Robot Explain Yourself
Enhancing Human-Robot Communication with Large Language Models

Willem Tom Romain - Adnet

June 21, 2025

Abstract

This report presents the design and implementation of a system for enhancing human-robot interaction by enabling a robot to explain its decisions, particularly those related to low-level perception data, through natural language using a Large Language Model (LLM). The project explores current methodologies, builds an integration framework, fine-tunes an LLM, and validates the system through user evaluations.

Contents

1 Different phases of the project

1.1 Phases of the project

The project started in March 2025 and as been divided into several phases. The main phases are:

- **Phase 1: Literature review** - Research existing HRI frameworks and LLM applications.
- **Phase 2: Framework design** - Develop a modular architecture for the HRI system.
- **Phase 3: LLM customization** - Fine-tune LLMs for specific HRI tasks.
- **Phase 4: Evaluation** - Test and validate the system with real users.
- **Phase 5: Documentation and reporting** - Document the framework, results, and lessons learned.

1.2 Project timeline

So firstly to understand what have already been done and what is needed was the main goal of the first phase. So I had to do a literature review of the current state of the art in HRI and LLMs but more generaly to understand how does work the local LLMs. This part took around 2-4 weeks but in fact it never ended because I always find new papers that can be useful for the project. Moreover, you always found new informations or new points that can make your LLM more useful but also more efficient. As a second phase I had to design the framework that will be used to answer the user questions. This part was quite easy but took some time (4-6 weeks)since we still have to find the most efficient way to do it and to choose the language to use, how to structure everything and how to make it easy to use. The third phase was to customize the LLM to make it more efficient and to make it able to answer the user questions. Next, the evaluation phase focused on optimizing the framework by rewriting parts of the code for improved efficiency and usability, as well as developing better prompts for the LLM. this previous phase took around 2-3 weeks. Finally, the documentation and reporting phase involved writing this report and preparing the final presentation, which took about 1-2 weeks.

2 Introduction

The integration of artificial intelligence with robotics has opened new frontiers in human-robot interaction (HRI). As robots become increasingly autonomous and deployed in complex real-world environments, the need for transparent and interpretable decision-making becomes paramount. This project addresses the challenge of making robotic systems more explainable by leveraging Large Language Models (LLMs) to translate low-level sensor data and perception information into natural language explanations.

2.1 Problem statement

Modern robots operate using complex algorithms that process vast amounts of sensor data to make navigation and behavioral decisions. However, these decisions often remain opaque to human users, creating a barrier to trust and effective collaboration. The challenge lies in bridging the gap between machine perception and human understanding.

The primary goal is to develop an AI-powered robot capable of evaluating its past decisions, particularly when revisiting locations. For example, the robot should be able to explain: "I recognize this area—I previously visited it on [date/time] and made a certain decision." If the environment has changed since the last visit, the robot should update its reasoning to reflect the new context, rather than relying solely on prior experiences.

2.2 Objectives

This project aims to:

- Design a framework for integrating LLMs with robotic perception systems
- Develop a prototype system that can explain robot path decisions in natural language
- Evaluate the effectiveness of LLM-generated explanations in enhancing human understanding
- Assess the impact on user trust and satisfaction in human-robot interactions

3 Literature review

3.1 Technologies and methodologies in Human-Robot Communication

Dialogue management in HRI : The current state of dialogue management in human-robot interaction is reviewed in survey of dialogue management in human-robot interaction [3]. They evaluate capabilities, methods, and challenges, and emphasize the need for structured approaches that effectively combine HRI with dialogue systems. They show how to discuss properly with a robot powered by Artificial Intelligence but also why it's important to understand the interactions.

Human–robot interaction : An overview of human–robot interaction is provided in wikipedia : Human-robot interaction [6]. The idea behind this article is to show that to interact we need to create a safe and intuitive interaction : reducing friction. To be specialize into a specific area is the key point to create this safe zone of communication. Involving also feelings, emotions and gesture makes the conversartions more real and push the user to feel confortable and then to continue this conversation/interactions.

