POLITECNICO DI TORINO

MASTER's Degree in ENGINEERING AND MANAGEMENT



MASTER's Degree Thesis

Optimising Retrieval-Augmented Generation for Industrial Collaborative Robot Programming

Supervisor

Professor Giulia Bruno

Co-Supervisor

Enrico Botticella

Candidate

Emaan Khan

JULY 2026

Abstract

Industry 5.0 is prompting the shift towards human-cyber-physical systems, where human operators and industrial collaborative robots (cobots) can work in synergy whilst using AI as a teammate instead of merely a tool. While Large Language Models (LLMs) offer intuitive, conversational interfaces for troubleshooting and cobot programming, their deployment in manufacturing environments is hindered by the risk of AI hallucinations and a lack of proprietary domain-related knowledge. The thesis presents an optimised Retrieval-Augmented Generation (RAG) framework designed specifically to help operators in programming and troubleshooting industrial cobots. The specific case of the YASKAWA cobot in the Politecnico di Torino Mind4LAB is considered, where its complex, unstructured and detailed manuals are utilised. The research addresses the critical bottlenecks of data ingestion and retrieval using AI. The work uses sophisticated parsing techniques, such as natural segmentation and multimodal extraction, to get around the drawbacks of conventional text extraction. In the end, the suggested localised AI architecture closes the gap between complex robotic systems and human operators. Furthermore, the research demonstrates the optimisation through parsing and pre-indexing pipelines within a RAG framework helps reduce LLM hallucinations, builds operator trust, and enables safe human-robot collaboration on the modern shop floor.

I dedicate this to all of my friends and family, especially to my husband, Hamza, my parents, Mama and Baba, my siblings, Incya and Ishmail, and finally to our little one, without whom this would not have been possible.

Acknowledgements

I would like to extend my deepest gratitude towards my thesis supervisor, Prof. Giulia Bruno for her guidance and patience.

I would like to further thank my co-supervisor Enrico Botticella for all of his help, advice, and guidance in this thesis.

Index

List of Tables VI

Acronyms VIII

Introduction 1

1 Theoretical Background 3
- 1.1 Industry 4.0, 5.0, Human Cyber-Physical System, and Collaborative Robots (Cobots) . . . 3
- 1.2 AI Chatbots, LLMs, and Cognitive Automation . . . 4
- 1.3 Retrieval-Augmented Generation (RAG) . . . 5
  - 1.3.1 RAG, Text Parsing, and Indexing . . . 6
  - 1.3.2 Indexing, Knowledge Graphs, and Database Construction . . . 7

2 State-of-the-Art 8
- 2.1 Literature Review Methodology . . . 8
- 2.2 Conversational AI and Human in the Loop Robot Training . . . 12
- 2.3 Advanced RAG Architectures for Industrial Applications . . . 13
- 2.4 AI Driven Code Generation and TRIZ Optimisation . . . 14
- 2.5 Autonomous Optimisation and Continuous Learning . . . 15
- 2.6 Thesis Objective . . . 16

3 Methodology 17
- 3.1 YASKAWA in the Mind4LAB . . . 17
- 3.2 Software Architecture . . . 20
  - 3.2.1 PrivateGPT . . . 20
  - 3.2.2 Dots.mocr . . . 21
- 3.3 Use Cases . . . 22
- 3.4 Evaluation of framework . . . 24
  - 3.4.1 Human-Recorded Metrics . . . 24
  - 3.4.2 RAGChecker Metrics . . . 25

3.4.3 RAGChecker Formulary 26

4 Results/ Case study 31
4.1 Results 31
4.1.1 Novice User Efficiency 31
4.1.2 System Quality 36
4.1.3 Overall Metrics of PrivateGPT 40
4.1.4 Retriever Metrics 43
4.1.5 Synthesis/ Generator Metrics 45
4.2 Discussion of the Results 50

5 Investigation for Future Works 53
5.1 Document Parsing 53
5.2 Parsing and Testing the Manuals 54

Conclusion 57

A Use Case Tasks and Queries 59

Bibliography 64

List of Tables

2.1 Literature Review 12

3.1 Use Case Description 23

5.1 Percetage Difference of Results in PDF vs. Markdown Files 55

A.1 Use Case 1 59
A.2 Use Case 2 60
A.3 Use Case 3 61
A.4 Use Case 4 62
A.5 Use Case 6 63

Acronyms

AI
: artificial intelligence

LLM
: Large Language Models

RAG
: Retrieval-Augmented Generation

Introduction

The manufacturing sector is undergoing a paradigm shift from the automation-driven Industry 4.0 to the human-centric Industry 5.0. In this new era, the use of collaborative robots (cobots) is no longer seen as a separate tool; they are considered teammates who work alongside human operators. However, programming and troubleshooting cobots, such as the YASKAWA Motoman YRC 1000 system deployed in the Mind4LAB at Politecnico di Torino, has historically required highly specialised knowledge, such as the YASKAWA Inform Language. This creates a cognitive barrier for novice operators on the shop floor.

While LLMs offer a natural language interface that can potentially translate human commands into robotic ones, their deployment is severely limited by the risks of factual hallucinations and a lack of domain-specific proprietary knowledge, which also poses a hazard to the user. To bridge this gap, Retrieval-Augmented Generation (RAG) has emerged as an imperative technique for grounding LLMs in verified technical manuals.

The thesis aims to implement, analyse, and optimise a localised RAG framework, PrivateGPT, to help novice users program and utilise the collaborative robot YASKAWA without having to sift through an abundance of manuals. Furthermore, by using raw technical manuals as a local database, the research aims to:

- Promote self-reliance and reduce dependencies on external expert assistance;
- Establish a rigorous dynamic framework using both human-recorded usability metrics (interaction turns, latency, etc) and RAG metrics performed through the RAGChecker;
- Assess the impact of advanced document parsing and chunking pipelines on RAG performance.

Chapter 1 discusses the Theoretical Background, providing the underlying knowledge necessary to understand the thesis's content.

Chapter 2 presents the state of the art, highlights the literature review, and discusses the content of the papers referred to in the thesis.

Chapter 3 begins to elaborate on the specifics of the thesis work with the work that was done in the Mind4LAB at Politecnico di Torino with the YASKAWA robot through the use of PrivateGPT. Specific Use Cases were utilised to query PrivateGPT and program the robot based on the results provided. This was done until satisfying results were reached for all of the tasks within the Use Cases. The Use Cases were of varying natures, ranging from basic motion programming and pick-and-place of objects through different physical modular attachments on the YASKAWA robot to the robot's programming and functionality. Furthermore, the results are analysed through a RAGChecker to evaluate the performance. Additionally, instead of simply having PDF manuals, the PrivateGPT is designed to ingest manuals in markdown format, versions of the file that are easier for LLMs to parse and extract information from.

Chapter 4 elaborates on the results from the work performed in the previous Chapter, discusses the results from the RAG Checker, and analyses the reasoning behind each.

Lastly, Chapter 5 discusses the possibility and further work on the parsing and pre-indexing of documents to allow for higher RAG Checker metrics through refinement of the original documents.

Chapter 1

Theoretical Background

The purpose of this chapter is to highlight the foundational concepts that lead to a better understanding of the chapters ahead. With inventions such as Artificial Intelligence (AI), the thesis will explore how industries have shifted to a human-approach, alongside the usage of collaborative robots, Large-Language Models (LLMs) and Retrieval-Augmented Generation (RAG). With the introduction of these tools, exploring the know-how of these digital resources becomes even more important.

1.1 Industry 4.0, 5.0, Human Cyber-Physical System, and Collaborative Robots (Cobots)

Modern industries are now moving from the age of Industry 4.0, the stage of digitisation, to Industry 5.0, which encompasses Cyber-Physical cognitive systems, Green Manufacturing, Cultural Collaboration, and Mass Customisation [1]. This allows a more human-centric focus of Industry 5.0. Industry 5.0 emphasises sustainability, resilience, and human-centrality, moving away from systems designed solely to improve efficiency toward Cyber-Physical-Human Production Systems (CPHPS). In these environments, technologies are deployed to augment human capabilities rather than replace them. A critical theoretical model in this space is "Sliding Work Sharing," which involves dynamically allocating tasks between human operators and AI agents based on real-time contextual conditions, user workload, and the specific capabilities of each agent [2][3][4][5][6].

Unlike traditional robots, collaborative robots, such as the YASKAWA robot, also add a feature that, as seen later, enables humans to work alongside it safely in shared workspaces. They are extremely adaptable and can be programmed intuitively using methods such as lead-through programming [7], in which a human operator physically guides a robot arm along a desired trajectory. This is highly

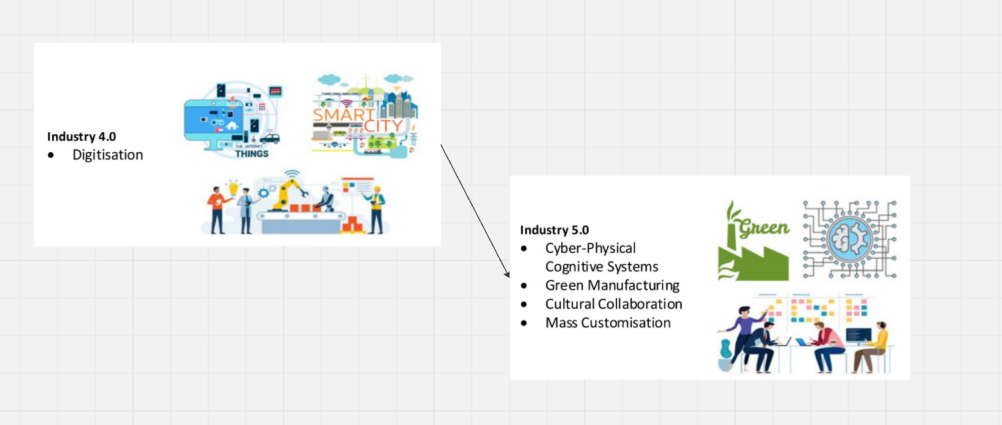


Figure 1.1: Industry 4.0 to 5.0 [19]

effective for spatial-pathing. With Industry 5.0, there is a shift towards robots that work alongside humans. Specifically, Artificial Intelligence has also enhanced the possibility of Human-Robot Collaboration [5].

1.2 AI Chatbots, LLMs, and Cognitive Automation

The collaborative robots, though extremely effective, struggle with complex logical conditions or adapting to variations on the shop floor. Hence, the introduction of cognitive software interfaces arises. With advances in AI, the need for its integration is becoming increasingly apparent. AI can now be seen as a "team member" rather than a simple tool [8]. This can be furthered by the use of the AI chatbot to improve the functioning of the robot, as will be seen in this thesis [4][9][10].

Achieving process-level autonomy requires a flexible cognitive core capable of understanding high-level human intent and translating it into machine execution. Large Language Models have taken rise to fill this gap, hence providing the possibility of enabling AI chatbots to act as intermediaries between human operators and cobots. In the "Large and Small Model" Collaboration paradigm [9], the LLM chatbot can serve as a high-level planner and communicator while small models, meaning the deterministic control algorithms, handle the precise, low-level execution of robotic hardware.

However, the issue arises with ready-to-use LLMs, as they are not designed for knowledge-intensive industrial domains and precise mathematical computation, and are therefore prone to "hallucinations". They can therefore generate factually incorrect or unsafe robotic commands [4][5][11].

In specialised fields, such as manufacturing, finance and medicine, where precision

and reliability are imperative, the inability of an LLM to verify its own outputs presents a severe operational risk [12]. Hence, Industrial environments demand strict adherence to proprietary safety protocols, specific hardware limitations, and precise coordinate geometries. General-purpose LLMs lack this intrinsic spatial and mechanical awareness because they are trained on vast, generalised internet datasets rather than deterministic control parameters [13].

Consequently, when a basic LLM attempts to generate operational commands without domain-specific anchoring, it tends to hallucinate plausible-sounding but mechanically impossible trajectories. While the "Large and Small Model" architecture effectively delegates physical actuation to deterministic algorithms, the "Large" cognitive model still acts as the primary decision-maker. If a chatbot misunderstands the operator's query or hallucination occurs, the physical execution can still result in hazardous conditions [9]. This highlights a critical bottleneck in deploying conversational AI on the shop floor: a mechanism must be established to empower the chatbot to accurately comprehend complex, proprietary robotic manuals and execute verified reasoning, bridging the gap between natural language flexibility and industrial-grade reliability [2].

