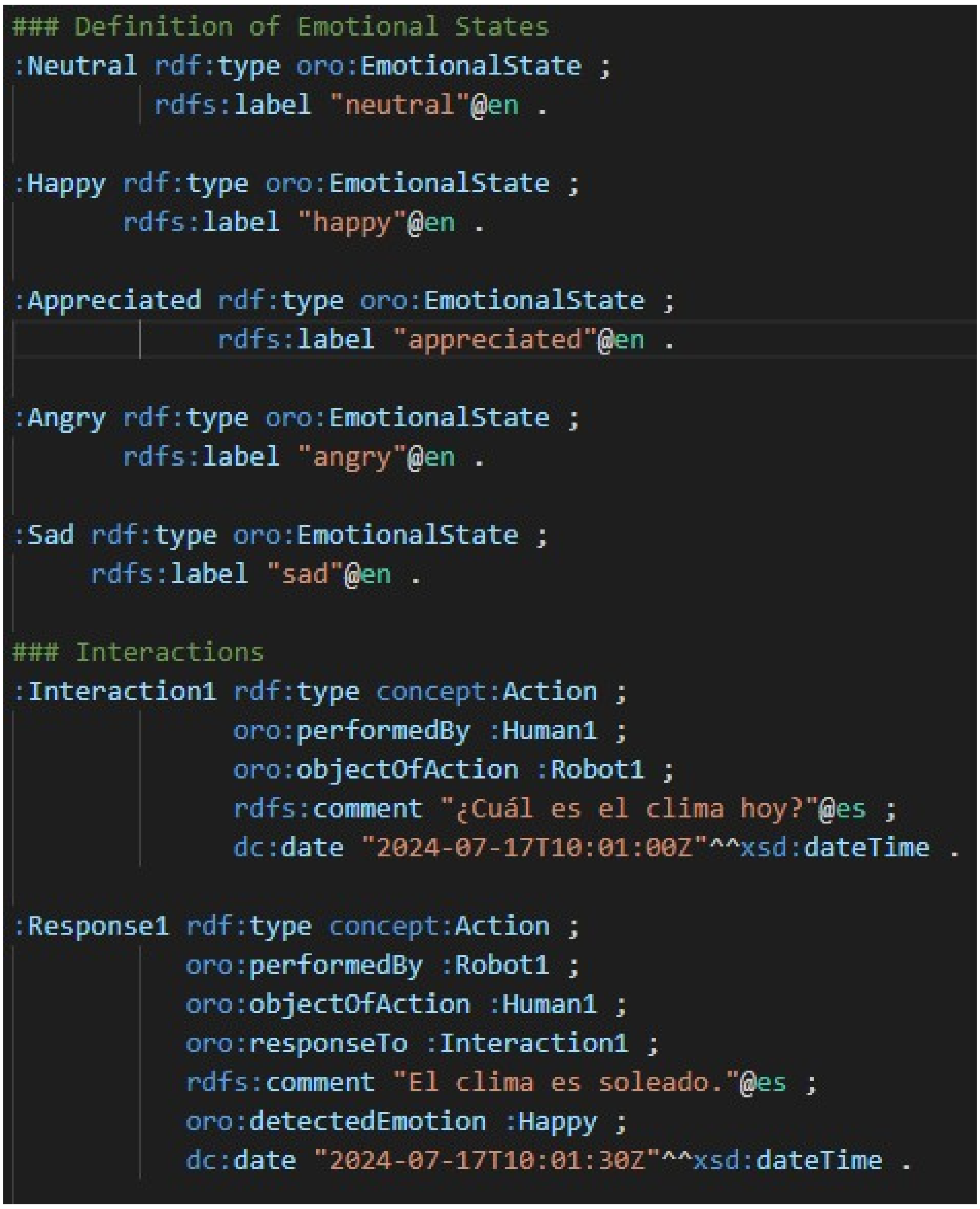
However, I would like to point out that this ontology is still under development. Although significant progress has been made in its development and design, the ontology needs to be more mature for current work. For this reason, we have chosen to use the AMOR ontology, a more general and contrasted ontology that has been slightly adapted to fit the needs of our project and our case study. The choice of the ORO ontology allows us to maintain a coherent semantic framework. At the same time, work continues on the specific ontology for AMOR, with the idea that once completed, it can replace the interim solutions we have used and offer a more suitable and customized model for our system.

Finally, once we have detailed how we have obtained the information from the robot records and the ML models associated with it and explained the processing of the data and the ontology used, we can see in Fig. 4.6 and in Fig. 4.7 the result showing how our resulting TTL file is structured after all these steps have been carried out. In this file, we can see the prefixes used associated with the ORO ontology and the definition of the different classes and instances for the different ML models, participant users, emotions and interactions.



```
@prefix : <http://kb.openrobots.org#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix oro: <http://kb.openrobots.org#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl: <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb: <http://www.w3.org/2003/11/swrlb#> .
@prefix concept: <http://sw.opencyc.org/concept/> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@base <http://kb.openrobots.org#> .

### Definition of machine learning models
:MLmodel1 rdf:type oro:Model ;
          rdfs:label  "classification_of_offensive_language".

:MLmodel2 rdf:type oro:Model ;
          rdfs:label  "bagging_experiment".

### Users definitions
:Human1 rdf:type oro:Human ;
        rdfs:label "George"@en ;
        oro:recognizedBy :Robot1, :Robot2 .

### Robots definitions
:Robot2 rdf:type oro:Robot ;
        rdfs:label "Auroa"@en ;
        oro:hasEmotionalState :Neutral ;
        dc:date "2024-05-20T10:05:00Z"^^xsd:dateTime .
```

Figure 4.6: TTL file example with prefix definition, ML models and users



```
### Definition of Emotional States
:Neutral rdf:type oro:EmotionalState ;
        rdfs:label "neutral"@en .

:Happy rdf:type oro:EmotionalState ;
       rdfs:label "happy"@en .

:Appreciated rdf:type oro:EmotionalState ;
             rdfs:label "appreciated"@en .

:Angry rdf:type oro:EmotionalState ;
       rdfs:label "angry"@en .

:Sad rdf:type oro:EmotionalState ;
     rdfs:label "sad"@en .

### Interactions
:Interaction1 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "¿Cuál es el clima hoy?"@es ;
              dc:date "2024-07-17T10:01:00Z"^^xsd:dateTime .

:Response1 rdf:type concept:Action ;
           oro:performedBy :Robot1 ;
           oro:objectOfAction :Human1 ;
           oro:responseTo :Interaction1 ;
           rdfs:comment "El clima es soleado."@es ;
           oro:detectedEmotion :Happy ;
           dc:date "2024-07-17T10:01:30Z"^^xsd:dateTime .
```

Figure 4.7: TTL file Example with interaction definition

4.5 Ethical Black-Box system

Once we have explained how to obtain the data that will feed our system, it is time to explain its operation and the components that make it possible. The section corresponding to our design of the Ethical Black Box comprises several elements, each with a specific function within the system, dividing our design into two main parts: the back end and the front end. The back-end part manages all the business logic and, in turn, is responsible for processing the data, managing the data, and ensuring its correct handling and storage. However, the front-end part is responsible for presenting the processed information in a visual and accessible way to the end user. The front end facilitates the user's interaction with the system through an intuitive interface, making the experience as clear and straightforward as possible. Both parts comprise several modules designed and configured to fulfill specific functions within our system. These modules work together, allowing the system to perform various functionalities efficiently and securely.

4.5.1 Ethical Black-Box Back-end

The backend of our system is based on a Flask server, a tool that facilitates the creation of lightweight and efficient web applications. This server hosts a REST API designed to handle the HTTP requests coming from the front-end web interface, which we will discuss later. The primary function of this API is to process and manage data using the KGTK framework, which is ideal for handling and creating KGs that will later be used in the front end of our application.

Thus, the process begins when we receive a file in TTL format, sent by the front end of our application through a HTTP request, whose file has been manually imported by the user. Because the TTL format is quite complex and KGTK does not fully support it, we use tools within KGTK to transform these data into a more manageable format, known as Tab-separated values (TSV). This transformation simplifies the creation and management of the KG, allowing interactions to be stored efficiently and accessible. In Fig. 4.8, an illustration that exemplifies this format conversion process can be visualized. Following similar logic, once we have the data in TSV format, we use other KGTK tools to convert it to a CSV format, ideal for working with Streamlit, which is the technology we use in our front-end to present the data. In addition, we generate an HyperText Markup Language (HTML) file that graphically represents the KG we use. Finally, all these files, both the CSV and the HTML, are returned to our web application through the REST API, ready to be used on the front end.

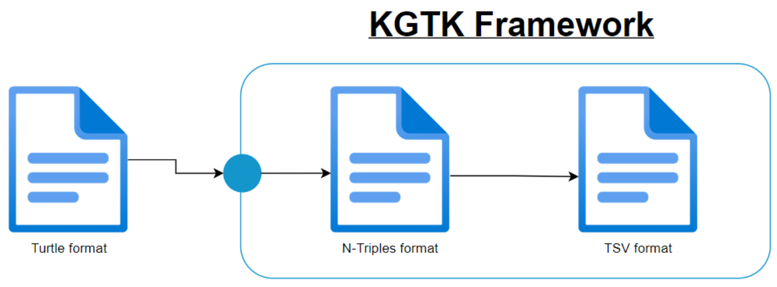


Figure 4.8: Format conversion model in KGTK

4.5.2 Web Frontend

Similarly, as mentioned above, the best way to visualize this information is through a website that we have developed using Streamlit [51]. This platform includes four modules or tabs, each designed to facilitate dynamic and efficient data interaction and analysis, as illustrated in Fig. 4.9. These tabs are intuitively located at the top of our website, and you can navigate between them by viewing the different analyses performed in each of them.

The first module allows us to load a TTL file obtained through the Data Processing element explained previously in Sect. 4.4. This Streamlit environment is ideal for working with this type of information, as it allows efficient and dynamic data manipulation. Once the data have been manually loaded, this module provides a view of the loaded data so that we can check that we have loaded the correct data and helps, in turn, to see the data structure we are working with; all this is explained in more detail in Chap. 5.

The second module of our web interface is designed to provide a dynamic visual representation of the KG we are working with. This graph is an interactive tool that lets us clearly and in detail visualize the relationships generated from the interactions between robots and users. Using this module, users can explore the graph by observing how the data is connected and discover essential patterns or trends that emerge from the interaction between robots and users. In addition, this section will highlight critical information that facilitates understanding and analysis of the relationships established within the system.

The third module of our web interface is designed to display various statistics related to the interactions between robots and humans that we have recorded. This module provides a detailed view of the collected data, allowing users to thoroughly explore and analyze the information. This module provides an overview of the data and allows for a deeper analysis of the trends and patterns that emerge from these interactions. By reviewing these statistics, users can identify repetitive behaviors, emotional changes, and other crucial patterns to understand the dynamics of robot-human interaction better. This analysis is essential to adjust and improve how robots respond to human emotions, thus optimizing future interaction quality.

Finally, the third module manages and visualizes the different ML models described in the aforementioned KG. These models have been stored in MLflow, which allows us to keep a detailed record of all the models used by the robots, along with their respective parameters. This module is beneficial for detecting and analyzing strange behavior or unusual patterns in robots. If unusual behavior is identified, the model in the KG that may have caused this behavior can be located and extracted, and a thorough audit can be performed to prevent recurrence. In addition, these models or experiments, as they are called in MLflow, can be manually saved and retrieved from the module on our website using a high-level API, which MLflow provides [32]. This facilitates the initiation and execution of processes in MLflow, providing complete control over the ML experiments and allowing for continuous adjustments and improvements.

The Web interface is also connected to an SQL database, adding functionality and flexibility to our application. This connection allows us to save and update our KG so that with this integration, the KGs we have created or modified are stored securely in the database. This facilitates loading them into the web application whenever needed and accessing them at any time to make adjustments, updates, or visualize them in the web interface. In addition, this storage capacity allows us to manage multiple KGs simultaneously, which is especially useful in complex projects where we need to work with different data sets. This functionality improves system efficiency and ensures that data management is robust and scalable, allowing us to adapt the application to future needs or incorporate new KGs.