3.2 Interpretability of low-level perception in robotics

Explainable robotic systems in scenarios : Puiutta and Veith [1] discuss explainability in robotic reinforcement learning agents. Explainable robotic systems are crucial for human-robot collaboration. This study compares three approaches to computing the probability of success for reinforcement learning agents in robotic scenarios, finding learning-based and introspection-based approaches to be suitable alternatives to a memory-based baseline.

Subsumption architecture : The subsumption architecture [7] is a reactive robotic architecture that decomposes behavior into sub-behaviors organized into a hierarchy of layers. Each layer implements a level of behavioral competence, with higher layers subsuming lower ones to create viable behavior. This architecture emphasizes iterative development, task-specific perception, and parallel control, enabling real-time interaction with dynamic environments.

3.3 Challenges and opportunities for understandable robotics

Trust in explainable robots : Trust calibration and explanation-specificity in robots are discussed in Wang et al. [5], who offer insight into designing systems that feel transparent and reliable. Explainable AI (XAI) and explainable robots literature aims to enhance human understanding and human-robot team performance. The paper discusses three trust-related considerations for explainable robot systems: bases of trust, trust calibration, and trust specificity.

Effects of robot explanations : The authors of [4] further evaluate the impact of robot explanations on user perception. Their findings indicate that explanations make robots appear more lively and human-like, leading to more interactive conversations and an increased likelihood of users believing the robot.

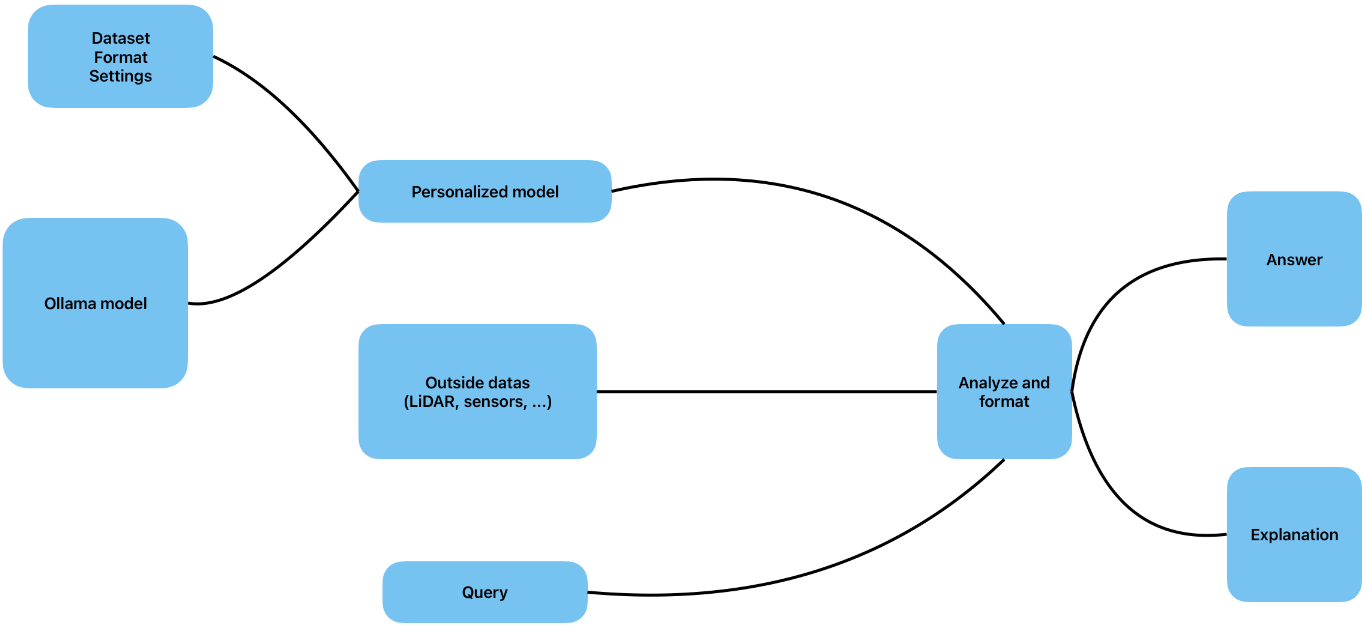
Ethical black box : Professor Marina Jirotka's concept of the "ethical black box" [2] suggests embedding inflight recorders in robots to promote accountability and auditability of robotic decisions.