1.3 Retrieval-Augmented Generation (RAG)

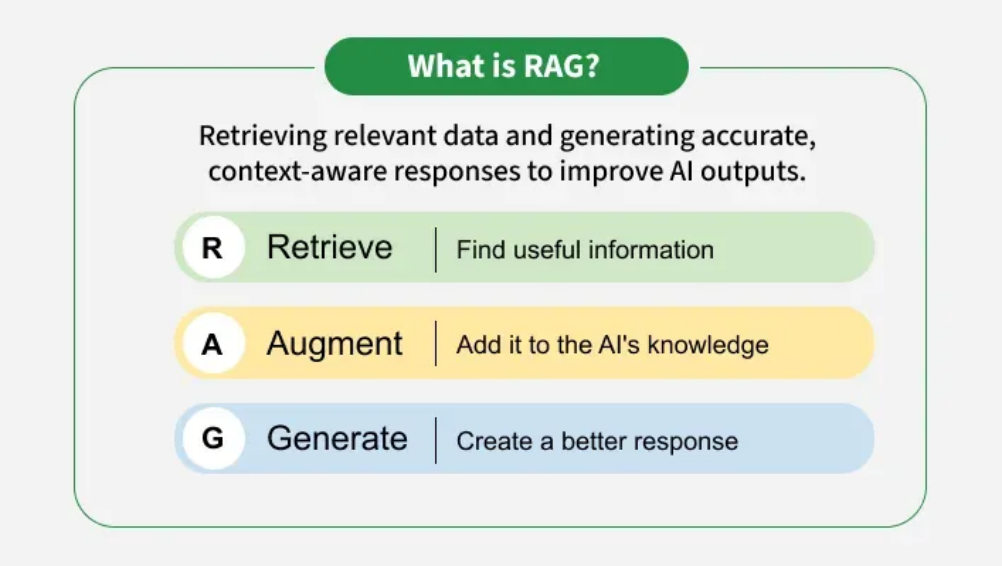


Figure 1.2: What is RAG? [14]

To mitigate the risk of the aforementioned hallucinations and to make AI chatbots viable for industrial training, Retrieval-Augmented Generation (RAG)

utilises an external knowledge retrieval mechanism in conversational AI. When a user communicates and questions the chatbot to train the cobot, the RAG system first and foremost queries a localised database, for example, the robot's manuals. The retrieved data provides a grounded context for the chatbot to generate evidence-based code or training parameters, depending on the query and situation. Furthermore, because the manufacturing data is highly sensitive in certain situations, industrial deployments rely on FedRAG (Federated RAG) and localised edge-computing to ensure cybersecurity and to keep the proprietary data on the shop floor while maintaining low-latency communication between the chatbot and the robot [15][9][16][17].

1.3.1 RAG, Text Parsing, and Indexing

Industry 5.0 drives innovation towards Human Cyber-Physical Systems (HCPS) [6]. With this, the transition from technology as a tool to Human AI Teaming (HAIT) becomes evident. Now, AI is a valuable team member [8]. To work as an efficient team member, AI models require a grounded, accurate understanding of their physical and operational environments.

Large Language Models (LLMs) serve as the cognitive core of these interactive systems by translating natural human language into executable Python code or commands. However, LLMs by themselves are prone to hallucinations and lack the knowledge for accurate and precise execution. For this reason, RAG is used to mitigate this by integrating external, domain-specific knowledge bases into the AI's reasoning process [15][13].

Before the information can be retrieved by the AI from complex documentation, such as the YASKAWA robot operation manuals, as will be seen in later chapters, these documents must undergo rigorous text parsing[18]. Parsing involves extracting unstructured text from formats such as PDFs and converting it into clean, structured data [19]. Recent literature highlights that modern parsing must often handle multimodal and "imperfect contexts". Oftentimes, the tool utilised strips away the irrelevant formatting and isolates raw text, data, and tables. Just as researchers use LLMs to extract comprehensive, sequential processing workflows from unstructured bioinformatics publications, industrial parsing must successfully reconstruct step-by-step robotic programming protocols [20]. As LLMs have limited context windows, the parsed text must be broken down into smaller chunks. Advanced implementations use "natural segmentation", which leverages document heading symbols to split text into parent blocks and child sentences, preserving the semantic continuity needed to understand multi-step operations [21]. Figure 5.1 highlights how the parsing and indexing work so that the data is processed in a more efficient manner.

1.3.2 Indexing, Knowledge Graphs, and Database Construction

Following parsing, the text chunks must be pre-indexed to create a searchable database. To maximise retrieval accuracy in knowledge-intensive and highly regulated tasks, such as navigating shop-floor safety protocols or dynamic customs clearance regulations, state-of-the-art systems employ a hybrid RAG approach [21]. The architecture combines vector similarity searches with term-frequency keyword extraction algorithms, such as BM25 and LLM-constructed knowledge graphs [19]. By structuring parsed data into knowledge graphs, the system can establish explicit logical relations among robotic components, errors, and safety limits. This allows the AI to help resolve ambiguity in user queries, actively quantify risks, and instantly locate specific troubleshooting syntax to produce evidence-backed responses [19].

Chapter 2

State-of-the-Art

2.1 Literature Review Methodology

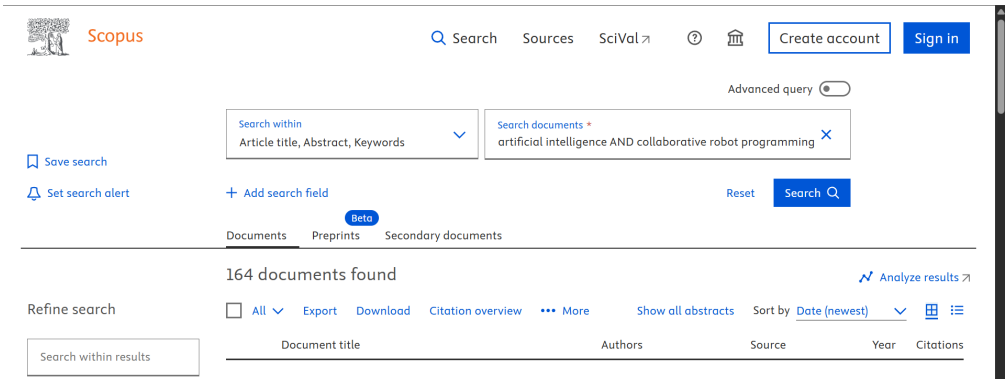


Figure 2.1: Scopus Results

For an in-depth understanding of the topic, a literature review was conducted.

The first search on SCOPUS (an abstract and citation database) yielded a result of 165 documents when "artificial intelligence AND collaborative robot programming" was searched.

The initial search on IEEE Xplore returned 254 results; however, selecting journals, magazines, books, and early access articles reduced the count to 175 for "artificial intelligence AND collaborative robot programming".

On SCOPUS, searching for "RAG" AND "collaborative robots" returned 13 papers; adding "AI" to the search bar returned only 8.

Furthermore, the search "Parsing" AND "LLM" AND "INDEX" led to the retrieval of 19 papers. Of these, only 5 were deemed relevant.

On IEEE Xplor, the combination of "collaborative robots", "AI", and "RAG"

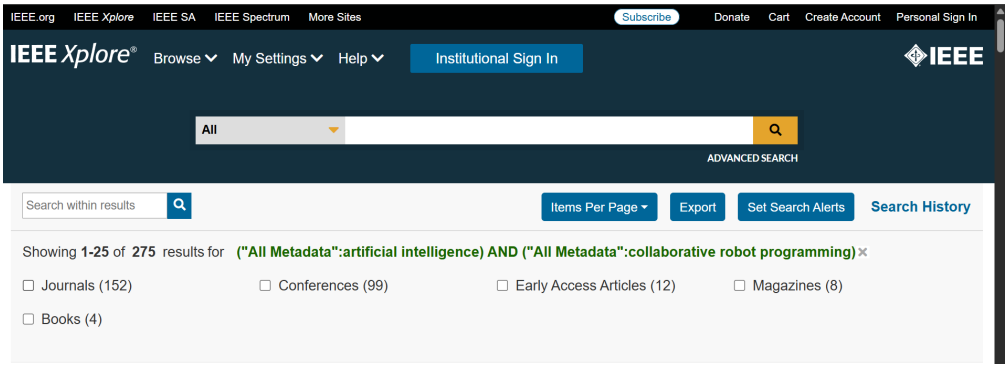


Figure 2.2: IEEE Xplore Initial Results

yields 3 papers. (("RAG" OR "Retrieval Augmented Generation" OR "Retrieval Augmented-Generation") AND ("AI" OR "Artificial Intelligence" OR "Collaborative Robot" OR "Collaborative Robot Programming")) was used to improve the search criteria.

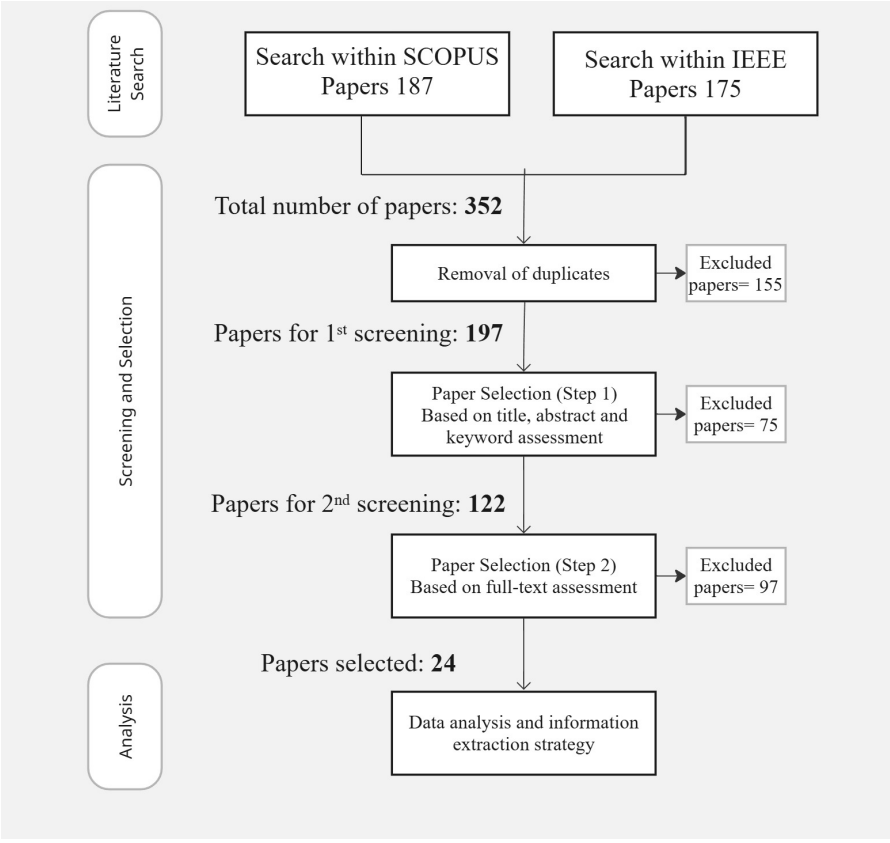


Figure 2.3: Literature Review Methodology

After removing duplicates and refining the search, only 197 papers remained. After the 1^{st} screening based solely on title, abstract, and keyword assessments, 122 papers remained. A thorough reading of the texts in order to perform the 2^{nd} screening resulted in 24 papers being used in total. However, 2 additional papers were retrieved using the RAG and Multimodal OCR frameworks, bringing the total to 26 papers.