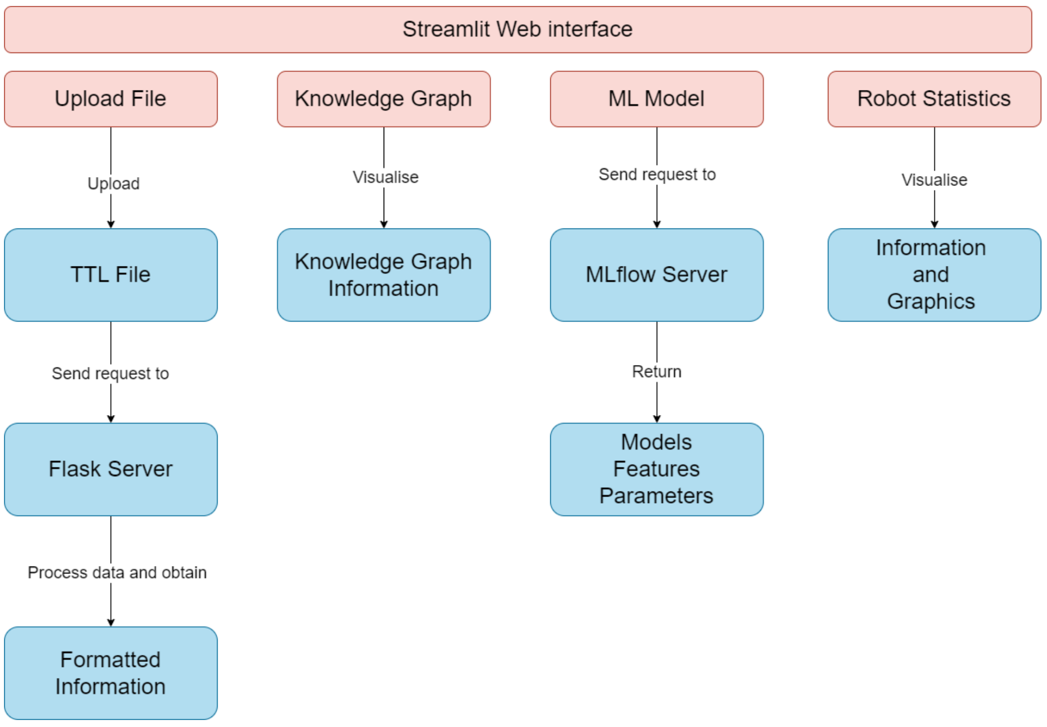


Figure 4.9: Web distribution

CHAPTER 5

Case study

In this chapter, we will describe a selected use case. This consists of an interaction between a social robot and a human, where we can observe the operation of our system from the moment the data is obtained and processed until its analysis and final visualization, thus understanding the operation of our architecture.

5.1 Scenario overview

In this project, we designed a system that allows users to interact with a robot trained in AI. During this interaction, the robot collects a series of data and generates real-time logs, which capture several key aspects of the communication between the user and the robot, including:

- What the robot understands about the user: Each time the user communicates with the robot orally, the robot processes the information and generates a detailed record of its understanding.
- The responses generated by the robot: In response to what the robot has understood, it issues a series of responses recorded in the logs. These logs allow analysis of how the robot responds in different situations and help to improve its interaction capabilities.
- The state of the robot's eyes: During the interaction, the state of the robot's eyes (e.g., open, closed, or showing signs of anger) is also recorded. This is important because the state of the eyes can reflect aspects of the interaction, such as the robot's attention or emotional response.

These data are recorded in a file in CSV format that stores all the information generated during the interaction. However, for this information to be helpful and used in the analyses we will carry out later, it must be transformed into a format that follows the ontology with which we have worked on this project. As mentioned above, this ontology is a structured model that allows us to organize the information coherently and understandably. Transforming the data from CSV format to ontology format is a crucial step, as this will enable us to have a richer and more structured data model that can serve us in different scenarios, facilitating the manipulation and analysis of information.

We will upload this file to our website once we have transformed our data into an ontology format. From the web page, a request is made to our Flask server, which we have implemented to handle the requests and process the data. Flask then receives the file with the data in ontology format and, using the KGTK that allows us to manipulate and transform the data efficiently, several conversions are performed on the data format, ensuring that the final format of the data is suitable for the creation of aKG.

The final objective of this whole process is to obtain a KG that facilitates the understanding and analysis of the relationships between the different elements of the data received from the interactions between the user and the robot, becoming a valuable tool on our website, as it allows users to explore and obtain intuitive and visual information, so that we can keep track of the behavior that has been followed and detected by our robot and in turn detect any unusual behavior by the robot or that the robot itself has obtained from the user.

In summary, all the previous steps will lead us to obtain a system called an ethical black box, which was the main objective of this thesis, allowing us to structure and analyze the information effectively, facilitating the extraction of valuable knowledge from the data collected during the interaction. The following is a detailed explanation of how this system works, covering its essential components and integration into the overall process. It describes the procedures necessary to ensure its functioning, from initial data capture to obtaining KG and its respective analysis.

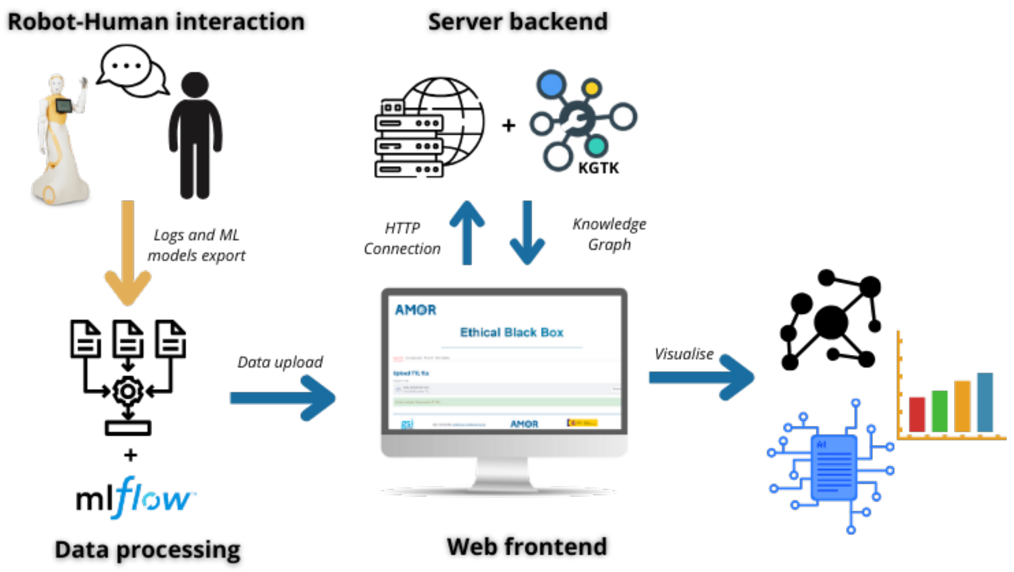


Figure 5.1: Case Use: Demo

5.2 Recording of interactions and data conversion

In this master's thesis, we have developed a system whose main objective is to capture and store detailed information about the interactions between a user and a robot. The purpose is to create a KG that allows deep analysis and control of the robot's behavior. This system is designed to detect patterns, identify anomalies, and predict behaviors from the robot-user interaction.

This process begins when a user communicates with the robot orally, using his voice to interact, and the robot responds by generating a spoken response after processing what it has received from the user. In Fig. 5.2, you can see an example of a robot interacting with a user. The system uses a custom data logger, explained in more detail in Sect. 4.4.1, to capture what the robot picks up from these interactions with the user and the robot's responses and behavior. As can be seen in Fig. 4.5, this data logger records what the user says until the final sentence is understood, for example, "read what your name is" or "that he tells me a joke", observing in these examples a specific error in capturing the message that may be due on the one hand to external agents such as ambient noise or voice projection in speech, but which is finally understood. These records are fundamental because they reflect the robot's understanding of the user's input for its analysis and to observe how the robot interprets different sentences.

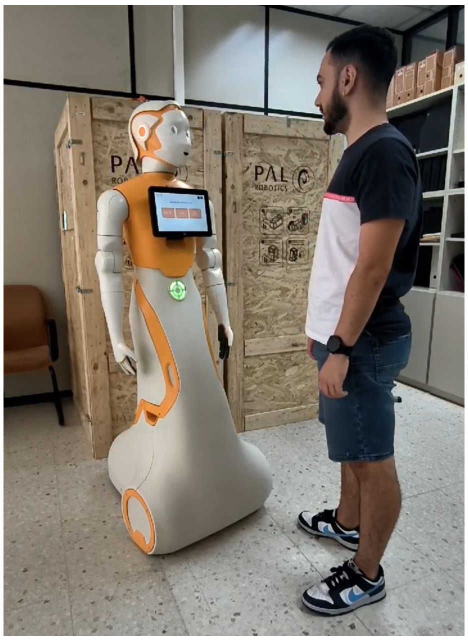
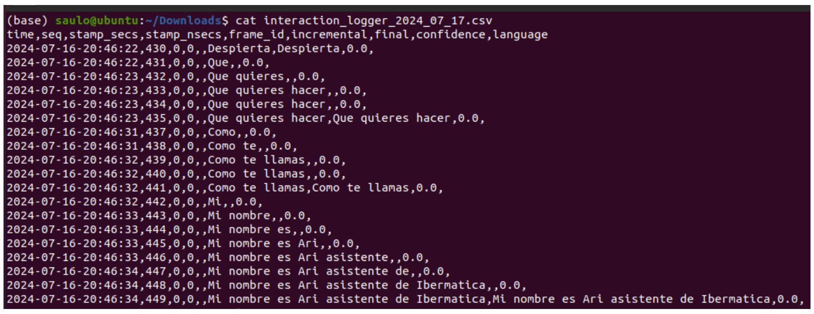


Figure 5.2: Robot-User interaction

The information stored is limited to what the robot understands. It considers the responses the robot generates based on that understanding, allowing for a detailed analysis of its behavior in different contexts. In addition, the state of the robot's eyes, whether open, closed, or showing some emotion, is also recorded. This is an exciting aspect, as the robot's eyes can reflect its attention or emotional response during the interaction, providing an enrichment when carrying out an analysis of the behavior performed by the robot or captured by the robot based on the user.

All this information is stored in a CSV file generated from the results obtained from logs such as those in Fig. 5.3, in which it can be seen how specific events in the dialogue are recorded, including the time, sequence, and exact content of what was said. This detailed recording is essential for the subsequent creation of the KG. The data logger, therefore, plays a crucial role in obtaining these logs, ensuring that all aspects of the interaction, from the user's understanding to the robot's responses and eye state, are accurately recorded. This data is then used to build a more structured knowledge model.