4 Framework Design

4.1 Architecture Overview

The proposed framework integrates three main components:

- **Perception processing layer :** Handles sensor data aggregation and context extraction
- **LLM integration layer :** Manages prompt generation and response processing
- **Human interface layer :** Provides natural language interaction capabilities



4.2 framework explanations

The current framework is designed to take care of all the previous informations and to provide a good explanation to the user. To do so we firstly choose a good LLM model that can provide us with the best answers and explanations. The we are going to give to the robot some real-time data about the environment, the sensors and the path that it is currently following or the availabale ones. The we are going to use the LLM to generate a prompt that will be used to answer the user question. This process is done in a loop, where the user can ask multiple questions and the robot will answer them using the LLM. Thanks to this way of doing the user can adapt the question to the answer given by the robot but also add some new informations about the different path that the robot can take.

4.3 Installation of the Project

The project is designed to be easily deployable on any computer with minimal setup requirements. The following prerequisites are needed:

- **Python 3.8+** (developed under: Python 3.11)
- **Ollama** - Local LLM server for model hosting

- **make** (optional, for automation)
- **git** for repository cloning

4.3.1 Step-by-Step installation process

1. Repository cloning

The project can be obtained from the GitHub repository:

```
git clone https://github.com/Vlor999/HCI.git
cd HCI
```

2. Ollama installation and setup

Ollama serves as the local LLM server, providing the computational backend for natural language processing:

- Download and install Ollama from `https://ollama.com/download`
- Start the Ollama server: `ollama serve`
- Pull the required model: `ollama pull llama3.2` (or any other model)

3. Project environment setup

The project uses automated setup through make commands:

```
make init
make install
```

This process creates the necessary directory structure and installs Python dependencies in a virtual environment. If you use **linux** or **macOS** with :

```
ls -al
```

will show a new **.venv** file wich containes all dependencies an the version of **python** we are going to use.

4. Optional code formatting For development consistency:

```
make format
```

But if you also want to stay consistent with the all code you have to run the following command :

```
pre-commit install
```

Like that for the next commits that you'll do everything will be checked before you submit it.

4.4 System execution

4.4.1 Running the application

The system can be launched using either automated **make** commands or direct **Python** execution:

```
make run
```

or manually:

```
.venv/bin/python main.py
```

Obviously you can add some CLI commands and if you are into the current **env** you can simply run :

```
python main.py
```

If you want to know all the commands that can be used try the help CLI argument.

```
python main.py --help
```

4.4.2 Testing Framework

The project includes comprehensive testing capabilities:

```
make test
```

For coverage analysis with HTML reporting:

```
make coverage
```

The coverage report can be viewed by opening `htmlcov/index.html` in a web browser. if you use firefox you can use :

```
firefox htmlcov/index.html
```

4.5 Data flow and processing pipeline

The system processes robot perception data through a structured pipeline:

- Environmental/Weather context extraction from sensor readings
- Path analysis and decision point identification
- Prompt template generation with structured information
- LLM query processing and response generation
- Natural language explanation delivery to users

4.6 Project structure and organization

The framework follows a modular architecture with clear separation of concerns:

```
project/
|-- data/                    # Example path scenarios
|   |-- documents/           # Points robots need to handle - Markdown
|   |-- explanations/        # Input-output guidance - Markdown
|   |-- facts/               # New facts for defined paths - JSON
|   +-- paths/               # Available path definitions - JSON
|-- doc/                     # Documentation
|-- evaluation/              # Evaluation scripts and results
|-- log/                     # Conversation logs
|   +-- conversations/       # Individual conversation files - Markdown
|-- src/                     # Source code modules
|-- tests/                   # Unit tests and test data
|-- LICENSE                  # Project license
|-- main.py                  # Main application entry point
|-- Makefile                 # Build automation commands
|-- pyproject.toml           # Environment configuration for pre-commit
|-- README.md                # Local setup instructions
+-- requirements.txt         # Python dependencies
```