No.	Author	Title	Key Concept
[1]	Adel A. et al, 2022	Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas	Human-centric AI chatbots
[2]	Fazlollahtabar, H. et al, 2025	Human-robot interaction using retrieval-augmented generation and fine-tuning with transformer neural networks in industry 5.0	Regret-based learning & error KPIs
[3]	Karapanagiotis, P. et al, 2025	Enabling interoperable human–AI teaming for automation in construction and manufacturing via Digital Twins and Sliding Work Sharing ontologies	Sliding Work Sharing
[4]	Miao H. et al., 2025	Modelling and Robust Control for Power Insurance Assessment SCARA Robots Using RAG Retrieval-Augmented Techniques	RAG retrieval-augmented techniques
[5]	Rrapi F. et al, 2026	From AI foundations to Large Language Models: A survey on challenges and opportunities in collaborative robotics	LLMs as cognitive supervisors
[6]	Lou, S. et al, 2024	A human-cyber-physical system enabled sequential disassembly planning approach for a human-robot collaboration cell in Industry 5.0	Human-cyber-physical system
[7]	Arjomandi, M. et al., 2026	A review of applications of collaborative robot in welding and additive manufacturing	Cobot safety and lead through programming
[8]	Berretta S. et al, 2023	Defining human-AI teaming the human-centred way: a scoping review and network analysis	AI as a teammate
[9]	Qunlong C. et al, 2026	Leveraging large and small model collaboration for advanced automation in smart manufacturing	Large and small model collaboration

Continued on next page

No.	Author	Title	Key Concept
[10]	Zhou T. et al, 2026	Towards intelligent knowledge assistance in abrasive belt grinding via a retrieval-augmented generation chatbot with reliability support	RAG chatbot
[11]	Haitao L. et al, 2026	Mutual information-based retrieval-augmented generation for domain question answering	Mutual information-based RAG
[12]	Ru D. et al, 2024	RAGCHECKER: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation	RAG Checker reference paper
[13]	Wang T. et al, 2024	An LLM-based vision and language cobot navigation approach for Human-centric Smart Manufacturing	Operator Chat to Robot Code
[15]	Shi, T. et al., 2026	Knowledge-driven autonomous materials research via collaborative multi-agent and robotic system	Multi-agent autonomous execution
[16]	Zhou, X. et al, 2026	LLM-based domain-specific knowledge graph construction and hybrid retrieval-augmented generation	Intent-constrained hybrid retrieval algorithm
[17]	He, H. et al, 2026	Federated Retrieval-Augmented Generation for Cybersecurity in Resource-Constrained IoT and Edge Environments: A Deployment-Oriented Scoping Review	FedRAG & cybersecurity
[18]	Chen K. et al, 2024	An Automatic Video Tag Extraction Method based on Large Language Model Text Content Parsing	Multimodal Parsing using OCR models
[19]	Zhenyang G. et al, 2025	LLMQua:Using LLM automation to enhance risk quantification in data processing for big data platforms	Parsed text into knowledge graphs for structured pre-indexing
[20]	Wang Y. et al, 2025	BioWorkflow: Retrieving comprehensive bioinformatics workflows from publications	Sequential Parsing
[21]	Rong H. et al, 2025	ICCA-RAG: Intelligent Customs Clearance Assistant Using Retrieval-Augmented Generation (RAG)	Chunking & Hybrid Indexing
[22]	Salunkhe, O. et al., 2026	Developing Code Agents for Robot Programming: Technical and Managerial Perspectives	AI Code Agents & Managerial Shifts

Continued on next page

No.	Author	Title	Key Concept
[23]	Balog, B. et al., 2025	AI-Driven Robot Programming Optimization Using TRIZ Principles for Efficient and Seamless Deployment	RAG & TRIZ optimisation
[24]	Bassiouny, A. et al, 2025	UJI-Butler: A Symbolic/Non-symbolic Robotic System that Learns Through Multi-modal Interaction	Human in the loop learning
[25]	Tharmilan T. et al, 2025	Enabling Conversational Intelligence in Service Robots Through Multi-modal Fusion and Resource-Aware Generative Models	Resource-aware generative models
[26]	Choudhary K. et al, 2025	A Retrieval-Augmented Generation (RAG)-Based LLM for Modern Warehouse Automation and Management	Autonomous robots in warehouses
[27]	Zheng H. et al, 2026	Multimodal OCR: Parse Anything from Documents	Multimodal OCR (dots.mocr)

Table 2.1: Literature Review

2.2 Conversational AI and Human in the Loop Robot Training

Recent literature highlights a rapid shift from rigid, vendor-specific scripting toward natural language programming. A prominent example is the UJI-Butler framework that learns through multimodal interaction [24]. This allows operators to train the robot using a combination of physical lead-through guidance and conversational feedback, establishing a robust human-in-the-loop that ensures safety and captures operator knowledge [7][23][3][4]. Furthermore, integrating conversational intelligence requires resource-aware generative models, which often strain the computational limit of onboard edge devices.

To address this computational bottleneck in edge environments, recent architectures propose a collaborative paradigm between large and small AI models. In these hybrid frameworks, computationally heavy Large Language Models (LLMs) operate via secure servers to interpret ambiguous human conversational intent and orchestrate high-level reasoning. Simultaneously, localised, lightweight "small models" handle the deterministic, latency-sensitive physical execution directly on the shop floor [9][13].

This distributed approach is particularly critical because continuous human-in-the-loop training relies heavily on sensitive proprietary manufacturing data. To maintain operational security and mitigate edge-device constraints, facilities increasingly deploy Federated Retrieval-Augmented Generation (FedRAG) systems

in resource-constrained IoT environments [17]. These systems allow robots to securely process parsed technical manuals and real-time conversational feedback at the edge, eliminating the need to transmit proprietary intellectual property to public cloud APIs.

Ultimately, this conversational interface acts as a vital bridge for Human-AI Teaming (HAIT), shifting the dynamic from simple machine operation to true collaborative problem-solving. When augmented with vision and language navigation capabilities, conversational AI enables collaborative robots (cobots) to contextually ground natural language instructions within their physical workspace. By integrating RAG and fine-tuned transformer networks, the cobot can retrieve domain-specific knowledge—such as diagnostic protocols from manuals to actively assist the operator during training or troubleshooting. This conversational closed-loop system not only mitigates the hallucinations typical of foundation AI models but also empowers a human-centric Industry 5.0 environment in which the robot adapts to the operator's instructions in real time [2][13].

2.3 Advanced RAG Architectures for Industrial Applications

A primary challenge in utilising chatbots for robotic training is the risk of retrieving noisy or irrelevant data. This can then lead to faulty code generation and other errors. To ensure the AI chatbot issues safe, appropriate commands, researchers have developed RAG architectures.

Intent-Constrained Hybrid RAG (IC-HRAG), as referenced by Zhou et al, introduced a system that builds an ontology-enhanced knowledge graph (KG) from industrial manuals. By recognising the user's explicit intent within the chat interface, the system constrains the retrieval search space, significantly improving diagnostic precision [16].

Mutual Information-Based RAG (MI-RAG), as highlighted in [11] by Lu et al., proposed a strategy utilising mutual information to accurately locate highly relevant knowledge subgraphs, effectively filtering out irrelevant noise that could mislead the LLM during conversational interactions.

Localised Conflict Filtering: In the domain of abrasive belt grinding, researchers developed a locally deployable RAG chatbot that utilises semantic conflict filtering prior to response generation. This proves that lightweight, localised chatbot models can achieve near-parity with massive API-based models while ensuring the deterministic reliability required for machine operation [10].

2.4 AI Driven Code Generation and TRIZ Optimisation

When an operator uses a chatbot to train a robot, the AI must instantly generate underlying operational code [23]. To optimise this systematically, the AI-generated code is now being merged via RAG with structured engineering methodologies such as TRIZ (Theory of Inventive Problem Solving). Balog et al. developed a framework that combines RAG with TRIZ conflict resolution principles to automatically generate and refactor robot programming code. The system can balance contradictory requirements, such as execution speed and spatial precision. This ensures that the chatbot generates scripts compliant with company standards while significantly reducing programming time [23]. Furthermore, integrating Large Language Models (LLMs) into the code generation pipeline requires strict syntactic adherence to vendor-specific programming languages [7]. While off-the-shelf LLMs possess general coding capabilities, they are highly prone to hallucinating robotic trajectories or referencing non-existent API commands [15]. By utilising a localised Retrieval-Augmented Generation (RAG) architecture, the AI cross-references the operator's natural language prompt against an indexed database of verified, proprietary robotic manuals and standardised codebase templates [15]. This mechanism grounds the generated scripts in factual documentation, ensuring the resulting code is not only logically sound but instantly executable by the specific robot controller on the shop floor.

The application of TRIZ within this RAG pipeline acts as a sophisticated cognitive filter. When programming collaborative robots (cobots), operators frequently encounter physical and operational constraints [23]. For example, maximising a robot's payload capacity often compromises its acceleration, while enhancing joint mobility may restrict structural stability. By indexing the classical inventive principles of TRIZ within a structured Knowledge Graph, the AI can actively identify these bottlenecks during the code generation phase. Instead of returning an error or halting the process, the chatbot autonomously queries the TRIZ matrix, retrieves the appropriate conflict-resolution protocol, and dynamically refactors the code to circumvent the limitation [23].

This autonomous optimisation loop fundamentally redefines Human-AI Teaming (HAIT) within Industry 5.0. As the AI assumes the burden of syntax formatting, error troubleshooting, and structural code refactoring, the human operator is freed from the cognitive fatigue of manual programming [7]. The operator's primary responsibility shifts to validating the AI's proposed solutions—often by testing the refactored code in a digital twin or simulation environment—before safely deploying the script to the physical cobot [15]. The managerial implications of such AI code are profound, as this synergy creates a highly resilient, human-centric

manufacturing environment where complex robotic tasks can be adapted and programmed dynamically by non-expert users, definitively shifting the operator's role from a manual programmer to a strategic supervisor [7][15][23].

2.5 Autonomous Optimisation and Continuous Learning

Modern industrial environments require continuously evolving manufacturing systems that leverage cognitive computing and self-learning algorithms to predict and optimise production line efficiency. By teaming up AI and humans, collaborative systems have to adapt dynamically, requiring agents to update shared mental models and mutually augment one another's intelligence. To go beyond static nature, state-of-the-art frameworks aim to enable robot chatbot systems to perform autonomous optimisation and continuous knowledge acquisition [5][1][8].

The existing AI networks utilise continuous learning loops to refine their operations over time [5]. Transformer Neural Networks use regret-based learning models to calculate the performance gap between actual and optional robotic actions, therefore, systematically minimising the execution time, error rates, and the need for human corrections over continuous production cycles [2]. Through In Context Learning, feedback-driven adaptive planning enables LLMs to retrieve success cases and instantaneously autocorrect operational plans using human feedback, achieving optimisation without costly model retraining [9]. Robotic systems integrate the "human in the loop" approach to store episodes of data from physical leads through guidance and conversational feedback [24]. Self-evolving interactive agents resolve linguistic ambiguities through continuous learning from unlabelled data and dynamically adapting to open-ended user instructions in noisy environments [25].

To ensure the accuracy of AI systems, the underlying knowledge bases that feed RAG models must be updated continuously. Mutual Information RAG (MI-RAG) architectures employ active update strategies in which the system itself can extract new structured knowledge from natural-language answers, thereby continuously expanding and correcting its internal Knowledge Graph [11]. Intent-constrained subgraph expansion mechanisms allow domain-specific knowledge graphs to perpetually control the retrieval scope, minimise noise, and iteratively enhance diagnostic precision for complex industrial equipment [14]. In resource-constrained edge environments, FedRAG enables distributed networks of sensors and gateways to collaboratively tune retrieval adapters and securely update knowledge bases, ensuring continuous learning while avoiding exposure of sensitive data [17]. Local RAG chatbots deployed for specialised tasks, such as abrasive belt grinding, utilise semantic conflict filtering to continuously adapt to complex, multi-factor working conditions and dynamic process parameters [10].

At the highest level of automation, AI systems orchestrate entirely closed-loop physical optimisation across the factory floor. Multi-agent systems, such as the MARS framework, coordinate multiple specialised LLM agents and domain-specific tools to autonomously execute material synthesis, analyse real-time experimental data and iteratively optimise material properties without manual intervention [15]. Warehouse automation architectures leverage RAG-based LLMs to consistently process real-time inventory updates from Unmanned Aerial Vehicles and dynamically dispatch mobile manipulators, adapting workflows instantly to maintain supply chain efficiency [26]. Vision and Language Navigation frameworks utilise LLMs to continuously generate and execute Python code using pathfinding algorithms, enabling Automated Guided Vehicles to dynamically adapt their trajectories around static and moving obstacles [13]. Human Cyber Physical Systems utilise improved metaheuristic algorithms, such as multi-objective hybrid grey wolf optimisation (MOHGWO), to continuously handle fuzziness and randomness in dynamic task allocation and sequential disassembly planning [6]. Dynamic task sharing is further enabled by ontologies that utilise Digital Twins to continuously adjust the autonomy levels of AI actors in response to changing environmental contexts and human fatigue [3]. At the hardware level, optimal robust control algorithms applied to SCARA robots allow joints to dynamically mitigate trajectory deviations caused by unexpected external disturbances during high-precision outcomes [4]. To ensure the underlying robotic programming evolves safely alongside these dynamic systems, frameworks merge RAG with TRIZ conflict resolution principles, continuously refactoring executable code to balance conflicting requirements such as speed vs spatial precision without violating rigid safety constraints [23]. This continuous, intelligent code generation effectively shifts operator workflows from manual scripting to strategic, high-level process supervision.

2.6 Thesis Objective

As we will see in the next few chapters, the thesis aims to use an AI chatbot through the utilisation of RAG to help a novice user program and operate a collaborative robot, YASKAWA. The user will be assigned tasks and use cases as objectives to achieve. Through the chatbot, namely PrivateGPT, the user is expected to successfully complete these tasks without having to sift through hundreds of pages of manuals to find the correct programming instructions. The thesis aims to promote self-reliance by enabling users to use the robot with minimal external assistance.

Chapter 3

Methodology

The thesis aims to evaluate the importance of RAG for the use of industrial collaborative robots. It considers the use of a specific industrial cobot, YASKAWA, and how the use of an Artificial Intelligence chatbot, PrivateGPT, can improve the implementation of the cobot on the floor.

In subsequent sections, additional details are provided to better understand how the chatbot and robot are used together to improve the robot's output. It is anticipated that the use of Retrieval-Augmented Generation in the chatbot should help reduce hallucinations commonly seen in Language Language models. Furthermore, it is expected that even a novice user can use the chatbot to learn how to program and operate any robot, specifically YASKAWA in this case.