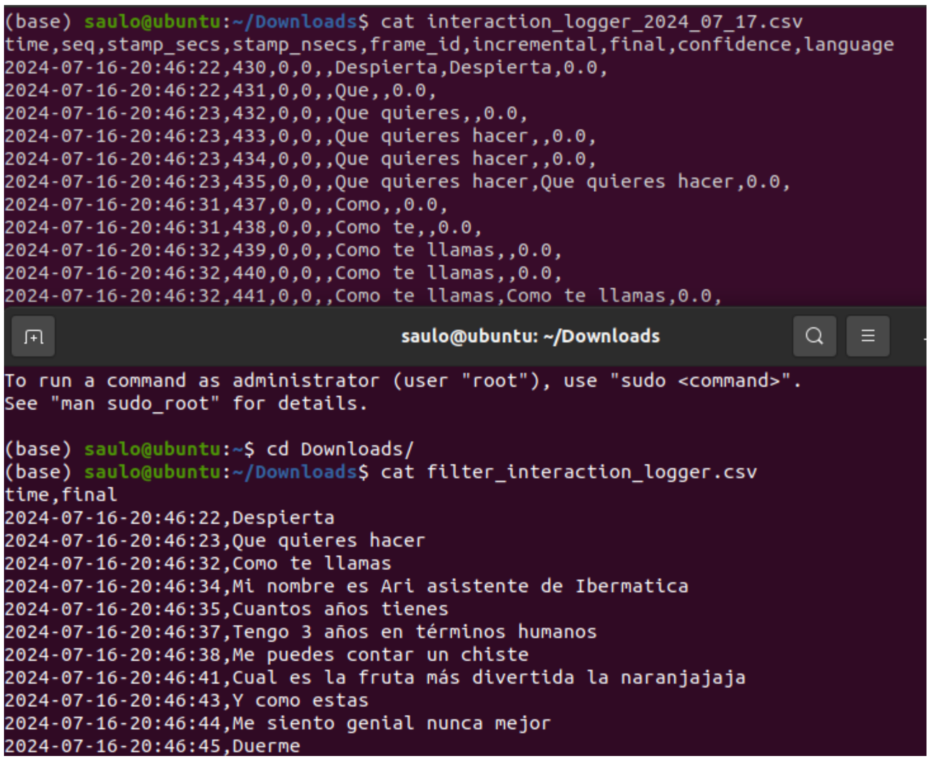
```
(base) saulo@ubuntu:~/Downloads$ cat interaction_logger_2024_07_17.csv
time,seq,stamp_secs,stamp_nsecs,frame_id,incremental,final,confidence,language
2024-07-16-20:46:22,430,0,0,,Despierta,Despierta,0.0,
2024-07-16-20:46:22,431,0,0,,Que,,0.0,
2024-07-16-20:46:23,432,0,0,,Que quieres,,0.0,
2024-07-16-20:46:23,433,0,0,,Que quieres hacer,,0.0,
2024-07-16-20:46:23,434,0,0,,Que quieres hacer,,0.0,
2024-07-16-20:46:23,435,0,0,,Que quieres hacer,Que quieres hacer,0.0,
2024-07-16-20:46:31,437,0,0,,Como,,0.0,
2024-07-16-20:46:31,438,0,0,,Como te,,0.0,
2024-07-16-20:46:32,439,0,0,,Como te llamas,,0.0,
2024-07-16-20:46:32,440,0,0,,Como te llamas,,0.0,
2024-07-16-20:46:32,441,0,0,,Como te llamas,Como te llamas,0.0,
2024-07-16-20:46:32,442,0,0,,Mi,,0.0,
2024-07-16-20:46:33,443,0,0,,Mi nombre,,0.0,
2024-07-16-20:46:33,444,0,0,,Mi nombre es,,0.0,
2024-07-16-20:46:33,445,0,0,,Mi nombre es Ari,,0.0,
2024-07-16-20:46:33,446,0,0,,Mi nombre es Ari asistente,,0.0,
2024-07-16-20:46:34,447,0,0,,Mi nombre es Ari asistente de,,0.0,
2024-07-16-20:46:34,448,0,0,,Mi nombre es Ari asistente de Ibermatica,,0.0,
2024-07-16-20:46:34,449,0,0,,Mi nombre es Ari asistente de Ibermatica,Mi nombre es Ari asistente de Ibermatica,0.0,
```

Figure 5.3: Custom data logger logs for Demo

Once the logs capturing the robot's interactions with the user and its environment have been obtained, extracting and structuring the most relevant information for subsequent analysis is necessary. The data collected in the data logger includes a series of records that show the evolution of the sentences captured by the robot. However, obtaining only the complete final sentences associated with each timestamp is most beneficial to achieve our goal since they represent the robot's final interpretation of what the user said.

To achieve this, we have used a script that first filters the initial log, extracting only the records where the column containing the final sentence is not empty, eliminating redundancies and focusing on the essential information, as we can see in Fig. 5.4. Subsequently, these filtered data are transformed into a standard and scalable format using the ontology explained in the previous section, which follows the rules of a semantic model. This facilitates the scalability of our system for further testing and the creation of a KG.

In conclusion, this log-conversion and structuring process is a crucial step toward constructing a system that uses KGs to store and analyze interactions that can be extended and improved. Using a KG will allow complex queries to be performed, hidden relationships between interactions to be discovered, and the system's efficiency to improve continuously.



```
(base) saulo@ubuntu:~/Downloads$ cat interaction_logger_2024_07_17.csv
time,seq,stamp_secs,stamp_nsecs,frame_id,incremental,final,confidence,language
2024-07-16-20:46:22,430,0,0,,Despierta,Despierta,0.0,
2024-07-16-20:46:22,431,0,0,,Que,,0.0,
2024-07-16-20:46:23,432,0,0,,Que quieres,,0.0,
2024-07-16-20:46:23,433,0,0,,Que quieres hacer,,0.0,
2024-07-16-20:46:23,434,0,0,,Que quieres hacer,,0.0,
2024-07-16-20:46:23,435,0,0,,Que quieres hacer,Que quieres hacer,0.0,
2024-07-16-20:46:31,437,0,0,,Como,,0.0,
2024-07-16-20:46:31,438,0,0,,Como te,,0.0,
2024-07-16-20:46:32,439,0,0,,Como te llamas,,0.0,
2024-07-16-20:46:32,440,0,0,,Como te llamas,,0.0,
2024-07-16-20:46:32,441,0,0,,Como te llamas,Como te llamas,0.0,
```

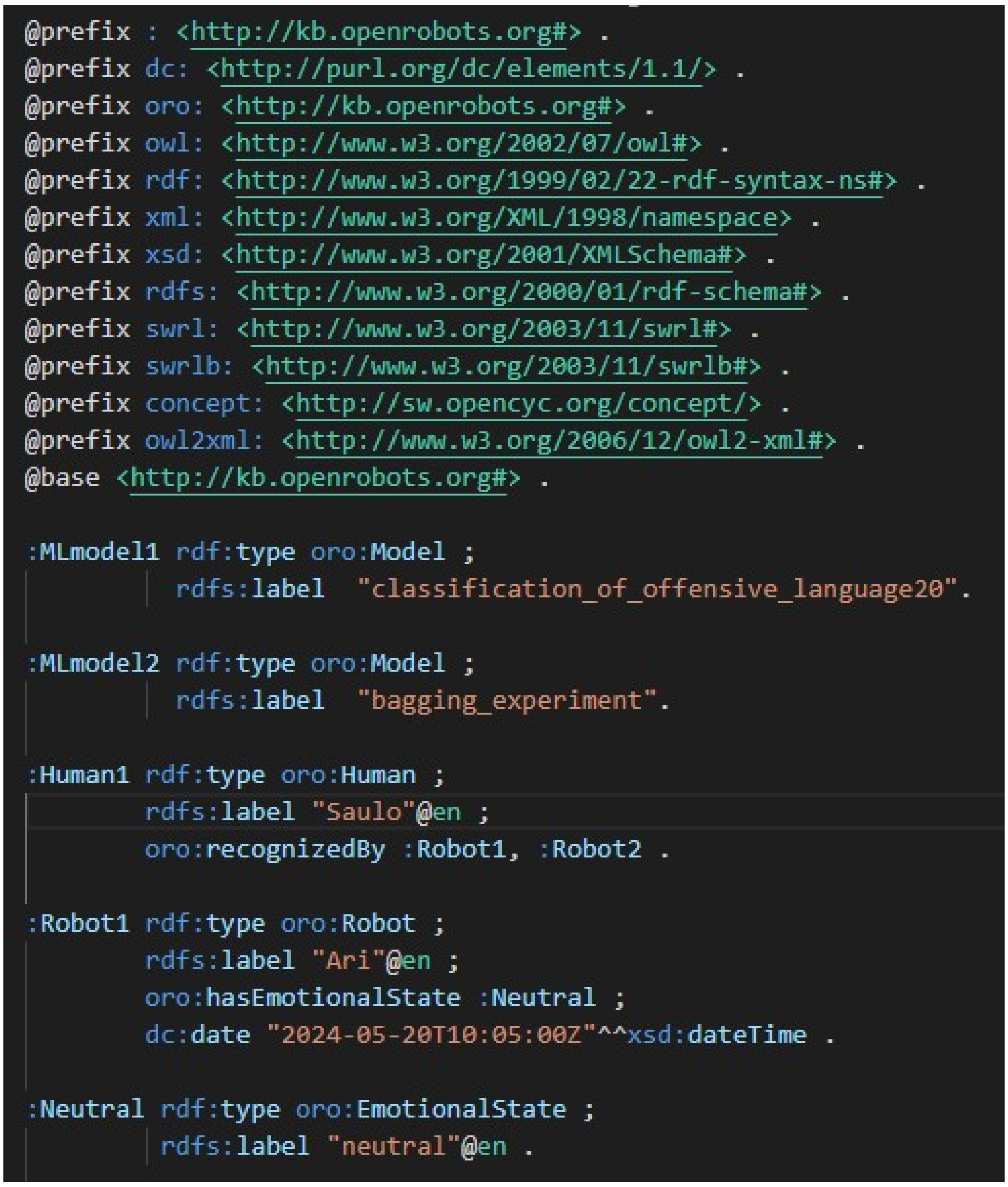
```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

(base) saulo@ubuntu:~$ cd Downloads/
(base) saulo@ubuntu:~/Downloads$ cat filter_interaction_logger.csv
time,final
2024-07-16-20:46:22,Despierta
2024-07-16-20:46:23,Que quieres hacer
2024-07-16-20:46:32,Como te llamas
2024-07-16-20:46:34,Mi nombre es Ari asistente de Ibermatica
2024-07-16-20:46:35,Cuantos años tienes
2024-07-16-20:46:37,Tengo 3 años en términos humanos
2024-07-16-20:46:38,Me puedes contar un chiste
2024-07-16-20:46:41,Cual es la fruta más divertida la naranjajaja
2024-07-16-20:46:43,Y como estas
2024-07-16-20:46:44,Me siento genial nunca mejor
2024-07-16-20:46:45,Duerme
```

Figure 5.4: Custom data logger logs and filter logs for Demo

5.3 Data analysis and visualization

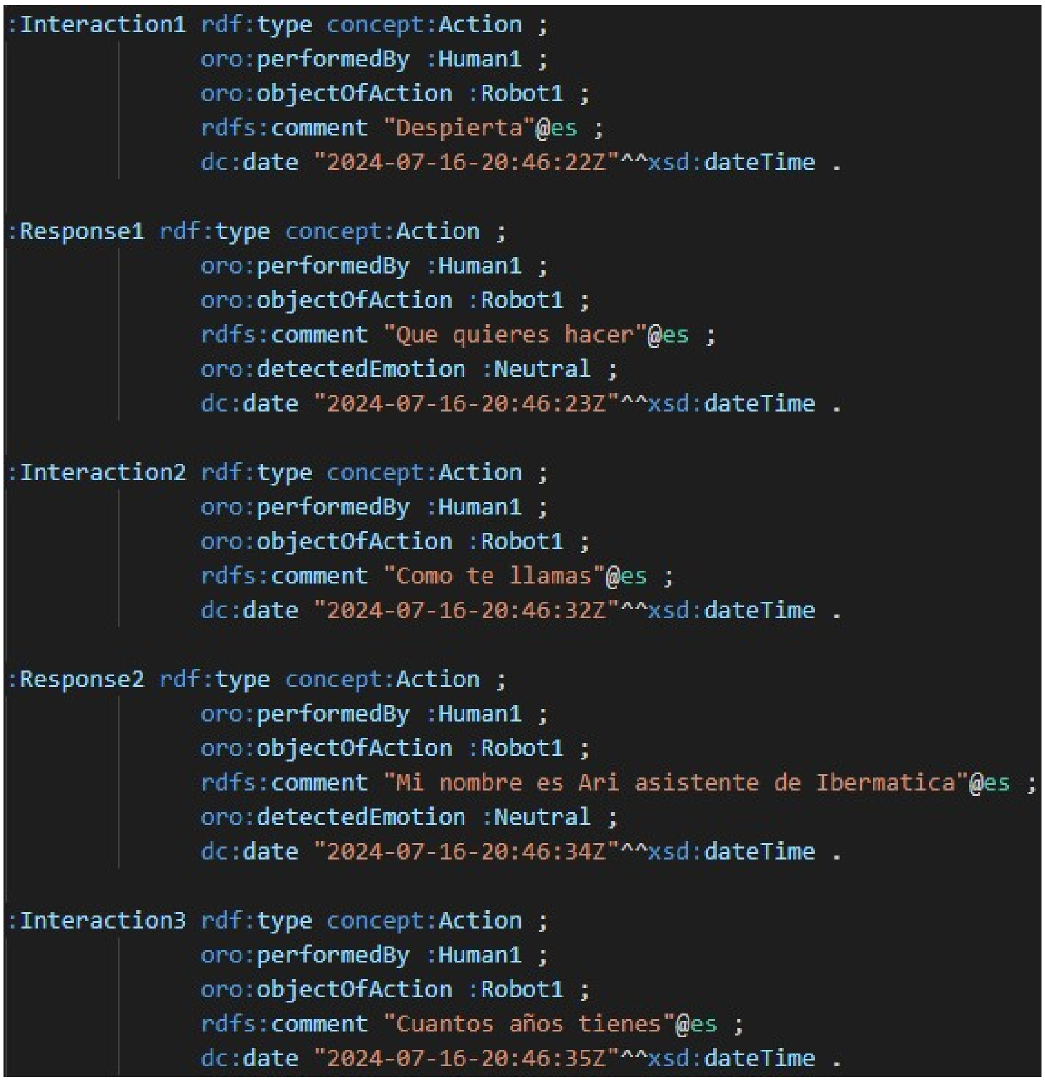
We have structured the robot interactions in a standard and easily interpretable format with the TTL file generated from the ontology in the previous step. This file contains the interactions between robots and humans and relevant information about the ML models used for this particular case study as illustrated in Fig. 5.5, in Fig. 5.6 and in Fig. 5.7 corresponding to the logs shown in Fig. 5.4. The next step is to integrate this file into our web interface developed with Streamlit, which allows users to load the TTL file directly and visualise the data in an intuitive and accessible way, as shown in Fig. 5.8, which in turn corresponds to the home screen of our web site.