4.7 Usage workflow

The typical user interaction follows this pattern:

- The system displays current robot path and environmental context
- Users can ask multiple questions about path decisions and conditions

- The LLM processes queries and generates contextual explanations
- Sessions are terminated with `exit` or `quit` commands
- Conversation logs are automatically saved in Markdown format in the `log/conversation` directory

4.8 Configuration and customization

The system supports various customization options:

- **Path scenarios :** Edit `data/paths.json` to add or modify navigation scenarios
- **LLM models :** The model name can be changed in the source code for different LLM variants. We can also change the temperature and all the settings of the used llm.
- **Logging :** Conversation logs are automatically generated and stored for analysis

4.9 Technical requirements

The framework is designed with several technical requirements in mind. It must support real-time processing to enable interactive use and immediate responses to user queries. The architecture should remain modular to facilitate adaptation across different robot platforms and allow for future extensions. Scalability is also essential, ensuring that the system can handle a variety of explanation types and increasing complexity as needed.

4.10 Troubleshooting and common issues

When deploying the system, several common issues may arise. First, it is important to ensure that the Ollama server is running by verifying that the `ollama serve` command is active and that the required model has been properly pulled. Port conflicts can also occur, as only one Ollama server instance should be running on port 11434 at any given time. Additionally, the Python environment must be correctly set up, make sure that the virtual environment (`.venv`) is properly activated before running any Python scripts. Finally, confirm that the necessary large language model (LLM) is downloaded and accessible to avoid runtime errors related to model availability. Please also don't forget to include the paths that you are using.

5 LLM Customization

5.1 Model selection and comparison

The project employed multiple LLM architectures to determine which one provided the most accurate answers while also offering the most comprehensive explanations. The LLM used during this project are the next ones :

Model	**Size**	**Performance**
Llama 3.2	2.0 GB	Baseline performance
nous-hermes2:latest	6.1 GB	small model that explain lightly
deepseek	8.1 GB	Good reasoning, efficient and clear
qwen3:30b-a3b	18 GB	big model that provides good answers but take some time

Table 1: LLM model comparison

5.2 Prompt engineering

Effective prompt design proved crucial for generating relevant explanations. The system uses structured prompts that include:

- Environmental context and sensor readings
- Historical path information
- User questions and interaction history
- Domain-specific constraints and objectives
- Additional datas provided by the human or captors
- Typical question/answer results

5.3 Fine-tuning approach

The customization process involved:

- Dataset creation with robot-specific scenarios
- Prompt template
- Response quality evaluation and iteration
- Integration with robot perception systems

5.4 Customization

The customizatino of the ollama model was done using firstly multiple models to compare the performance of each one. Then with the model we do some fine-tuning, like giving to the llm some background information about the robot, the environment and the sensors. But also how does the robot must answer to the user question, like giving a short answer or a long one. There is also points about which points it must consider when answering and take in consideration will answering.

6 Human-Robot Interaction Prototype

6.1 System implementation

The prototype system was implemented using **Python** with the following key components:

- **Path processing module :** Handles environmental data and context extraction.
- **LLM interface :** Manages communication with local Ollama server.
- **Conversation manager :** Maintains interaction history and context.
- **User interface :** Provides command-line and potential interaction.
- **Storing conversations :** Store all the previous conversations but also the new datas provided during the conversation.

6.2 Core features

The implemented system supports:

- Interactive questioning about robot path decisions.
- Real-time explanation generation.
- Context-aware responses based on environmental conditions.
- Conversation logging and history management.
- Multiple scenario support for testing and evaluation.

6.3 Technical architecture

The system architecture follows a modular design:

- `src/robotPathExplanation.py`: Main application logic
- `src/core/path.py`: Data structures for path and environmental information
- `src/llm/llmModel.py`: LLM integration and prompt management
- `src/logging/conversationLogger.py`: Interaction recording and analysis

6.4 Usage scenarios

The prototype supports various interaction scenarios:

- Path selection queries (e.g., "Which path should I take if I want the easiest route?")
- Safety-related questions (e.g., "I have a heavy load. Which path is safest?")
- Time-constrained decisions (e.g., "I am in a hurry but want to avoid danger")
- Real-time updated decision (e.g., "Which path should i took according to the new datas that I provided ?")