The thesis further examines how to improve AI chatbot results to better use the cobot.

3.1 YASKAWA in the Mind4LAB

YASKAWA is a collaborative robot present in the Mind4LAB at Politecnico di Torino, as seen in Fig.3.1. This robot is also used in manufacturing industries.



Figure 3.1: YASKAWA in the Mind4LAB

The robot is programmed using the MOTOMAN YRC 1000 controller, as shown in Fig.3.2.



Figure 3.2: MOTOMAN YRC 1000

The Yaskawa robot is a 6-axis collaborative robot designed not only to be programmed but also to be employed as a team member on the shop floor. The robot can move on all 3 axes (x, y, z) and rotate about 3 axes as well. It can lift loads up to 20 kg using several robot attachments, such as the OnRobot RG6 and

MG10, provided in the Mind4LAB.

The RG6 Fig.3.3 is a robotic arm gripper with a load capacity of 6 kg. The robot can be aligned with whichever object needs to be picked and, using the pincers, can pick it up once programmed by the YRC 1000 controller with specific commands.



Figure 3.3: RG6 robotic gripper

The MG10 Fig.3.4 is a multi-faceted magnetic gripper with several applications in the manufacturing, automotive and space industries for material handling, assembly and machine tending applications. Using specific commands, the user can program it to lift up to 10 kg once it is positioned correctly and brought into contact with the object.



Figure 3.4: MG10 Magnetic Gripper

3.2 Software Architecture

3.2.1 PrivateGPT

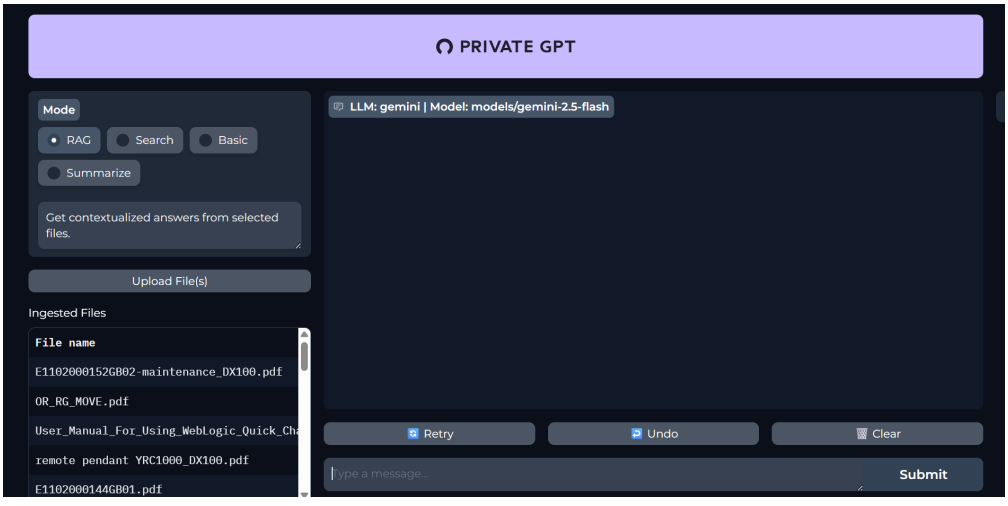


Figure 3.5: Private GPT

PrivateGPT is an AI chatbot platform, as seen in Fig. 3.5. The unedited complex manuals and PDFs are uploaded to the PrivateGPT environment, which depends on RAG. This platform is then utilised by the user to ask questions about the YASKAWA robot and to program and run it based on the results provided by PrivateGPT. The objective is to run the YASKAWA robot efficiently without having to study all the manuals extensively beforehand or require external assistance.

The configuration of the PrivateGPT is described as follows. The parser uses SemanticSplitterNodeParser; the Parser Settings are at the 98th percentile (buffer=3), the embedding model is BAAI/bge-large-en-v1.5, and the Vector DB is Qdrant. Furthermore, the retrieval model is Hybrid (Vector + BM25), the fanout is 3x, the reranking model is ms-marco-MiniLM-L-2-v2, the LLM utilised is Gemini-2.5-flash, and the top-k value is 5.

To run PrivateGPT, it is first installed on the Politecnico di Torino LEP PC and then run after adding a Google API key. PrivateGPT uses the Gemini 2.5 flash model to run smoothly.

3.2.2 Dots.mocr

Dots.mocr, a multimodal OCR, is one of the pre-processing tools available on GitHub [27] that allows parsing of text and graphics in documents into unified textual representations that are easier for an AI to read and extract data from. This tool diverges from basic OCR systems, which focus on text recognition and leave graphical regions as cropped pixels. This MOCR too: reconstructs both text and graphics, supports end-to-end training to enable the models to establish relationships between text and visual components, and converts previously discarded graphics into reusable code-level supervision, thereby enabling multimodal supervision in existing documents. The model utilises the Gemini 3 PRO in this case [27].

The next steps in the research are implementing text parsing and pre-indexing the YASKAWA manuals and PDFs to assess how the results improve.

To download dots.mocr, the paper [27] and the website is utilised is [28].

The following commands are used for installation:

```
conda create -n dots_mocr python=3.12
conda activate dots_mocr

git clone https://github.com/rednote-hilab/dots.mocr.git
cd dots.mocr

# Install pytorch, see https://pytorch.org/get-started/previous-
    versions/ for your cuda version
```

```
# pip install torch==2.7.0 torchvision==0.22.0 torchaudio==2.7.0 --index-url https://download.pytorch.org/whl/cu128
# install flash-attn==2.8.0.post2 for faster inference
pip install -e .
```

Further, for deployment, the following are applied:

```
# Launch vLLM model server
## dots.mocr
CUDA_VISIBLE_DEVICES=0 vllm serve rednote-hilab/dots.mocr --tensor-parallel-size 1 --gpu-memory-utilization 0.9 --chat-template-content-format string --trust-remote-code

## dots.mocr-svg
CUDA_VISIBLE_DEVICES=0 vllm serve rednote-hilab/dots.mocr-svg --tensor-parallel-size 1 --gpu-memory-utilization 0.9 --chat-template-content-format string --trust-remote-code

# vLLM API Demo
# See dots_mocr/model/inference.py and dots_mocr/utils/prompts.py for details on parameter and prompt settings
# that help achieve the best output quality.
## document parsing
python3 ./demo/demo_vllm.py --prompt_mode prompt_layout_all_en
## web parsing
python3 ./demo/demo_vllm.py --prompt_mode prompt_web_parsing --image_path ./assets/showcase/origin/webpage_1.png
## scene spoting
python3 ./demo/demo_vllm.py --prompt_mode prompt_scene_spotting --image_path ./assets/showcase/origin/scene_1.jpg
## image parsing with svg code
python3 ./demo/demo_vllm_svg.py --prompt_mode prompt_image_to_svg
## general qa
python3 ./demo/demo_vllm_general.py
```

The results are again run through the RAGChecker framework in order to establish how the parsing and indexing impacts the answers obtained.

3.3 Use Cases

There were 5 proposed Use Cases, each with several subsections and different focuses. Whilst some focused on basic programming, others proposed queries related to different attachments on the YASKAWA robot provided in Mind4LAB, such as the OnRobot RG6 Gripper and the OnRobot MG10 Gripper, as seen below.

Use Case 1 focuses on Basic Motion Programming of the YASKAWA Robot, such as how to create a job or delete instructions. The queries are presented in

Use Case	Objective
Use Case 1	Basic Motion Programming
Use Case 2	Basic Pick and Place by Rg6
Use Case 3	Programming and Functionality
Use Case 4	Set TCP for MG10 Manually
Use Case 6	Basic Pick and Place by MG10

Table 3.1: Use Case Description

greater detail in the appendix A to familiarise the user with the YASKAWA robot and how to program it effectively.

Use Case 2 employs the RG6 attachment, as seen in section 3.1 and Figure 3.3, and asks queries related to pick-and-place using this gripper. The queries, as seen in the appendix A, aim to focus on the programming of the robot with RG6, including its initialisation before performing any task, how to grip an object, how to move the robot to the correct position and how to place the object once the correct position has been attained. Here, the mechanism of picking and placing depends on the gripper's width: to pick up an object, the width must be programmed to match the object's width; to place the object, the width must be significantly larger than the object's width so the gripper can drop it safely.

Use Case 3 emphasises the cobot's Programming and Functionality in general. As highlighted in the appendix A, the purpose is to familiarise the user with the programming and what each command means and aims to achieve. For example, understanding what the "WHILEEXP" command means or how to deploy a specific argument to further be able to program specific functions.

Use Case 4 highlights the usage of the Tool Center Point (TCP) using the MG10 gripper, as seen in section 3.1 and Figure 3.4 as present in the appendix A. The objective of Use Case 4 is to manually configure, calibrate, and verify the Tool Center Point for the OnRobot MG10 gripper on a Yaskawa robot controller. The focus is on defining the TCP as the working reference point/tip of the tool, understanding why the TCP must be updated whenever the gripper is changed (due to changes in tool geometry and fingertip offset), understanding the physical parameters (Z-distance from the flange face) and tool load profiles (weight and centre of gravity).

Use Case 6, similar to Use Case 2, as shown in the appendix A, focuses on pick-and-place commands with the Magnetic Gripper (MG10). Use Case 6 also focuses on the initial steps to take before any other task, as outlined in the MG10 attachment. Here, instead of a pincer-like mechanism, there is a magnet, so the robot works significantly differently. To move to the object, pick up, move to the correct position, and place it, the robot now depends on magnetic force rather than

the width of the gripper. At first, the magnetic force must be greater than the object's weight (in Newtons) to lift it safely; then, to place it, the magnetic force must be reduced so the object can be positioned correctly.

As mentioned in the previous section 3.4, each Use Case has several metrics that must be completed by the novice user to assess the efficiency and usability of PrivateGPT. Some metrics are filled in by the user„ whilst others are produced by the RAGChecker, as mentioned in the Formulary in section 3.4.3.

This is done in both scenarios: either when the scenarios are uploaded directly to PrivateGPT, or when the manuals are parsed and pre-indexed through docs.mocr (as seen in section 3.2.2), then uploaded to the PrivateGPT and assessed. Both results can then be compared.

3.4 Evaluation of framework

3.4.1 Human-Recorded Metrics

Using the YASKAWA robot in the Mind4LAB, several Use Cases were provided to help analyse the resourcefulness of the PrivateGPT environment. Each query is input into the PrivateGPT space, where it either provides an answer based on what it can find in the manuals or simply states "the answer is not provided in the context". If the provided answer does not lead to a logical execution of commands on the YASKAWA robot, the query is rephrased and re-asked till either an appropriate enough answer is obtained or the query is considered "failed".

Depending on the answers, an Excel file is filled with the following metrics:

- Total Interaction Turns: how many queries were asked to solve this task;
- Theoretical Minimum Turns: how many queries were needed minimum to solve the task;
- Effective Turns (ET): how many queries were semantically correct;
- Wrong Queries (WQ): how many queries were semantically incorrect;
- Learning Rate (LR):

$$\frac{Total, Interaction, Turns}{Effective, Turns, (ET)}; \tag{3.1}$$

- Average Time [s]: the average time taken by all the responses generated to answer the query;
- Intervention: was there any intervention needed;

- Task Success Instruction: how many of Effective Turns were effective, useful or helpful to reach task completion;
- Ineffective / Interfering Instruction: how many of Effective Turns did not lead to useful commands or information;
- Refusal: how many resulted in the PrivateGPT responding with, “the answer is not provided in the context”;
- Instruction Success Rate (ISR):

$$\frac{Effective, Turn, (ET)}{Task, Success, Instruction}; \tag{3.2}$$

- Latency [s]: the time taken for the final answer to be generated in response to the query;
- State (TSR): this is either 0 or 1, depending on whether the task succeeds or fails.

3.4.2 RAGChecker Metrics

The RagChecker employs 8 diverse RAG systems to demonstrate their properties and differences. The two retrievers are BM25 and E5-Mistral. For the four generators, GPT-4, Mixtral-8x/B, Llama3-8B, and Llama3-70B are used. Llama3-70B is employed as the claim extractor and checker, as implemented in RefChecker [12].

The RAGChecker is first downloaded from the directory, as described in on the website [29] [12]. This is done on the PC using the following commands:

```
pip install ragchecker
```

Running the query through Private GPT resulted in a .json file, which is then input with the “ground truth” answer for comparison. The ground truth is the real answer to the provided queries in the use cases. Using this, the RAGChecker is able to apply the formulary mentioned in the next subsection 3.4.3.