Once the TTL file is loaded into the interface, a request is sent to our backend server, which is developed with Flask. This server processes the file using the KGTK framework, which converts the TTL file into a KG containing all the recorded interactions, resulting in a more manageable and structured set of files for exploration and analysis. The conversion process not only facilitates the manipulation of the graph but also allows the generation of different graph types, which helps to visualize patterns in the data, such as the frequency of specific interactions or the emotions detected in the robot's responses. All these



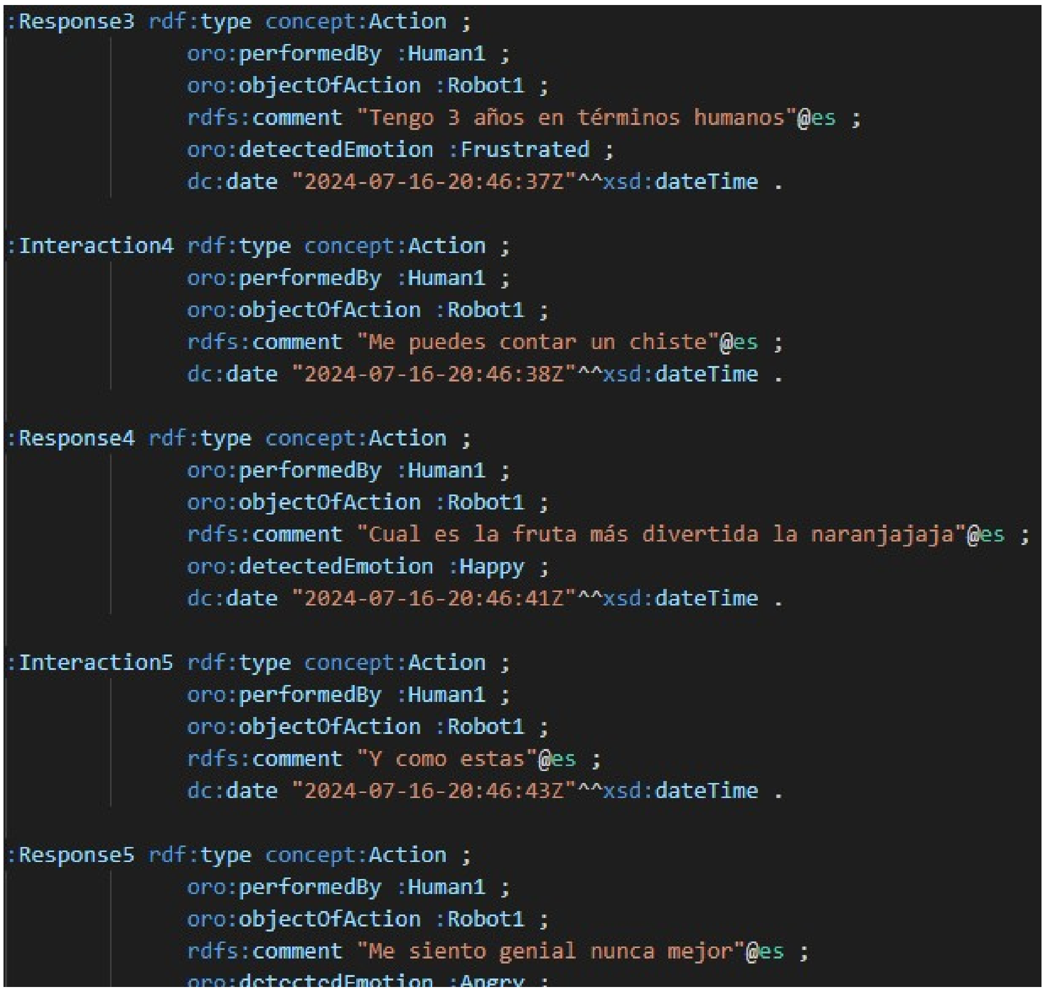
```
@prefix : <http://kb.openrobots.org#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix oro: <http://kb.openrobots.org#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl: <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb: <http://www.w3.org/2003/11/swrlb#> .
@prefix concept: <http://sw.opencyc.org/concept/> .
@prefix owl2xml: <http://www.w3.org/2006/12/owl2-xml#> .
@base <http://kb.openrobots.org#> .

:MLmodel1 rdf:type oro:Model ;
          rdfs:label  "classification_of_offensive_language20".

:MLmodel2 rdf:type oro:Model ;
          rdfs:label  "bagging_experiment".

:Human1 rdf:type oro:Human ;
        rdfs:label "Saulo"@en ;
        oro:recognizedBy :Robot1, :Robot2 .

:Robot1 rdf:type oro:Robot ;
        rdfs:label "Ari"@en ;
        oro:hasEmotionalState :Neutral ;
        dc:date "2024-05-20T10:05:00Z"^^xsd:dateTime .

:Neutral rdf:type oro:EmotionalState ;
         rdfs:label "neutral"@en .
```

Figure 5.5: TTL file with prefix definition, ML models and users for the described case study



```
:Interaction1 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Despierta"@es ;
              dc:date "2024-07-16-20:46:22Z"^^xsd:dateTime .

:Response1 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Que quieres hacer"@es ;
              oro:detectedEmotion :Neutral ;
              dc:date "2024-07-16-20:46:23Z"^^xsd:dateTime .

:Interaction2 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Como te llamas"@es ;
              dc:date "2024-07-16-20:46:32Z"^^xsd:dateTime .

:Response2 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Mi nombre es Ari asistente de Ibermatica"@es ;
              oro:detectedEmotion :Neutral ;
              dc:date "2024-07-16-20:46:34Z"^^xsd:dateTime .

:Interaction3 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Cuantos años tienes"@es ;
              dc:date "2024-07-16-20:46:35Z"^^xsd:dateTime .
```

Figure 5.6: TTL file with interaction definition for the described case study



```
:Response3 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Tengo 3 años en términos humanos"@es ;
              oro:detectedEmotion :Frustrated ;
              dc:date "2024-07-16-20:46:37Z"^^xsd:dateTime .

:Interaction4 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Me puedes contar un chiste"@es ;
              dc:date "2024-07-16-20:46:38Z"^^xsd:dateTime .

:Response4 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Cual es la fruta más divertida la naranjajaja"@es ;
              oro:detectedEmotion :Happy ;
              dc:date "2024-07-16-20:46:41Z"^^xsd:dateTime .

:Interaction5 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Y como estas"@es ;
              dc:date "2024-07-16-20:46:43Z"^^xsd:dateTime .

:Response5 rdf:type concept:Action ;
              oro:performedBy :Human1 ;
              oro:objectOfAction :Robot1 ;
              rdfs:comment "Me siento genial nunca mejor"@es ;
              oro:detectedEmotion :Angry ;
```

Figure 5.7: TTL file with more interaction definition for the described case study

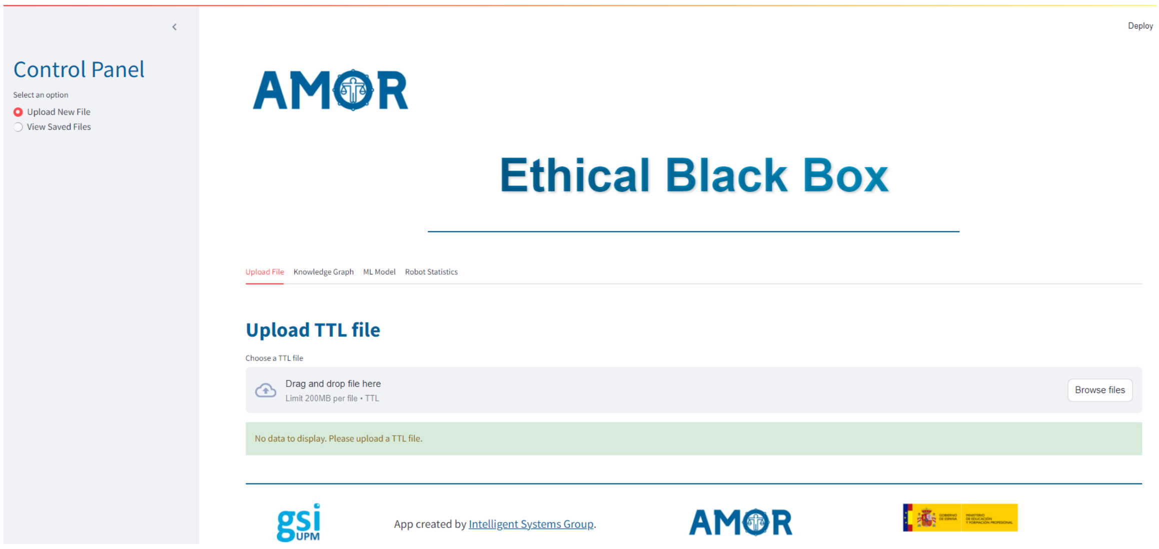
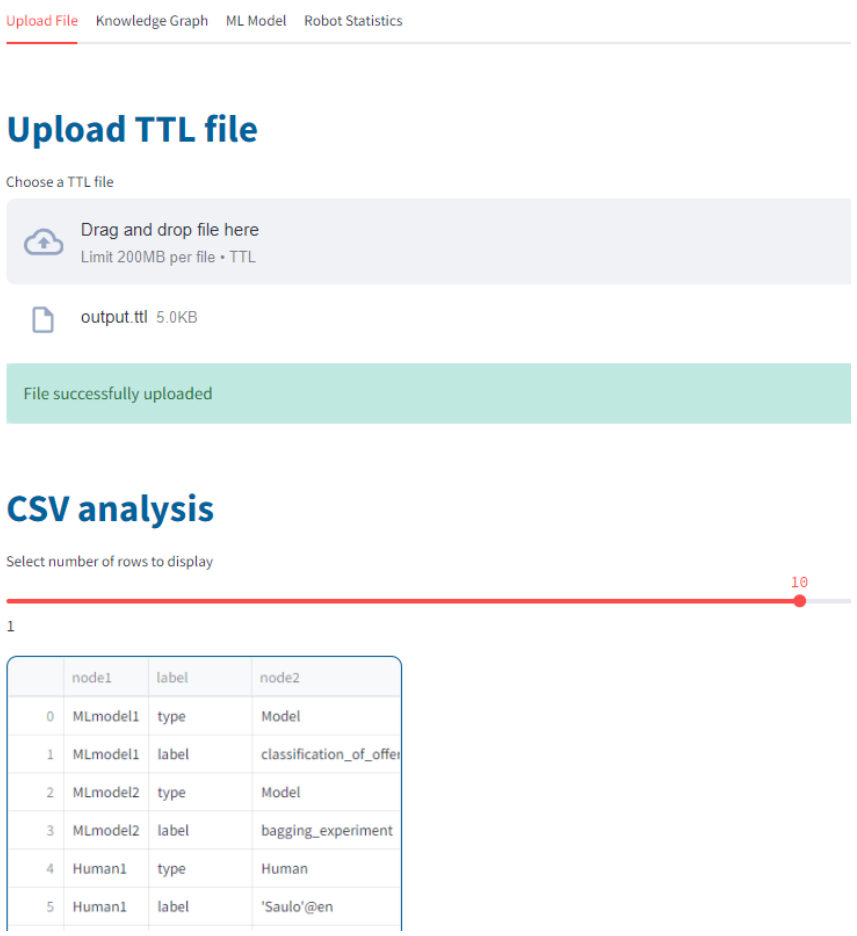


Figure 5.8: Website Home screen

functionalities are explained below.