7 Evaluation

7.1 Evaluation Methodology

Our evaluation employed a multi-pronged strategy to assess both technical performance and user experience. We used four main methodologies: automated testing, explanation quality assessment, user studies, and performance metrics analysis. Automated unit tests, integrated into our CI pipeline, ensured reliability across components such as input processing, reasoning algorithms, and output generation. For explanation quality, we combined keyword matching and semantic similarity metrics to evaluate clarity, relevance, and accuracy. User studies engaged participants in realistic tasks, providing feedback on usability and comprehension. Performance evaluation focused on response time, answer accuracy, and user satisfaction, prioritizing semantic correctness over speed.

7.2 Quantitative Results

We used keyword coverage (targeting $\geq$75%) as a primary metric for explanation completeness. Factual accuracy was validated through expert review and knowledge base cross-checks, yielding high correctness rates across query types. We also evaluated consistency by testing the system with semantically similar queries. Results showed stable performance, though some variability suggested potential for improvement through additional fine-tuning.

7.3 Qualitative Findings

Users valued clear, jargon-free explanations in natural language and appreciated responses tailored to context. Context-aware answers enhanced understanding and reduced cognitive load. Complex scenarios sometimes led to less focused responses, while simple cases produced consistently clear outputs. Interactive capabilities, such as follow-up questions, increased user trust and engagement.

7.4 Limitations and Challenges

Key limitations included high computational costs for real-time explanations and occasional over-reliance on rigid templates. Context retention weakened in long conversations, affecting coherence. Finally, domain-specific fine-tuning remains essential for high performance in specialized fields, as general models lacked sufficient adaptability to expert terminology and expectations.

8 Conclusion

8.1 Summary of contributions

This project successfully demonstrated the feasibility of using Large Language Models to enhance human-robot communication through natural language explanations of robotic decisions. Key contributions include:

- A novel integration framework for LLMs in robotic explanation systems
- A working prototype that translates low-level perception data into natural language (theoradicaly)
- Empirical evaluation demonstrating improved user understanding and engagement
- Open-source implementation available for further research and development

8.2 Impact and implications

The work addresses a critical gap in human-robot interaction by making robotic decision-making more transparent and interpretable. This has implications for:

- Increased user trust in autonomous systems
- Enhanced collaboration in human-robot teams
- Improved debugging and system maintenance
- Better user training and system adoption

8.3 Future work

Several directions for future research emerge from this work:

- **Real-time integration :** Developing more efficient processing pipelines for live robot systems.
- **Multimodal explanations :** Incorporating visual and gestural explanation modalities.
- **Personalized explanations :** Adapting explanation style to individual user preferences.
- **Domain expansion :** Extending beyond path planning to other robotic decision domains.
- **Fine-tuning optimization :** We can develop robot-specific language models to improve performance, or we can directly train a robot to perform the exact task we are searching for.

8.4 Final remarks

The integration of Large Language Models with robotic systems represents a promising approach to bridging the communication gap between humans and machines. As LLM technology continues to advance, we can expect even more sophisticated and natural human-robot interactions, ultimately leading to more effective and trustworthy autonomous systems. The open-source nature of this implementation encourages further research and development in this important area of human-robot interaction.

References

[1] F. Cruz, R. Dazeley, P. Vamplew, and I. Moreira. Explainable robotic systems in reinforcement learning scenarios. 2020.

[2] M. Jirotka. Ethical black box for robots, 2024. Wikipedia entry on Marina Jirotka's work on accountability in robotics.

[3] M. M. Reimann, F. A. Kunneman, C. Oertel, and K. V. Hindriks. A survey of dialogue management in human-robot interaction. 2023.

[4] L. Sanneman and J. A. Shah. Effects of robot explanations on human perceptions. 2020.

[5] Daisy Zhe Wang et al. Trust calibration in explainable robots. 2020.

[6] Wikipedia contributors. Human–robot interaction, 2024. Wikipedia article.

[7] Wikipedia contributors. Subsumption architecture, 2024. Wikipedia article.