The aforementioned .json files are obtained and have the following format:

```
{
  "results": [
    {
      "query_id": "<query id>", # string
```

```
      "query": "<input query>", # string
      "gt_answer": "<ground truth answer>", # string
      "response": "<response generated by the RAG generator>", #
    string
      "retrieved_context": [ # a list of retrieved chunks by the
    retriever
        {
          "doc_id": "<doc id>", # string, optional
          "text": "<content of the chunk>" # string
        },
        ...
      ]
    },
    ...
  ]
}
```

Then, each individual query file is run through the RAGChecker using the following commands:

```
ragchecker-cli ^
  --input_path " " ^
  --output_path " " ^
  --extractor_name "gemini/gemini-2.5-flash-lite" ^
  --checker_name "gemini/gemini-2.5-flash-lite" ^
  --extractor_api_base "https://generativelanguage.googleapis.com/
    v1beta" ^
  --checker_api_base "https://generativelanguage.googleapis.com/
    v1beta" ^
  --batch_size_extractor 1 ^
  --batch_size_checker 1 ^
  --metrics all_metrics
```

The RAGChecker employs the Gemini 2.5 Flash Lite. This results in a file with the parameters found in the next subsection 3.4.3. The metrics are then applied to understand how efficient the answer produced has been in reaching the desired output.

3.4.3 RAGChecker Formulary

The RAGChecker framework produces the following metrics to evaluate the success rate of retrieved answers. The metrics are calculated using the formula provided in the paper [12], as shown below.

The metrics were Overall Metrics of PrivateGPT, Retriever Metrics, and Synthesis/ Generator Metrics.

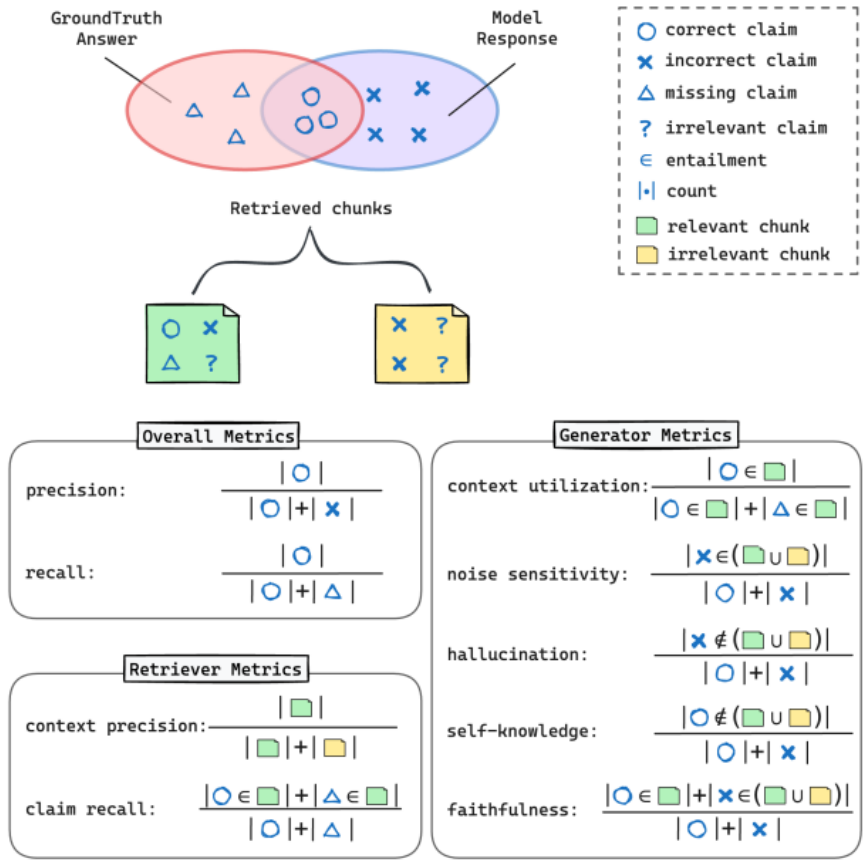


Figure 3.6: Retrieval Augmented Generation Formulary [12]

The model response is denoted as m, the ground truth answer as gt, and the retrieved chunks are $chunk_j$. Leveraging RefChecker, the text is decomposed into a set of claims c_i. Relevant chunks are labelled as $r - chunk_j$, which is if it contains at least one claim $c_i^{(gt)}$, whilst irrelevant ones are labelled as $irr - chunk_j$. k is the total number of all chunks retrieved by the system.

The framework defines a modular RAG system as RAG = R, G, where R is the retriever, and G is the generator. Given a query q and documents D, it first retrieves top-k relevant context $chunk_j$ = R(q, D, k), and then generates a model response $m = G(chunk_j, q)$. For simplicity, we can also represent the overall RAG generation process as $m = RAG(q, D)$ [12].

There are three main metrics used. The first one is Overall Metrics, which evaluates the overall response quality of an RAG system from the user's perspective. Specifically, we first extract claims from a model response m and a ground-truth answer gt as $c_i^{(m)}$ and $c_i^{(gt)}$ respectively. Then, we define correct claims in the response

as $c_i^{(m)}|c_i^{(m)}\epsilon, gt$, and correct claims in the ground-truth answer as $c_i^{(gt)}|c_i^{(gt)}\epsilon, m$. The two metrics, precision and recall, can be computed directly. [12]

Precision

Precision is the proportion of correct claims in all response claims [12].

$$Precision = \frac{|c_i^{(m)}|c_i^{(m)}\epsilon, gt|}{|c_i^{(m)}|} \tag{3.3}$$

Recall

Recall is the proportion of correct claims in all ground-truth answer claims [12].

$$Recall = \frac{|c_i^{(gt)}|c_i^{(gt)}\epsilon, m|}{|c_i^{(gt)}|} \tag{3.4}$$

F1 Score

Further, the harmonic mean of precision and recall yields the F1 score, the overall performance metric [12].

The next set of metrics is Retriever Metrics, which ideally return all claims needed to generate the ground-truth answer. Hence, the measurement of how many claims in the ground-truth answer are covered by the retrieved chunks.

Claim Recall

Hence, claim recall can be computed as the proportion of necessary ground-truth claims that were successfully retrieved [12].

$$Claim, Recall = \frac{|c_i^{(gt)}|c_i^{(gt)}\epsilon, chunk_j|}{|c_i^{(gt)}|} \tag{3.5}$$

Context Precision

The context precision measures the proportion of retrieved chunks that contain useful ground-truth information among the total number of chunks returned by the retriever, k [12].

$$Context, Precision = \frac{|r - chunk_j|}{k} \tag{3.6}$$

The last metrics are Synthesis/Generator Metrics, which, given k retrieved chunks, a perfect generator can identify and include all ground-truth gt relevant claims while ignoring the rest. This is split into six further metrics, which are calculated as follows.

Context Utilization

It signifies how effectively the generator extracts and incorporates the correct facts from the retrieved relevant documents. It is calculated as the percentage of all the correct facts in the ground-truth answer that were actually present in the retrieved text: what percentage did the generator successfully output [12].

$$Context, Utilization = \frac{|c_i^{(gt)}|c_i^{(gt)}\epsilon, chunk_j, and, c_i^{(gt)}\epsilon, m|}{|c_i^{(gt)}|c_i^{(gt)}\epsilon, chunk_j|} \tag{3.7}$$

Noise Sensitivity in Relevant

The generator's tendency to incorporate irrelevant or off-topic information (noise) present in the relevant documents. The proportion of claims in the response that are supported by the relevant retrieved documents but are not part of the ground-truth answer. Here, lower values are better [12].

$$Relevant, Noise, Sensitivity = \frac{|c_i^{(m)}|c_i^{(m)}\notin, gt, and, c_i^{(m)}\epsilon, r - chunk_j|}{|c_i^{(m)}|} \tag{3.8}$$

Noise Sensitivity in Irrelevant

Similarly, here the focus is the generator's tendency to get distracted and copy information from completely irrelevant documents that the retriever fetched by mistake. The proportion of claims in the model's response that are supported only by the irrelevant retrieved documents (documents that do not contain the correct answer). Here too, the lower, the better [12].

$$Irrelevant, Noise, Sensitivity = \frac{|c_i^{(m)}|c_i^{(m)}\notin, gt, and, c_i^{(m)}\epsilon, irr - chunk_j|}{|c_i^{(m)}|} \tag{3.9}$$

Hallucination

Hallucination highlights the generator's tendency to fabricate completely new information that is neither in the retrieved documents nor in the ground-truth

answer. This means the proportion of incorrect claims in the response that are not supported by any of the retrieved documents (pure model hallucination) [12].

$$Hallucination = \frac{|c_i^{(m)}|c_i^{(m)} \not\epsilon, gt, and, c_i^{(m)} \not\epsilon, chunk_j|}{|c_i^{(m)}|} \tag{3.10}$$

Self-knowledge

The self-knowledge focuses on the generator's reliance on its pre-trained internal memory rather than the provided context documents to state correct facts. It is the proportion of claims in the response that are correct (i.e., match the ground truth) but are not present anywhere in the retrieved documents. In RAG, the closer it is to 0%, the better, as it ensures answers are strictly verifiable against the manuals [12].

$$Hallucination = \frac{|c_i^{(m)}|c_i^{(m)} \epsilon, gt, and, c_i^{(m)} \not\epsilon, chunk_j|}{|c_i^{(m)}|} \tag{3.11}$$

Faithfulness

Faithfulness is how strictly the generator's response is grounded in and supported by the retrieved documents (the opposite of hallucination/ outside knowledge). Meaning, out of all the claims the model made in its final response, what percentage are directly supported by the retrieved context. Here, higher values are better [12].

$$Faithfulness = \frac{|c_i^{(m)}|c_i^{(m)} \epsilon, chunk_j|}{|c_i^{(m)}|} \tag{3.12}$$

This concludes the metrics Human-Recorded and RAGChecker metrics data.

Chapter 4

Results/ Case study

4.1 Results

4.1.1 Novice User Efficiency

This section is filled by the novice user while using PrivateGPT. Each provided query is used to ask PrivateGPT how to achieve the specific output, and the user then fills in the data accordingly, depending on how long it takes and how beneficial the acquired responses have been.

Total Interaction Turns, Effective Turns, and Wrong Queries

Remembering that,

$$Total\ Interactive\ Turns = Effective\ Turns + Wrong\ Queries \tag{4.1}$$

Figure 4.1 highlights the Total Interactions, the Effective Turns, and the Wrong Queries achieved for all the Use Cases, respectively. The graph shows the average across all 5 Use Cases. We see that the minimum values generally overlap with the lower quartiles for all Use Cases, except UC 1, which has the lowest value of 1 and a lower quartile of 1.5 in both cases. For Wrong Queries, there are 5 particular tasks that had non-zero values, namely, UC 2.4, UC 2.7, UC 2.14, UC 3.3, and UC 4.6

The Total Interactive Turns overlap with the Effective Turns, as seen in the figure, but there are still some Wrong Queries, as seen in Use Case 2.14, resulting in 8 wrong queries. The specific query 2.14 is, "Set Function Call for place", which leads to vague incorrect queries being asked to PrivateGPT, some even resulting in refusals to provide any answers. This particular query provides the anomalous maximum value of 8 for wrong queries. The minimum number and lower quartile of Total Interactive Turns and Effective Turns are 1, showing that a significant

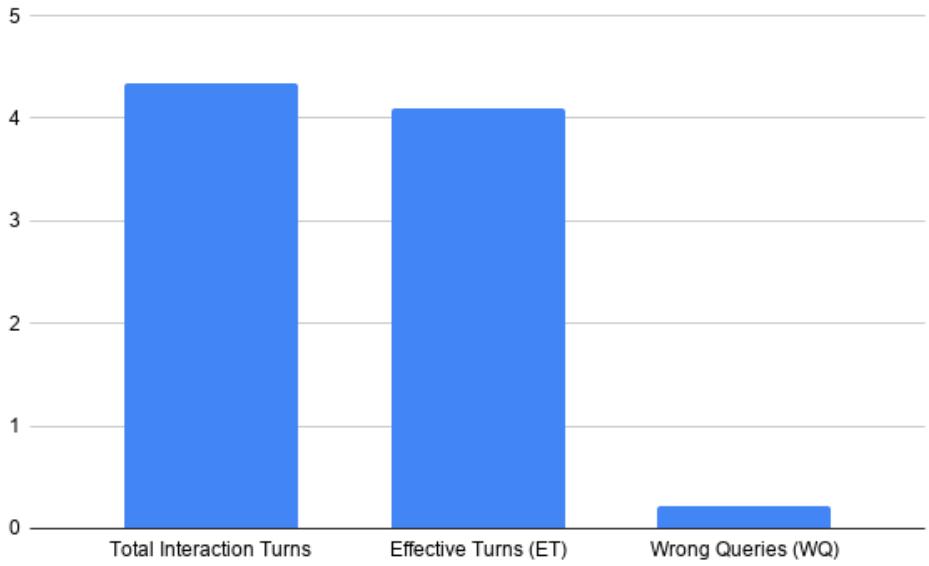


Figure 4.1: Total Interaction Turns, Effective Turns and Wrong Queries

number of use cases are resolved by simply asking PrivateGPT a single query. The upper quartile for both is 6.25, indicating that a small number of tasks do require a high number of queries. The maximum number for both is 17, which stems from two base Use Cases, UC 2.9 and UC 6.10. UC 2.9 had the query, "set the first argument and understand the meaning"; this results in vague and inaccurate answers, which caused the high number of interactions. Meanwhile, UC 6.10, "Move to the safe area before picking up," the manuals do not contain the keyword "safe area", and even after modifying the query several times to achieve the task at hand, this Use Case is considered "failed", as will be seen in the future section 4.1.3.