Four primary tabs have been arranged in the web interface corresponding to our architecture's front end to facilitate navigation and analysis. The main tab, that is, the one displayed when entering the website, is the entry point where users can load the TTL file discussed above. Once uploaded, the interface displays a table with a representative sample of the information, allowing users to verify that the uploaded file is the correct one and also to see the structure of the data. In addition, this tab includes an interactive visualization of the KG, which represents the relationships between the different interactions and the information collected, allowing the connections to be explored intuitively and in-depth.


Upload File Knowledge Graph ML Model Robot Statistics

Upload TTL file

Choose a TTL file

Drag and drop file here
Limit 200MB per file • TTL

output.ttl 5.0KB

File successfully uploaded

CSV analysis

Select number of rows to display

10

1

	node1	label	node2
0	MLmodel1	type	Model
1	MLmodel1	label	classification_of_offe
2	MLmodel2	type	Model
3	MLmodel2	label	bagging_experiment
4	Human1	type	Human
5	Human1	label	'Saulo'@en

Figure 5.9: Website home screen with load file

When accessing the second tab of our interface, relevant information about the KG is displayed. This section details the number of principal and secondary nodes and the relationships that make up the graph, as shown in Fig. 5.10. This information is crucial to determine whether we are working with a large- or small-scale graph, allowing us to make appropriate decisions depending on its size. If we continue exploring this tab, we will find several interactive and detailed visualizations representing the KG. The first visualization corresponds to Fig. 5.10, where a graph is shown with the textual identifiers of each node, without distinction between the source and destination nodes. This representation provides an initial view of the graph structure and its components. The following visualization, Fig. 5.11, focuses on the relationships between nodes, providing a graphical representation similar to the previous one but on the existing connections between the different nodes. Finally, Fig. 5.12 presents a complete graph that combines the information of the two previous graphs, the nodes, and the relationships between them. This graph allows distinguishing the types of relationships by using different colors, assigned according to the nature of each connection. These visualizations facilitate the identification of patterns within the graph and offer a deeper and more dynamic understanding of its structure, becoming a valuable tool for further analysis and decision-making.

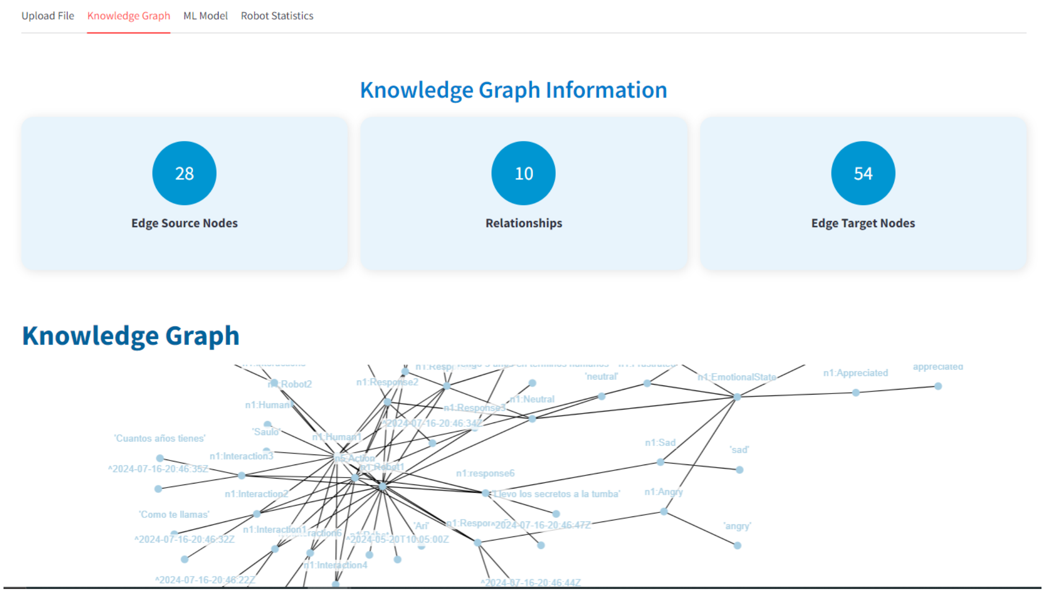


Figure 5.10: Second tab, with information and visualization of the knowledge graph

On the other hand, when accessing the third tab of our interface, we find all the details related to the ML models stored in our KG. These models correspond to the functionalities of the robot, such as identifying whether the user with whom it is interacting is someone it knows, for which it may have information about their name, face, or voice, or, on the other hand, it is a person it has not yet interacted with. Another critical function is emotion detection, where the robot analyses the user's emotional signals during the interaction. These are just some of the many functions that the robot could perform.

However, the robot is a relatively recent addition, and many of these functions are still in the testing phase, and others still need to be configured. Therefore, to illustrate what

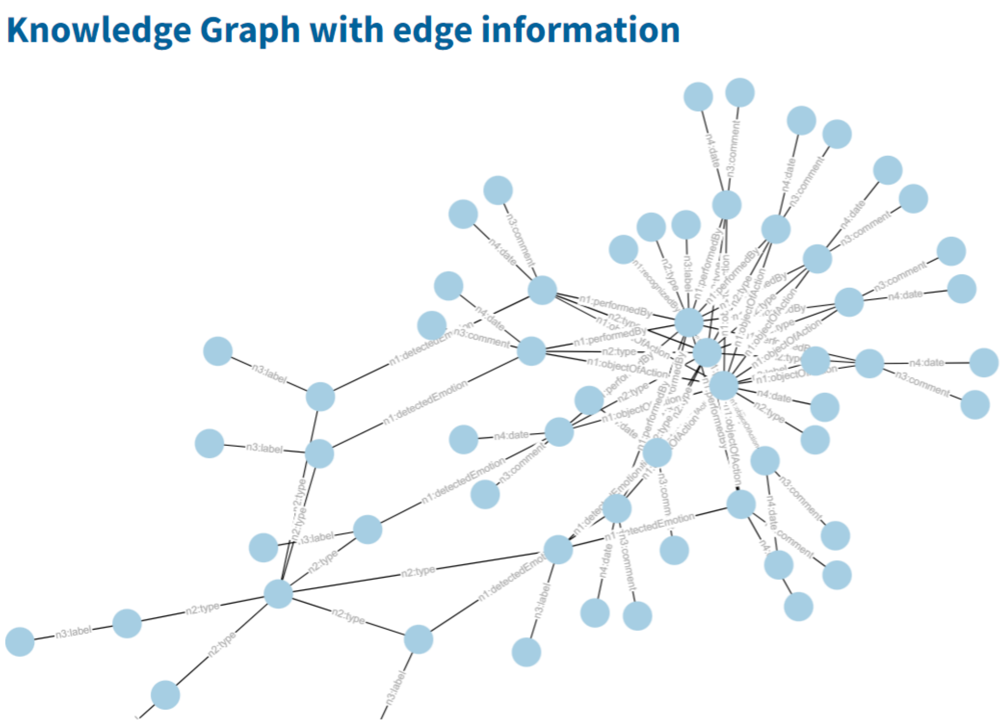


Figure 5.11: Knowledge graph with information on the relationships between nodes

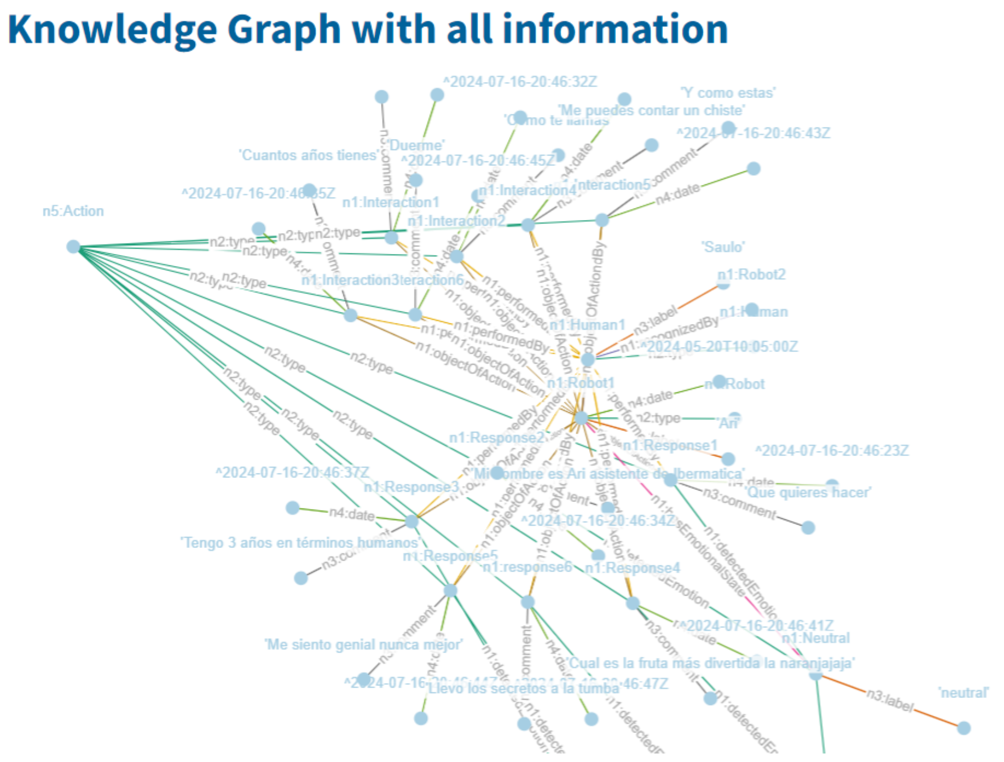
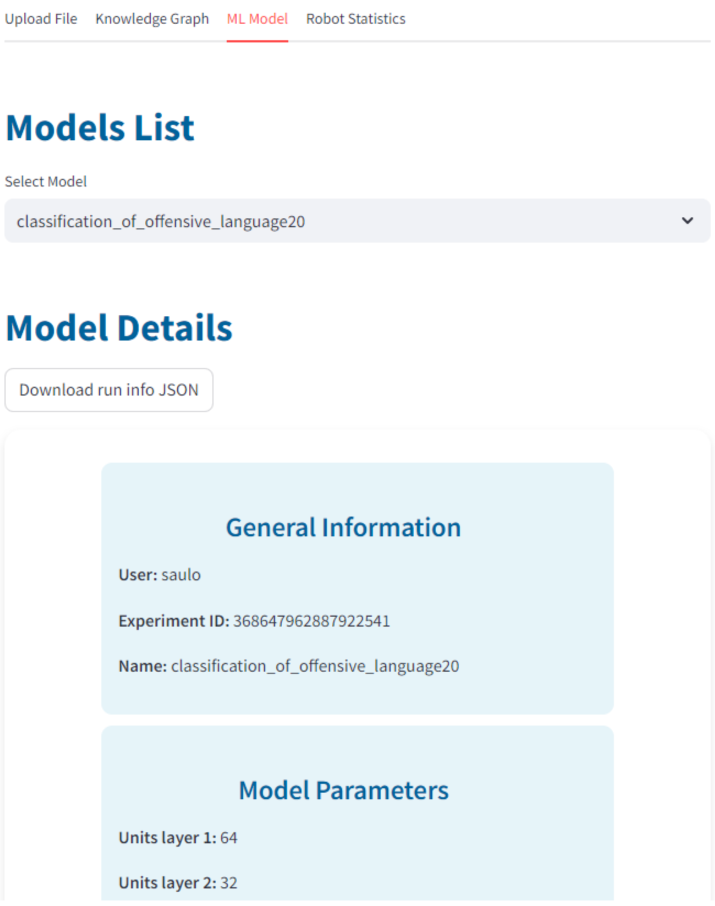


Figure 5.12: Knowledge graph with all the information of the nodes and their relationships and with color distinction according to the type of relationship between nodes

this section would look like in the entire operation, we have manually added two ML models that, although not directly related to these functionalities, allow us to visualize what kind of information could be obtained from the models and what analyses could be performed. These models have previously been manually loaded into MLflow employing a script and integrated into the TTL triples file by adding the name of the experiment created in MLflow for each model. When accessing this tab, the interface requests the MLflow Python API to obtain the list of all the models stored in that server for the KG, the parameters with which the model has been trained, and the characteristics obtained when it is executed. An example of this can be seen in Fig. 5.13, which represents the tab on the web page where the list of models and their parameters can be found. In turn, Fig. 5.14 and Fig. 5.15 represent some of the graphs found in this section of our website, which provide information on the results obtained by these models.

The fourth and last tabs analyze the information in the KG. This section presents critical data, such as the number of robots in the graph and the number of users these robots have recognized. In addition, the names of all registered robots and users are listed, giving us a clear understanding of the interaction participants. In this tab, we also find several informative and descriptive graphs. The first graph shows the evolution of emotions captured by robots from different users over time, allowing us to analyze emotional changes or unusual behaviors. The second graph presents a global distribution of the emotions detected by all the robots, providing an aggregated view of the emotional responses captured. An additional graph details the number of interactions that have taken place between each user and each robot, providing a complete perspective of how these interactions unfold. In addition, the tab includes an option to filter the information to focus on a specific robot, facilitating a more detailed analysis of the emotions captured by that particular robot. This is especially useful for more in-depth studies or fine-tuning specific network behaviors.

As mentioned in Chap. 4, this web interface is connected to an SQL database, which stores the files uploaded to the first tab of the interface. To access these stored files, we must go to the left-hand panel and select the "View Saved Files" option as a radio button. Clicking on this option will take us to a screen resembling the main screen; the difference is that on this new screen, a dropdown menu will appear containing the list of all the files we uploaded to the website and stored in the database. From this point onward, the interface and functionalities will be identical to those we have already seen and described for the main screen, where we manually uploaded the files. That is, we will be able to select and manage these files in the same way, with the additional advantage that we are now working with files previously stored in the database, which facilitates their management and organization over time, providing convenience and scalability to deal with large projects that require



Models List

Select Model

classification_of_offensive_language20

Model Details

General Information

User: saulo

Experiment ID: 368647962887922541

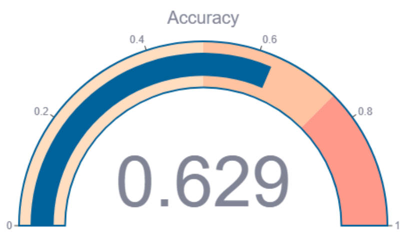
Name: classification_of_offensive_language20

Model Parameters

Units layer 1: 64

Units layer 2: 32

Figure 5.13: Machine learning models tab



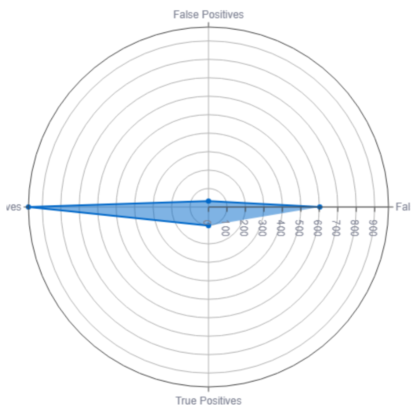
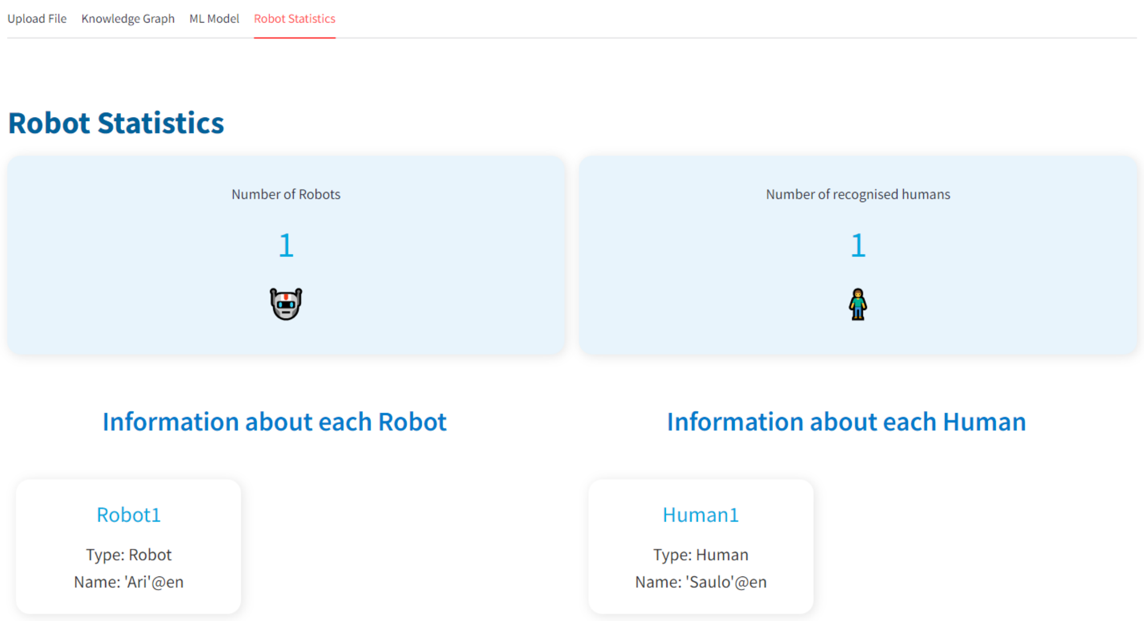


Figure 5.14: Accuracy Graphic for model experiment

Figure 5.15: Radar Graphic for model experiment



Robot Statistics

Number of Robots

1

Number of recognised humans

1

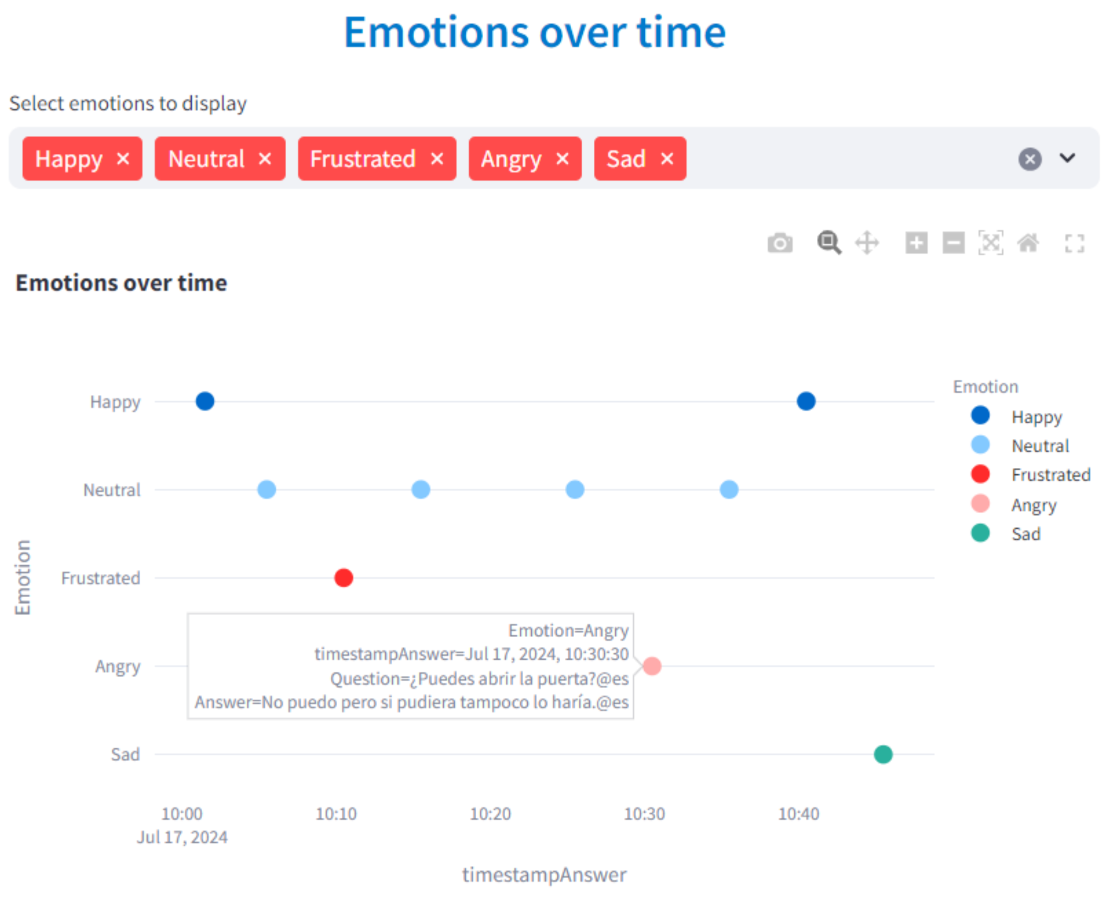
Information about each Robot

Information about each Human

Robot1
Type: Robot
Name: 'Ari'@en

Human1
Type: Human
Name: 'Saulo'@en

Figure 5.16: Robot Statistics tab



Emotion=Angry
timestampAnswer=Jul 17, 2024, 10:30:30
Question=¿Puedes abrir la puerta?@es
Answer=No puedo pero si pudiera tampoco lo haría.@es

Figure 5.17: Emotions over time graph

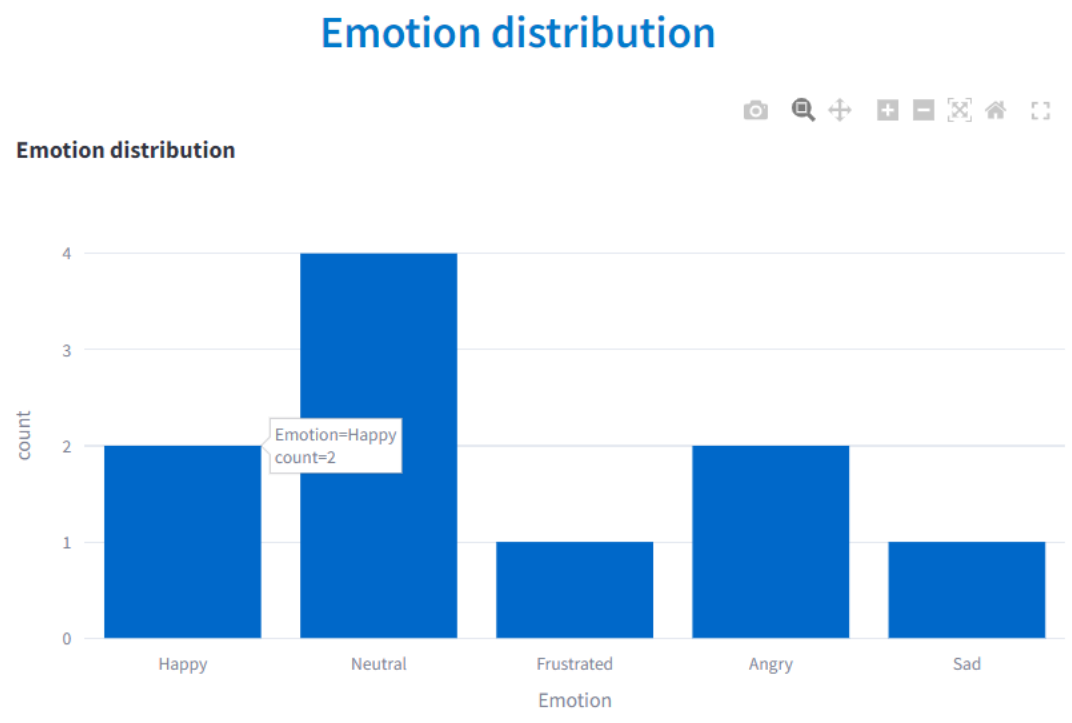


Figure 5.18: Emotions distribution graph

several KGs.