Theoretical Minimum Turns

The Theoretical Minimum Turns are the minimum number of interactions that the user deems are necessary to obtain a relevant answer. The average values of each Use Case overall can be seen in Fig.4.2.

For Use Case 1, it ranges from 1 to 3, which results in the average of 1,29. This is higher still, despite being the most straightforward Use Case, since it is the first use case attempted.

For Use Case 2, there are 17 individual tasks listed as seen in the Appendix A; due to this, the highest value is observed, which is 2,18. This stems from the lowest value being 1 and the highest being 8. The highest value is observed for UC 2.14, "Set Function Call for place"; this particular task requires the query to be

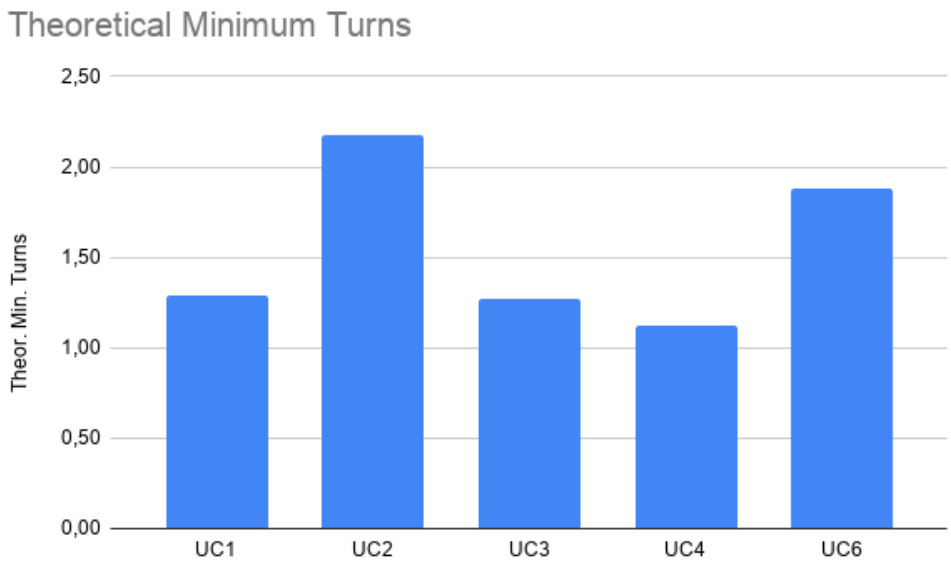


Figure 4.2: Theoretical Minimum Turns

reformulated several times before an appropriate solution is obtained. Even though UC 2.4, 2.5, and 2.9 had a high number of Total Interactions, the minimum number of needed ones is deemed quite low by comparison, which is simply a consequence of not asking the correct, straightforward query in the first instance.

For Use Case 3, the value of 1,27 is seen. This is a direct consequence of all individual tasks within the use case having the theoretical minimum value of 1, except for UC 3.2 and UC 3.8, which have values of 3 and 2, respectively. UC 3.2 also had a high number of interactions overall, as the task, "CHANGE LD variables", lead to answers that do not help achieve the task till several modifications are made to the original query. Furthermore, for UC 3.8, "CREATE OR_SETCU16", the query asked is misunderstood by PrivateGPT till it is reformed and refined, which is why the theoretical minimum interactions is higher.

For Use Case 4, the values fluctuate the least, resulting in the lowest average value of 1,13. Only one, UC 4.7, "TOOL CALIBRATION", required 2 minimum theoretical interactions, as the query needed to be modified slightly; the rest, however, result in only 1 query being needed theoretically.

For Use Case 6, again, 17 individual tasks are observed, which provide us with the second-highest average of 1,88. The highest two values are observed from UC 6.10, "Move to the safe area before picking up," and UC 6.5, "check position[R1] to reset position", with 5 and 4, respectively. For UC 6.10, the keywords "safe area" are incomprehensible to PrivateGPT as they are not present as such in the

manuals. Hence, to avoid hallucination, the query does not provide a valid answer until these wordings are substituted with others actually found in the manuals. For 6.5, the task objective is difficult to understand, hence needing a higher number of interactions.

Learning Rate

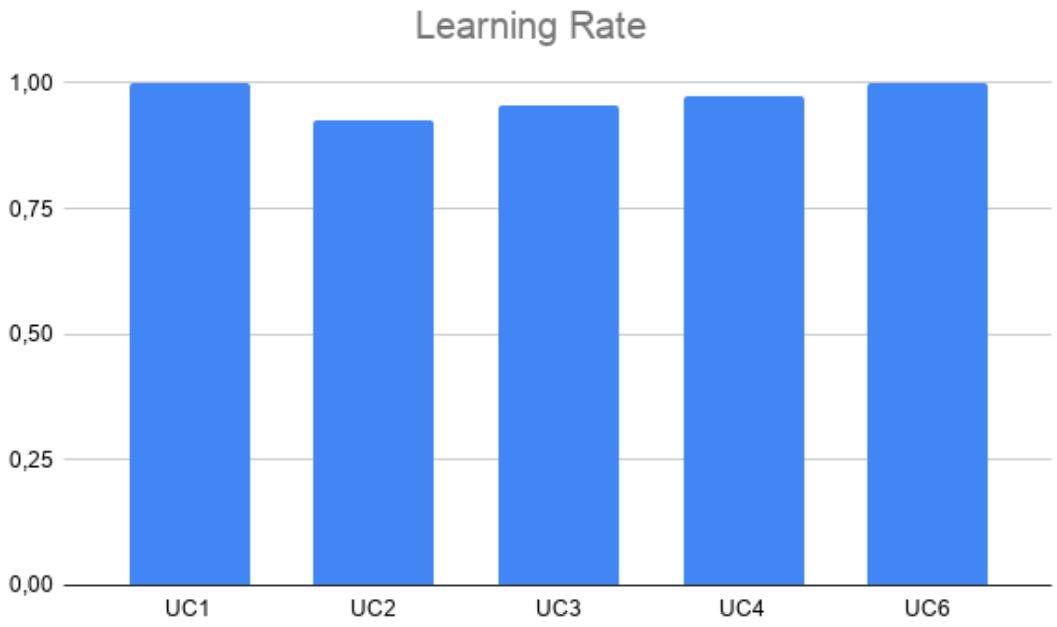


Figure 4.3: Learning Rate

The Learning Rate, as seen in Formula (3.1), is observed as 1, as shown in Fig.4.3 for all queries on average, with the exception of UC 2.4, UC 2.7, UC 2.14, UC 3.3, UC 3.8, and UC 4.6. The highest attainable value is 1, which is when the Total Interaction Turns are equal to the Effective Turns. For cases where it is less than 1, this corresponds to instances where wrong queries are asked, thereby significantly lowering the Learning Rate. For example, for UC 2.7, the lowest Learning Rate of 0,33 is observed when the Effective Turns are 1, and the Wrong Queries are 2, which again relates to the "safe area" wording, leading to 2 semantically incorrect queries being asked.

Intervention

Intervention simply refers to whether a task required any external assistance. There are several kinds of interventions, such as referring to the manuals, needing the assistance of a supervisor, issues with setting parameters, any errors, and/or quick. The value is noted as 1 for yes and 0 for no.

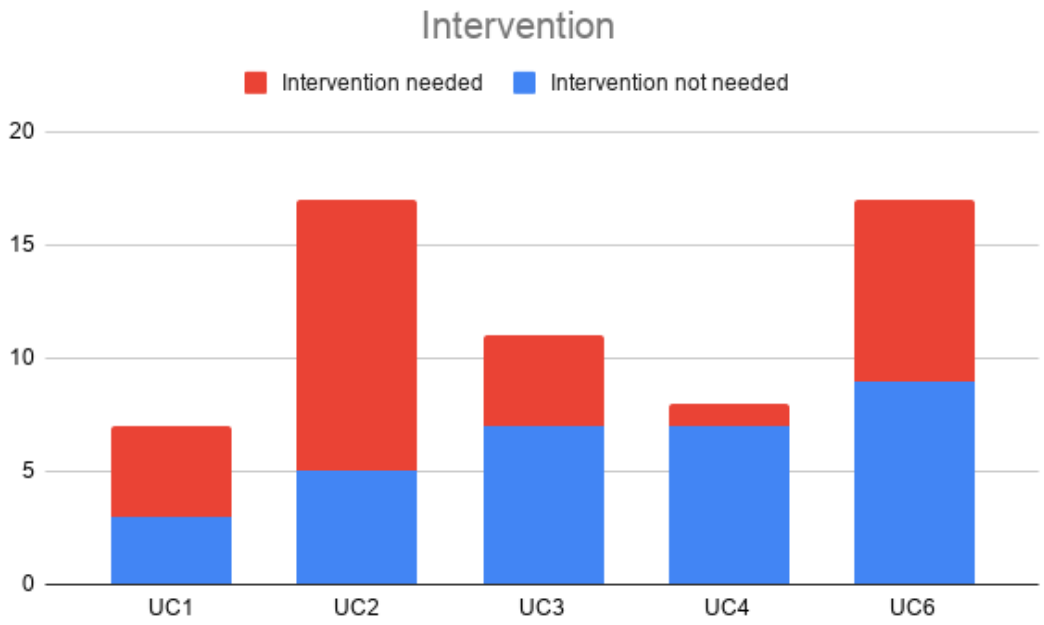


Figure 4.4: Intervention

In UC 1.4, specific wording is searched for in the manual to understand what the provided answer means. For UC 2.1, due to the ambiguity of the task, intervention needing the supervisor's assistance is unavoidable. UC 2.14, UC 2.15, UC 6.3, UC 6.11, and UC 6.16 all require the assistance of the lab supervisor.

Average Time and Latency

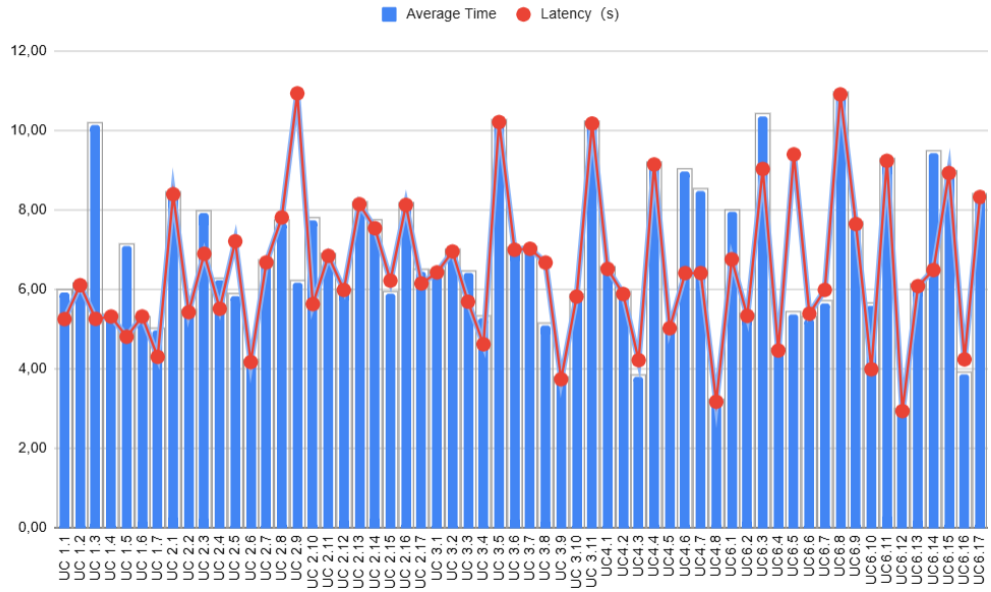


Figure 4.5: Average Time and Latency Original

Average Time is part of Novice User Efficiency while Latency is part of System Quality. However, it is interesting to note the comparison between how long the system took to produce the query deemed correct by the user and the average time taken for all related queries. Queries that lead to refusal are not included in the average. Furthermore, some Average Time and Latency values are equal because only one query needs to be sent to PrivateGPT to achieve the desired output.

As seen in the Fig. 4.5, average times and latency are observed as quite coherent except for a handful of cases, such as UC 1.3, UC 1.6, UC 4.6 and UC 4.7, UC 6.1, UC 6.3, and UC 6.12 in which the Latency is lower than the Average Time of the overall queries, whilst cases like UC 2.9 and UC 6.5 show higher Latency compared to the Average Time.