The presented case study underscores the functionality and importance of our ethical black-box system, applied to a concrete example of interaction between a robot and a human. This system allows for thorough observation and understanding of the entire process, from extracting the robot's interactions to processing and presentation in a web interface designed to manage and analyze the KGs generated by these interactions.

The web interface, structured in four main tabs, facilitates loading, visualization, and verification and provides interactive tools to intuitively explore the structure of the KGs and the associated ML models. This user-friendly design enables detailed and accessible analysis of the relationships and patterns that emerge from robot-human interactions. Ultimately, this interface represents an essential component of our ethical black-box system, allowing us to systematically and transparently record and monitor interactions between social robots and humans.

Finally, this case study demonstrates how this system ensures detailed and structured monitoring, providing a solid basis for audits, further analysis, and decision-making. In this way, our system significantly contributes to social robots' ethical and responsible use in different domains and collaboration with humans, underscoring its societal impact.

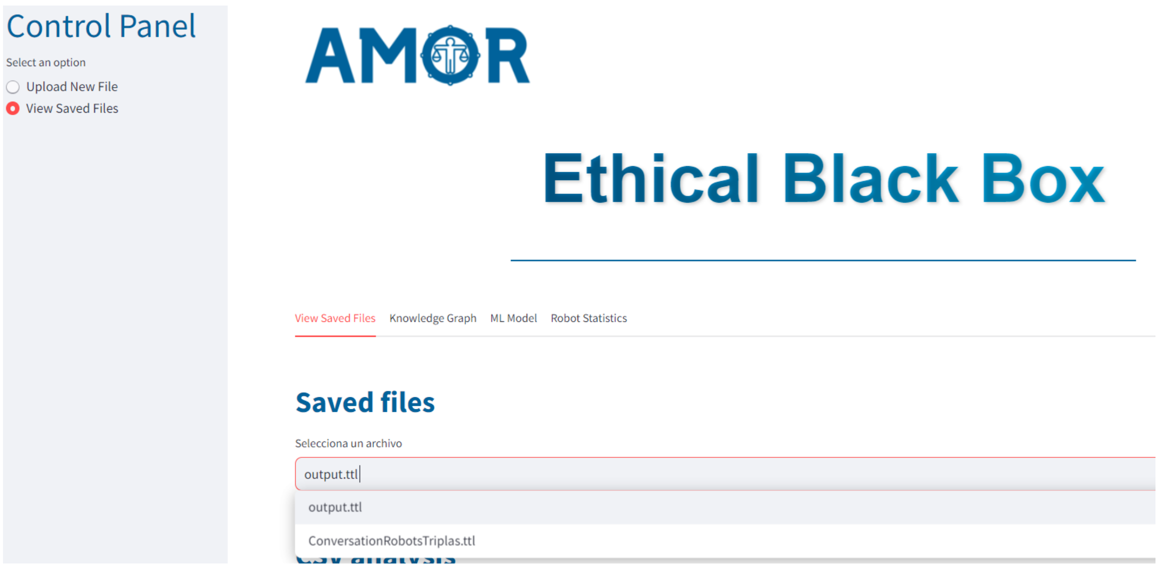


Figure 5.19: Page connected to SQL database

CHAPTER 6

Conclusions and future work

This chapter will describe the objectives achieved by the master's thesis following some of the key points developed in the project, as well as possible improvements that could be made in the future.

6.1 Conclusion

To conclude this master thesis document, it is important to recapitulate the developed system. We have created a system called "Ethical Black Box", designed to store and monitor the interactions of social robots, both with humans and with other robots. This system allows information to be represented through KGs, which facilitates auditing in case of detecting unusual behavior. Thanks to this tool, it is possible to record and analyze interactions with users of different personalities, cultures, and beliefs, allowing robots to adapt more effectively to each type of person.

This system is being developed within the scope of the AMOR project in strong collaboration with the GSI of the UPM. The main objective of the AMOR project is to foster critical thinking and emotional management about the information consumed on the media and social networks, helping to combat misinformation, hate speech, and the phenomenon of 'clickbait.' The "Ethical Black Box" system we have developed consists of a web interface that includes several modules for data loading and processing with the help of a Flask server. This server adapts the data into KGs, which can be visualized and analyzed through other modules on the same website. In addition, the stored interactions and ML models used by the social robots are also analyzed.

Two highly relevant lines of research have driven the development of this project. On the one hand, the detection of emotions in people of different ages, beliefs, cultures, and personalities, and on the other hand, the fight against disinformation and fake news. The information collected and analyzed in this project provides a solid basis for the AMOR project, allowing audits of both the datasets and the AI algorithms to be carried out. These audits are essential to improve or correct unusual or unethical behavior in social bots. Ultimately, this system represents a significant advance in the systems field using AI and its application in social bots, which are increasingly present in our society. It is designed to adapt to various situations and handle different graphs, ensuring its robustness and flexibility.

6.2 Achieved Goals

This project will explain the following objectives, highlighting the progress in developing and implementing the system. However, it is worth mentioning that some objectives could not be completed due to unforeseen complications, including the arrival of the social robots with which we worked. This slowed the project's progress and meant that what was initially planned in the pre-project regarding auditing the datasets could not be carried out in time.

The goals achieved for this project are the following.

- We have designed and implemented an Ethical Black Box system capable of storing and monitoring the interactions of social robots with humans and other robots.
- We have developed a web interface that facilitates data loading, visualization, and analysis. This interface allows users to work with KGs representing interactions and ML models extracted from social robots.
- We have processed and adapted the data to an established ontology, ensuring coherent and efficient information management for further analysis and auditing.
- We have tested our system to ensure it works in different situations and can handle various graphs.
- We have laid a solid foundation for future audits and improvements of datasets and AI algorithms, ensuring ethical and transparent behavior in social robots.

6.3 Future work

This section details possible improvements or new features that could be implemented in the project in the future:

- Automate the system's processes, reducing the need for manual intervention in capturing and analyzing interactions between social bots and users, which would optimize the efficiency and accuracy of the system.
- Migrate the system to a centralized server in the cloud, allowing easier access to stored data and facilitating the system's scalability for implementation in different environments.
- Integrate different KGTK modules to perform embeddings of KGs obtained. This could generate valuable information, such as identifying behavioral patterns in users with similar profiles, predicting emotional behavior in specific situations, or detecting possible ethical conflicts in robot-human interactions.
- Improve the web interface by adding explanations and context to data visualizations, thus facilitating the interpretation and understanding of the information by users and enhancing the use of graphs in ethical decision-making.

- Realise audits of datasets using the information collected to ensure the integrity and quality of the data and identify possible biases or errors in the AI algorithms used.

APPENDIX A

Impact of this project

This final thesis project promotes critical thinking and emotional management when consuming information in media and social networks. In a context where we are exposed to a constant flow of information, it is crucial to differentiate truthful information from manipulated information that seeks to influence users' opinions and actions. This project addresses problems such as misinformation, hate speech, and clickbait, helping users assess the credibility of sources and form informed opinions. To this end, several general objectives have been set, focusing on the use of social bots that interact with humans, recording these interactions in an ethical black box, and being able to use tools that audit behavior in case irregularities are detected. This approach facilitates the analysis of the behavior of users with different personalities, cultures, or beliefs, allowing social bots to better adapt to the people with whom they interact. Sect. A.3, Sect. A.4 and Sect. A.5 of this appendix discuss the social, economic, and environmental impact of the project. Sect. A.2 explores possible ethical and professional implications.

A.1 Introduction

This annex explores the ethical, economic, social, and environmental aspects of designing and developing an ethical black box for social robots. It analyses the relevant impacts of creating and implementing this system, considering its possible repercussions in different fields. In this way, it intends to offer a vision of the implications of this project for society and other environments.

A.2 Ethical Implications

This project, which uses social robots to store interactions and analyze moral sentiments and emotions, presents significant ethical challenges. On the one hand, it is essential to guarantee the privacy of personal data collected during interactions, ensuring their protection against unauthorized access. In addition, transparency in how decisions are made and analyses are generated is crucial for users to understand and trust the system. Finally, avoiding bias or inappropriate behavior that could lead to discrimination is essential, ensuring equal treatment for people of different personalities, cultures, or beliefs.

A.3 Social Impact

This thesis, linked to the abovementioned AMOR project, has a significant potential social impact. By helping users develop critical thinking and better manage their emotions in the face of information, this system can contribute to a more informed society that is less susceptible to misinformation and hate speech. Moreover, social bots promote inclusion and respect for diversity by adapting to different personalities and cultures. However, these technologies must be accessible to all, regardless of their socioeconomic background, to avoid a digital divide that could exclude certain groups from the benefits offered by the system.

A.4 Economic Impact

This project may also generate critical economic impacts. On the one hand, implementing this system requires initial investments in technological infrastructure and trained personnel, which can represent a significant cost. However, in the long term, the ability of these robots to adapt to the environment and improve interaction with users can optimize resources in numerous sectors such as education, customer service, or catering, among others, reducing operating costs. In addition, this system can foster greater trust in information and media, which translates into economic benefits for platforms that promote reliable and ethical content.

A.5 Environmental Impact

There are also specific critical environmental impacts that must be considered. These include the technological infrastructure required to store and process large volumes of data, especially in cloud data centers, which generate significant energy consumption, contributing to an increase in the carbon footprint. In addition, the maintenance and operation of social robots involve additional energy expenditure, which must also be considered. Therefore, exploring different methods to improve this energy efficiency in the data centers and the robots is essential, thus minimizing the associated environmental impact and promoting more sustainable development of the system and the project.

APPENDIX B

Economic budget

This appendix summarizes the costs involved in developing this work, including human resources, materials, and possible licenses.

B.1 Human resources

This section considers the time spent on this system's design, development, and testing. To find the cost of the project's development, we will give an approximation based on the average salary of a Telecommunication Engineer.

To make this estimation, we have used the European Credit Transfer System (ECTS) credits of the master thesis. There are 12 ECTS, each of which corresponds to about 25 hours of work, for a total of approximately 300 hours of work. Assuming that the hourly rate for a junior telecommunications engineer is about 15 euros, this adds up to 4500 euros.

This cost includes designing, developing, and testing the tasks for creating the system. The cost of maintaining and operating the system is considered low.

B.2 Material resources

A Huawei Matebook D laptop computer with the following technical specifications was used for the development of the project:

- Processor: Intel Core i7 8550U 8M Cache 4.00 GHz.
- Graphics: NVIDIA GeForce MX150 2GB.
- Memory: 8GB RAM.
- Hard Disk: SSD 256GB.
- Operating System: Genuine Windows 10 Professional.

If purchased in the year this project was developed, this laptop would cost approximately 1200 euros.

B.3 Licenses

This project's software is open source, so no additional cost has been incurred.

B.4 Total Budget

In summary, the project's total cost will be the sum of human and material resources, which is 5700 euros. Knowing that the project has been developed and commercialized in Spain, the corresponding applicable IVA, which in this case is 21% , must be added to this figure. Therefore, the final budget will be €6,897.

Bibliography

[1] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. Pykeen 1.0: A python library for training and evaluating knowledge graph embeddings. *CoRR*, abs/2007.14175, 2020.

[2] Daniel J Barrett, Richard E Silverman, and Robert G Byrnes. *SSH, The Secure Shell: The Definitive Guide: The Definitive Guide.* " O'Reilly Media, Inc.", 2005.

[3] Jacob Biba. What is a social robot? — built in. `https://builtin.com/robotics/social-robot`, 2024. Accessed: 2024-04-01.

[4] Oscar Araque; J. Fernando Sánchez-Rada; Sergio Muñoz López; Álvaro Carrera Barroso Carlos A. Iglesias. Project - amor. análisis de sentimiento moral en datos textuales. `https://gsi.upm.es/es/investigacion/proyectos`.

[5] Scott Clifford and Jennifer Jerit. How words do the work of politics: Moral foundations theory and the debate over stem cell research. *The Journal of Politics*, 75(3):659–671, 2013.

[6] QuestionPro Collaborators. Data bias: Identifying and reducing in surveys and analytics. `https://www.questionpro.com/blog/data-bias/#Different_Types_of_Data_Bias`, 2024. Accessed: 2024-03-18.

[7] Vineesh Cutting and Nehemiah Stephen. A review on using python as a preferred programming language for beginners. *International Research Journal of Engineering and Technology*, 8:4258–4263, 08 2021.

[8] Daniel. Nlp natural language processing : Introducción. `https://datascientest.com/es/nlp-introduccion`, 2024. Accessed: 2024-03-18.

[9] Davis David. Cómo usar streamlit y python para crear una aplicación de ciencia de datos — hackernoon. `https://hackernoon.com/es/como-usar-streamlit-y-python-para-crear-una-aplicacion-de-ciencia-de-datos`, 2024. Accessed: 2024-06-18.

[10] Rosa Martínez Rubio Esther Sánchez Lucas. Las ontologÍas y su aplicaciÓn en el Ámbito de la documentaciÓn. `https://personales.upv.es/ccarrasc/doc/2000-2001/Ontolog%C3%ADas/Index.htm`, 2024. Accessed: 2024-09-05.

[11] Flask. Welcome to flask — flask documentation (3.0.x). `https://flask.palletsprojects.com/en/3.0.x/`, 2024. Accessed: 2024-08-21.

[12] Python Software Foundation. Welcome to python.org. https://www.python.org/, 2024. Accessed: 2024-08-21.

[13] Leonardo J. Caballero G. 5.1. streamlit - entrenamiento de data scientist en python - nivel básico. https://entrenamiento-data-scientist-python.readthedocs.io/leccion5/streamlit.html, 2024. Accessed: 2024-06-18.

[14] Devndra Ghimire. Comparative study on python web frameworks: Flask and django. *Metropolia, University of Applied Sciences*, 2020.

[15] Alexander S. Gillis. What is machine learning bias? — definition from whatis. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-bias-algorithm-bias-or-AI-bias, 2024. Accessed: 2024-03-18.

[16] Carlos Gonzalo. Ontologías y la web semántica. https://www.carlosgonzalo.es/ontologias-y-la-web-semantica/, 2024. Accessed: 2024-09-04.

[17] Google. Equidad: Tipos de sesgo— machine learning — google for developers. https://developers.google.com/machine-learning/crash-course/fairness/types-of-bias.

[18] Jindong Gu and Daniela Oelke. Understanding bias in machine learning, 2019.

[19] Jonathan Haidt. *The righteous mind: Why good people are divided by politics and religion.* Vintage, 2012.

[20] Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Rogers, Ronpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe, and Pedro Szekely. KGTK: A toolkit for large knowledge graph manipulation and analysis. In *International Semantic Web Conference*, pages 278–293. Springer, 2020.

[21] Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Rogers, Ronpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe, and Pedro Szekely. import-ntriples - kgtk documentation. https://kgtk.readthedocs.io/en/latest/import/import_ntriples/#input-files, 2024. Accessed: 2024-07-28.

[22] Filip Ilievski, Daniel Garijo, Hans Chalupsky, Naren Teja Divvala, Yixiang Yao, Craig Rogers, Ronpeng Li, Jun Liu, Amandeep Singh, Daniel Schwabe, and Pedro Szekely. Kgtk documentation. https://kgtk.readthedocs.io/en/latest/#how-to-cite, 2024. Accessed: 2024-07-28.

[23] Redacción KeepCoding. Lematización en python — keepcoding bootcamps. https://keepcoding.io/blog/que-es-la-lematizacion-en-python/, 2024. Accessed: 2024-03-18.

[24] KGTK. Kgtk documentation. https://kgtk.readthedocs.io/en/latest/, 2024. Accessed: 2024-04-17.

[25] Selina Khoirom, Moirangthem Sonia, Borishphia Laikhuram, Jaeson Laishram, and Tekcham Davidson Singh. Comparative analysis of python and java for beginners. *Int. Res. J. Eng. Technol*, 7(8):4384–4407, 2020.

[26] Adriano Koshiyama, Emre Kazim, Philip Treleaven, Pete Rai, Lukasz Szpruch, Giles Pavey, Ghazi Ahamat, Franziska Leutner, Randy Goebel, Andrew Knight, Janet Adams, Christina Hitrova, Jeremy Barnett, Parashkev Nachev, David Barber, Tomas Chamorro-Premuzic, Konstantin Klemmer, Miro Gregorovic, Shakeel Khan, and Elizabeth Lomas. Towards algorithm auditing: A survey on managing legal, ethical and technological risks of ai, ml and associated algorithms. *SSRN Electronic Journal*, 2021.

[27] Séverin Lemaignan, Raquel Ros, Lorenz Mösenlechner, Rachid Alami, and Michael Beetz. Oro, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553, 2010.

[28] Lu Liang, Yong Li, Ming Wen, and Ying Liu. Kg4py: A toolkit for generating python knowledge graph and code semantic search. *Connection Science*, 34(1):1384–1400, May 2022.

[29] Aitor Martín Romo. Design and Development of Automatization Pipelines for Machine Learning Projects based on the MLOps Frameworks. Tfm, Universidad Politécnica de Madrid, ETSI Telecomunicación, February 2024.

[30] Charles Tapley Max Berrendorf. pykeen/pykeen: A python library for learning and evaluating knowledge graph embeddings. `https://github.com/pykeen/pykeen?tab=readme-ov-file`, 2024. Accessed: 2024-04-19.

[31] Wes McKinney and PD Team. Pandas-powerful python data analysis toolkit. *Pandas—Powerful Python Data Analysis Toolkit*, 1625, 2015.

[32] MLflow. mlflow — mlflow 2.15.0rc0 documentation. `https://mlflow.org/docs/latest/python_api/mlflow.html#module-mlflow`, 2024. Accessed: 2024-07-28.

[33] a Series of LF Projects MLflow Project. What is mlflow? — mlflow 2.12.2 documentation. `https://mlflow.org/docs/latest/introduction/index.html#`, 2024. Accessed: 2024-05-16.

[34] LLC MLflow Project, a Series of LF Projects. Mlflow model registry — mlflow 2.15.1 documentation. `https://mlflow.org/docs/latest/model-registry.html`, 2024. Accessed: 2024-08-21.

[35] LLC MLflow Project, a Series of LF Projects. Mlflow tracking — mlflow 2.15.1 documentation. `https://mlflow.org/docs/latest/tracking.html`, 2024. Accessed: 2024-08-21.

[36] Grzegorz Mrukwa. Machine learning development process: From data collection to model deployment. `https://www.netguru.com/blog/machine-learning-development-process#understanding-the-concept-of-machine-learning`, 2024. Accessed: 2024-03-18.

[37] Mohammad Robihul Mufid, Arif Basofi, M. Udin Harun Al Rasyid, Indhi Farhandika Rochimansyah, and Abdul rokhim. Design an mvc model using python for flask framework development. In *2019 International Electronics Symposium (IES)*, pages 214–219, 2019.

[38] NumFOCUS. pandas - python data analysis library. `https://pandas.pydata.org/about/index.html`, 2024. Accessed: 2024-08-21.

[39] USC University of Southern California. Knowledge graph toolkit. `https://usc-isi-i2.github.io/kgtk/`, 2024. Accessed: 2024-04-17.

[40] Plotly. Graph objects in python. `https://plotly.com/python/graph-objects/`, 2024. Accessed: 2024-08-21.

[41] Plotly. Plotly express in python. `https://plotly.com/python/plotly-express/`, 2024. Accessed: 2024-08-21.

[42] Plotly. Plotly python graphing library. `https://plotly.com/python/`, 2024. Accessed: 2024-08-21.

[43] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. Clickbait detection. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*, pages 810–817. Springer, 2016.

[44] Antonio Pérez, M. Isabel García, Manuel Nieto, José L. Pedraza, Santiago Rodríguez, and Juan Zamorano. Argos: An advanced in-vehicle data recorder on a massively sensorized vehicle for car driver behavior experimentation. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):463–473, 2010.

[45] Astrid Duque Ramos. Ontologías en inteligencia artificial: sus aplicaciones y métodos para evaluar su calidad – cio. `https://cio.umh.es/2021/12/01/ontologias-en-inteligencia-artificial-sus-aplicaciones-y-metodos-para/-evaluar-su-calidad/`, 2024. Accessed: 2024-09-05.

[46] Ajay Rawat. A review on python programming. *International Journal of Research in Engineering, Science and Management*, 3(12):8–11, 2020.

[47] PAL Robotics. How-to: Log and retrieve data from the robot - pal sdk 23.12 documentation. `https://docs.pal-robotics.com/ari/sdk/23.12/management/data-logging.html#data-logging`, 2024. Accessed: 2024-07-28.

[48] PAL Robotics. The webcommander tool - pal sdk 23.12 documentation. `https://docs.pal-robotics.com/ari/sdk/23.12/management/webcommander.html`, 2024. Accessed: 2024-07-28.

[49] Iuliana Sandu, Menno Wiersma, and Daphne Manichand. Time to audit your ai algorithms. *Maandblad voor Accountancy en Bedrijfseconomie*, 96(7/8):253–265, September 2022.

[50] Nanditha Saravanakumar. Data bias - definition, examples, types, how to avoid? `https://www.wallstreetmojo.com/data-bias/`, 2024. Accessed: 2024-03-18.

[51] Streamlit. Streamlit documentation. https://docs.streamlit.io/, 2024. Accessed: 2024-07-28.

[52] streamlit. streamlit/streamlit: Streamlit — a faster way to build and share data apps. https://github.com/streamlit/streamlit, 2024. Accessed: 2024-06-18.

[53] Iryna Sydorenko. What is a dataset in machine learning: The complete guide — label your data. https://labelyourdata.com/articles/what-is-dataset-in-machine-learning, 2024. Accessed: 2024-03-18.

[54] Jake B. Telkamp and Marc H. Anderson. The implications of diverse human moral foundations for assessing the ethicality of artificial intelligence. *Journal of Business Ethics*, 178(4):961–976, February 2022.

[55] UNIR. Nlp (natural language processing): ¿qué es y para qué sirve? https://www.unir.net/marketing-comunicacion/revista/nlp-procesamiento-lenguage-natural/, 2024. Accessed: 2024-03-18.

[56] Wikipedia. Robot social - wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Robot_social, 2024. Accessed: 2024-04-01.

[57] Alan F. T. Winfield and Marina Jirotka. *The Case for an Ethical Black Box*, page 262–273. Springer International Publishing, 2017.

[58] Alan F. T. Winfield, Anouk van Maris, Pericle Salvini, and Marina Jirotka. An ethical black box for social robots: a draft open standard, 2022.

[59] Khensani Xivuri and Hossana Twinomurinzi. *A Systematic Review of Fairness in Artificial Intelligence Algorithms*, page 271–284. Springer International Publishing, 2021.

[60] Kinza Yasar. What is black box ai? definition from techtarget. https://www.techtarget.com/whatis/definition/black-box-AI, 2024. Accessed: 2024-03-18.

[61] Hanyu Zheng. Gender bias in hiring algorithms. amazon's hiring algorithm was once... — by hanyu zheng — medium. https://medium.com/@hzhe0050/gender-bias-in-hiring-algorithms-8a39ea49a594, 2024. Accessed: 2024-03-18.