4.1.2 System Quality

The System Quality refers to PrivateGPT. It asks the user to fill in the metrics based on whether the Effective Turns are useful and on the refusal rate. The metrics are: Task Success Instruction, Ineffective/ Interfering Instruction, Refusal, and Instruction Success Rate (ISR).

Task Success Instruction

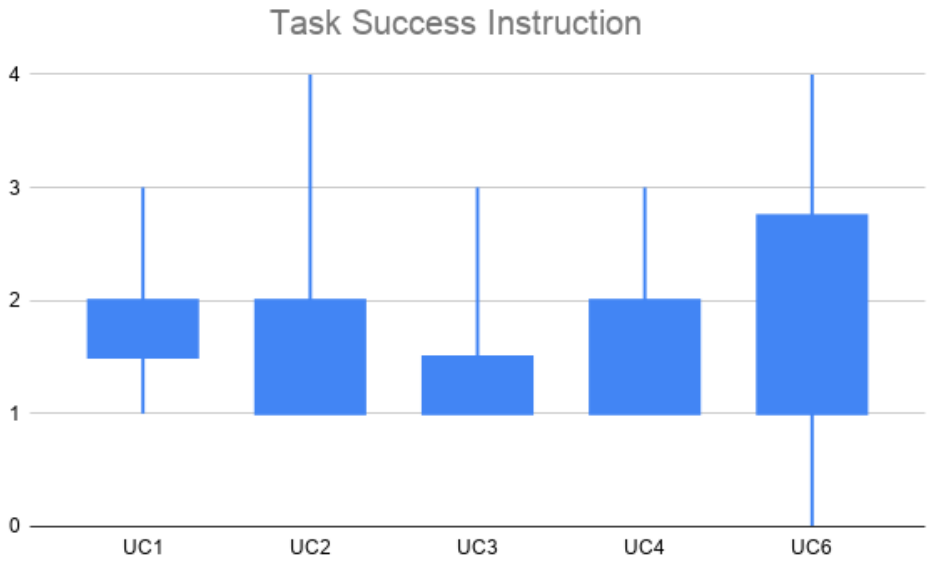


Figure 4.6: Task Success Instruction

The Task Success Instruction refers to how many of the Effective Turns are effective, useful, or helpful in reaching task completion. In this, queries that lead to refusal or do not help reach the correct answer are ignored. However, even if a query provides some context, such as what is present in the manual and what is not, so that the subsequent query could be changed accordingly, it is counted in the Task Success Instruction.

As shown in Fig. 4.6, the lowest value is 0, observed in the case of UC 6.17. This particular task failed, and it is observed that despite the high number of interactions, no particular query is useful in moving closer to the actual answer. The lower quartile and lowest value are 1 for UC 2, UC 3, and UC 4, while the upper quartile is 2 for UC 1, UC 2, and UC 4. The highest value of 4 is observed for three cases: UC 2.4, UC 6.10, and UC 6.14, which is due to the high number of queries asked to PrivateGPT, resulting in a high number of Task Success Instructions, as these were necessary to achieve a valid answer.

Ineffective/Interfering Instruction

The Ineffective/Interfering Instruction usually has the same range, with the exception of some queries. UC 2.3, "CHOOSE THE RIGHT Coordinates" with 6. UC 2.5, "RESET GRIPPER" with 5. UC 2.9, "Set the first argument and understand

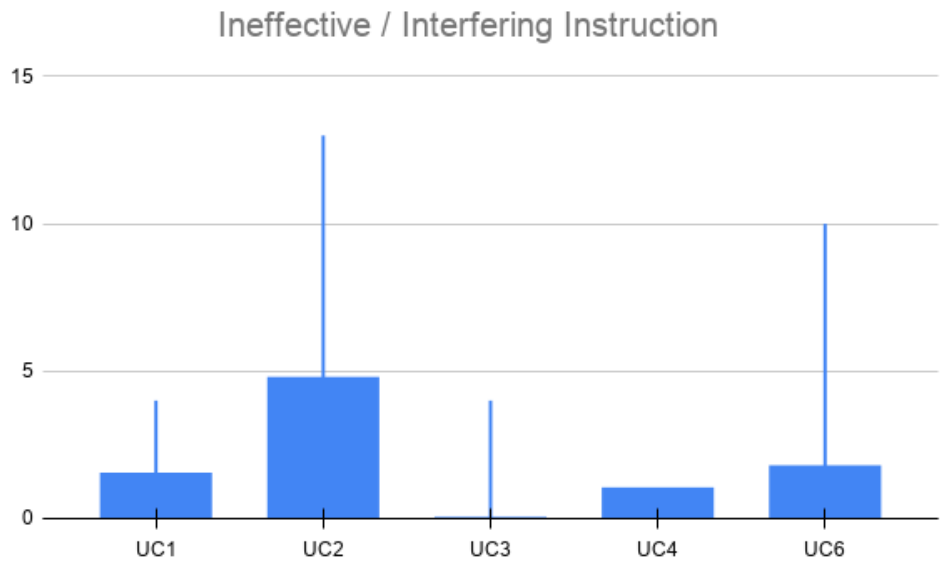


Figure 4.7: Ineffective/Interfering Instruction

the meaning" with 13, UC 2.14, "Set Function Call for place" with 5. UC 6.10, "Move to the safe area before picking up" with 5. UC 6.17, "Return to the starting position" with 10. They exhibit higher numbers due to the nature of the tasks alongside their particular phrasing. This also means that 2.17, 6.10, and 6.17 result in failed queries, as seen in the future subsection 4.1.3.

As seen in Fig.4.7, the lowest value and the lower quartile for all 5 Use Cases is 0. The upper quartiles are 1, 5, 4,75, 0, 1, and 1,75, respectively, while the highest values for each use case, in order, are 4,00, 13,00, 4,00, 1,00, and 10,00. The highest in UC 1 is given by UC 1.5, "CHANGE EXISTING POSITION " and UC 3.8, "CREATE OR_SETCU16" in UC 3. In both cases, the task leads to vague queries that are not useful for the final results. For Use Case 4, the upper quartile and maximum are both 1.

Refusal

Refusals are cases in which, regardless of the queries, the result from PrivateGPT is "The answer is not in the context provided."

As seen in Fig. 4.8, the minimum values for all overall use cases and the lower quartile are 0. Meanwhile, the upper quartile values vary from UC 1 to UC 6 as 0, 2, 0,25, 0, and 1,75. It is interesting to note that UC 1 and UC 4 have upper-quartile refusal values of 0. UC 2 exhibits the highest upper-quartile value of 2, alongside also the second-highest refusal value of 7 due to UC 2.16, "Lift Up", as the manuals

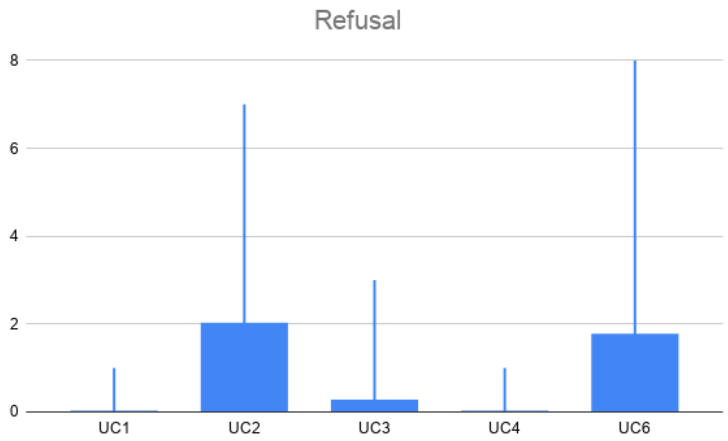


Figure 4.8: Refusal

do not have the exact words, and this leads to a higher refusal rate. Furthermore, UC 6 has the second-highest upper quartile value of 1.75 and the highest maximum value of 8, attributable to UC 6.10, "Move to the safe area before picking up," due to the keyword "safe area".

Instruction Success Rate (ISR)

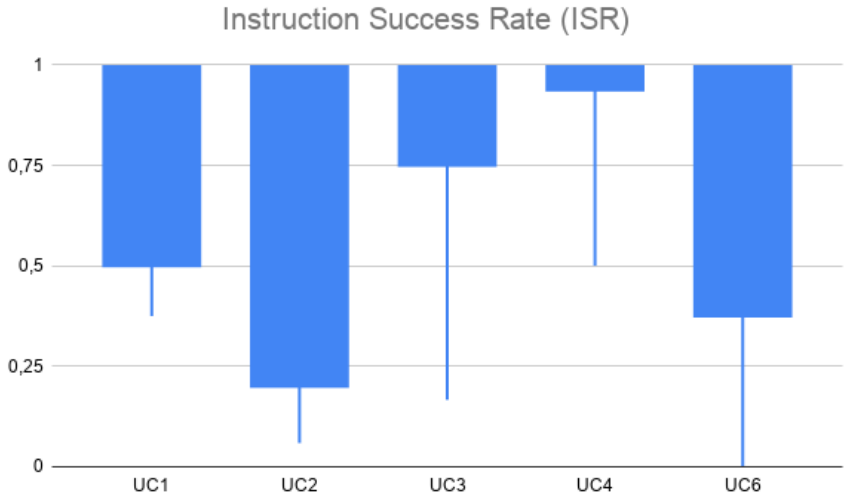


Figure 4.9: Instruction Success Rate

As mentioned previously, the ISR is given by the Formula (3.2). It is the ratio of Task Success Instruction and the Effective Turns.

As shown in Fig.4.9, the upper quartile and the highest value for each overall Use Case are 1. The lowest values in each of the cases are: 0,38, 0,06, 0,17, 0,50, and 0,00. The lowest value of 0 is from UC 6, due to the failed task UC 6.17, "Return to the starting position". UC 4 has the lowest Interquartile Range and therefore the best overall Instruction Success Rates. UC 2 has the highest Interquartile Range, also producing the highest variety of values and the lowest lower quartile of 0,20.

4.1.3 Overall Metrics of PrivateGPT

The Overall Metrics of PrivateGPT are the metrics obtained by the RAG Checker after answers are retrieved from PrivateGPT. These answers are then compared to the Ground Truth, i.e., the actual answer, and analysed for accuracy.

State- TSR

Starting with the first metric, the TSR is simply a measure of the success rate:

- 1, if successful in retrieving a relevant response;
- 0, if a failure in retrieving a relevant response.

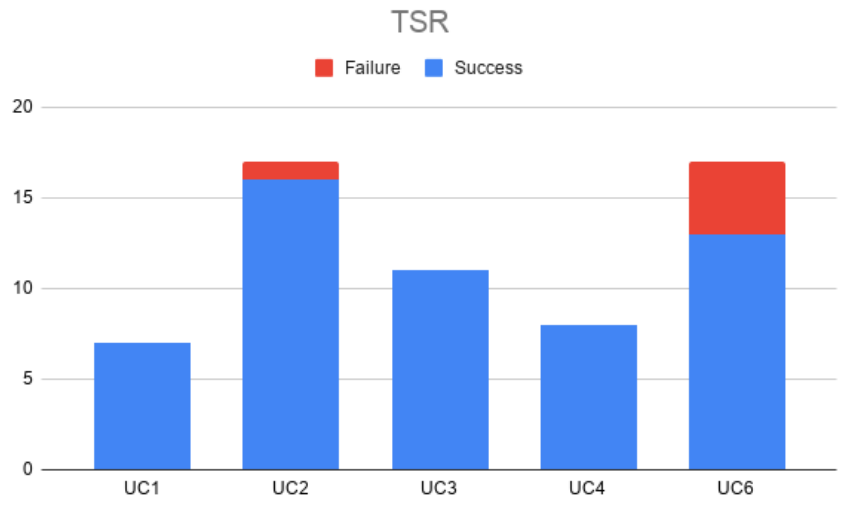


Figure 4.10: State-TSR

As seen in Fig. 4.10, the proportion of failed tasks in UC 2 is 5,88% while for UC 6 it is 23,53%.

There are five use cases, 2.17, 6.10, 6.12, 6.14, and 6.17, that are the failed queries.

The UC 2.17 and UC 6.17, "Return to the starting position", fail because the term "starting position" is not directly found in the manuals. Looking at the ground truth, the objective is to copy the initial position and paste it again to bring the robot back to the starting point, but with either attachment, PrivateGPT fails to provide a valid answer.

The UC 6.10, "Move to the safe area before picking up", fails because the phrase "safe area" is not found in the original PDFs. The objective is to simply move the robot down to the correct position above the object, but the answers provided do not come close to achieving this.

The UC 6.12, "SET ARGUMENT OF instance" fails as the PrivateGPT only provides the same repeated answer no matter how the query is phrased, "'ARG(1):Instance' is retrieved using 'GETARG LD001 IARG#(1)'" although what is needed is to select the "Call Job" feature and select the argument as 1.

Lastly, UC 6.14, "Move to the placement point," fails because the "placement point" in the query is misunderstood by PrivateGPT. Although the answer mentions that the robot must be moved to the correct location, it does not specify how to do so.

Precision

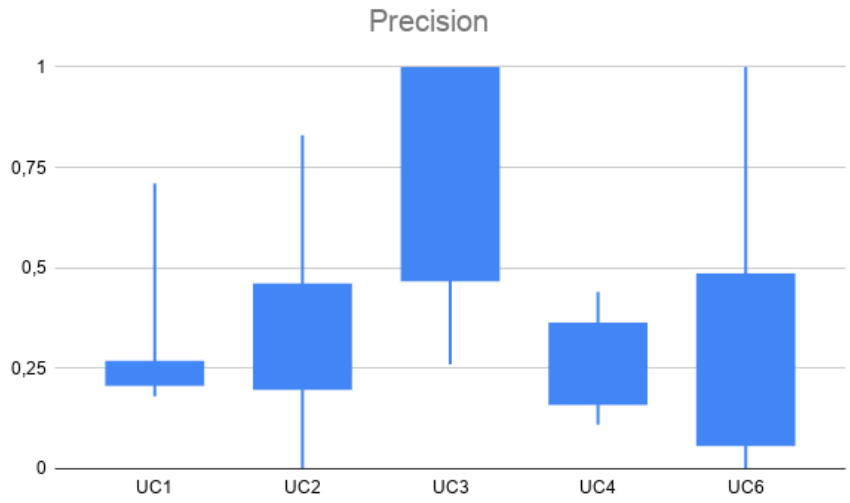


Figure 4.11: Precision

The Precision metric includes the proportion of claims in the generated response that are correct (i.e. contained in the ground truth). The formula (3.3) is as seen in the previous Chapter in Section 3.4.3.

As seen in Fig. 4.11, the minimum values for each Use Case in order are: 0,18, 0,00, 0,26, 0,11, and 0,00. The 0,00 values in UC 2 and UC 6 arise from the failed tasks, as predicted. UC 1 has the lowest IQR, while UC 3 has the highest, indicating greater variation in precision. Overall, the highest maximum value and upper quartile are also seen in UC 3 with 1, which is the maximum possible value. UC 6 also has several cases with 1 as the maximum value. Overall, UC 6 shows the widest range of precision values due to the wide range of tasks and the high number of tasks in this Use Case. The highest average precision is exhibited by Use Case 3 with a value of 0.7, again highlighting the best performance precision-wise.

Recall

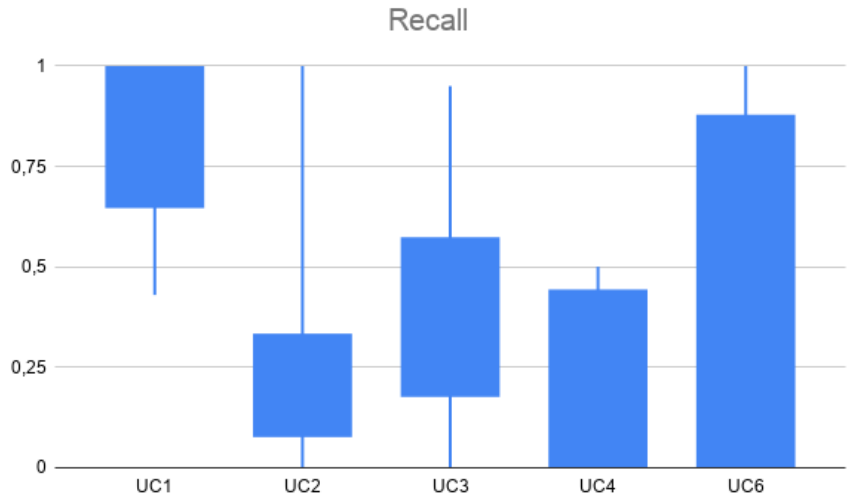


Figure 4.12: Recall

The proportion of ground-truth facts (claims) that are successfully covered or mentioned in the model's final response. It measures the completeness of the answer. If the ground truth has 10 instructions, and the model lists 8 of them in its response, the overall recall is 80%. This is also seen in Formula (3.4).

As seen in Fig. 4.12, UC 1 has the best recall, with the maximum value and the upper quartile both at 1, whilst the lower quartile is at 0.65 and the minimum is 0.43. Meanwhile, UC 6 again has the highest IQR and range due to its variance and the number of tasks within it.

Furthermore, as seen in Fig. 4.12, UC 2 has a range of 1 overall, meaning there are values with recall 0 and 1, but it has a much smaller IQR. UC 3 has a high range, with the maximum value at 0.95. UC 4 has a lower overall recall, with a maximum value of 0.5, due to the disconnect between the answer provided and the ground truth.

F1 Score

Lastly, the F1 score is the harmonic mean of overall precision and recall.

Fig. 4.13 highlights the F1 scores retrieved overall. For UC 6, we see the highest global maximum value of 1 due to UC 6.1, UC 6.2, and UC 6.3, despite other values and an average at the lower end, as the lower quartile aligns with the lowest value due to several other tasks. The best performance is displayed by UC 1 with an average value of 0,46, closely followed by UC 3, with 0,44. Overall, the F1 score yields an average of 0,317, consistent with the Precision and Recall results.

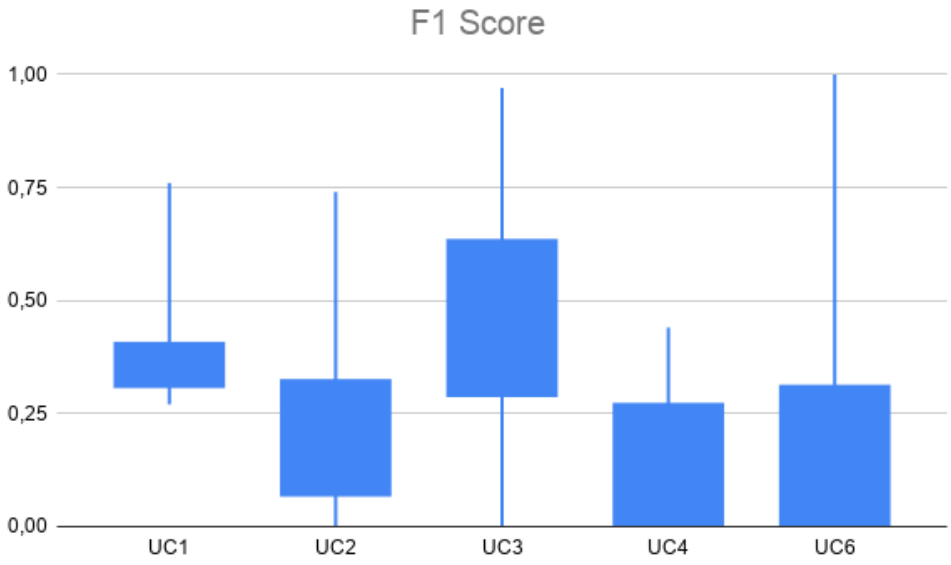


Figure 4.13: F1 Score

4.1.4 Retriever Metrics

The Retriever Metrics also depend on the RAGChecker, the retrieved answer, and the Ground Truth.

Claim Recall

Claim Recall refers to the proportion of ground truth claims that are present in the retrieved context, which measures the completeness of retrieval.

The Claim Recall averages 0,415 across all use cases and queries. This is calculated using the Formula (3.5).

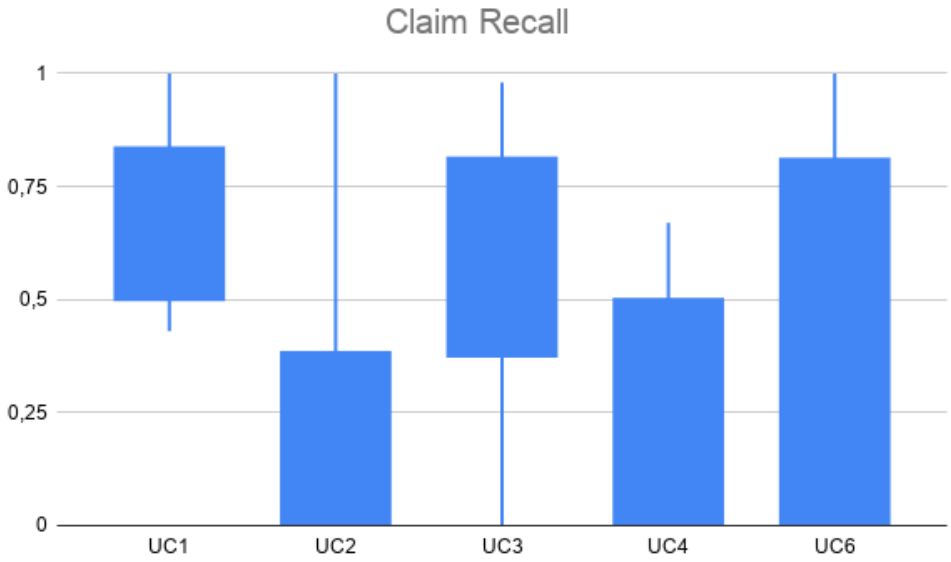


Figure 4.14: Claim Recall

Observing Figure 4.14, it is noted that UC 1 performs the best, not only exhibiting the maximum value of 1 but also having the highest upper quartile, lower quartile and minimum value, giving an average of 0,7 overall. Meanwhile, UC 3 also performs well; the only outlier is one task, UC 3.5, which yields a low Claim Recall of 0, a problem in previous parameter evaluations as well. For UC 2 and UC 6, they have tasks with the highest Claim Recall, but their lower quartiles at 0 indicate that several tasks have low Claim Recall. This is also attributed to the high number of tasks in both these use cases. For Use Case 4, the lowest average of 0.26 is observed.

Context Precision

Context Precision entails the proportion of retrieved context chunks that are relevant to the query. If Context Precision is low, the chatbot is flooded with irrelevant manual pages. This forces the LLM to read through thousands of words of useless text. This increases the time to answer (latency) and the risk of the

LLM becoming confused and generating an incorrect step. Additionally, a low Context Precision score indicates that the embedding model (the mathematical search engine) or the chunking strategy is weak. It is pulling in documents that share similar keywords but are completely irrelevant to the actual question.

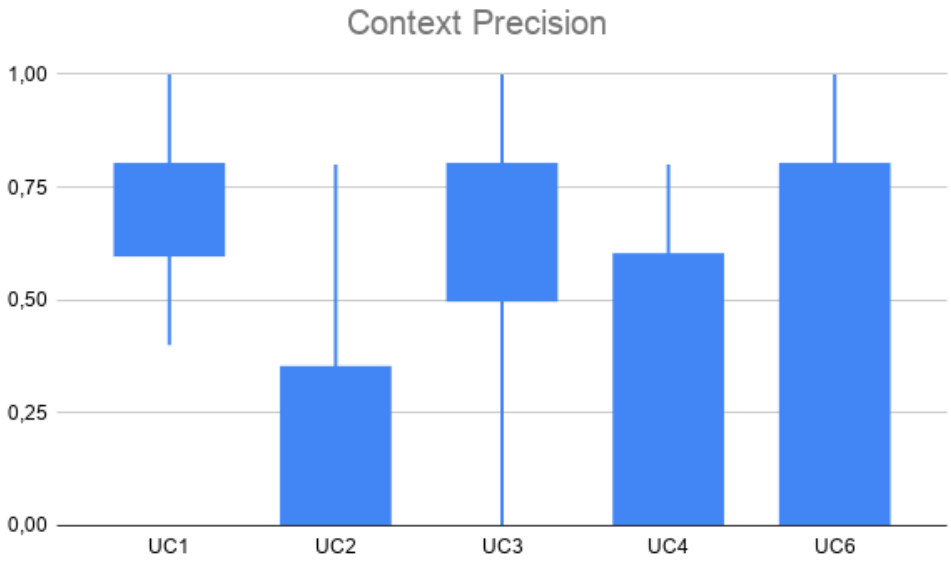


Figure 4.15: Context Precision

The Context Precision gives the overall average of 0,437. This is calculated using the Formula (3.6). As seen in Figure 4.15, the UC 1, UC 3, and UC 6 all exhibit maximum values of 1, with upper quartiles at 0,8 for all except UC 3, which has 0,6. UC 1 displays the best results, with a minimum value of 0,4, while UC 2, UC 4, and UC 6 display lower quartiles of 0, indicating that a relevant number of tasks have 0 context precision. For UC 3, the minimum value of 0 is due to one specific task, UC 3.5, which concerned understanding data loops; this query, too, was unable to reach an adequate answer.

4.1.5 Synthesis/ Generator Metrics

In the RAG Checker framework, the Synthesis/ Generator Metrics evaluate how well the AI chatbot reads the retrieved manual pages and generates the final response. It comprises six individual metrics.

Context Utilization

Beginning with Context Utilisation, which is the proportion of relevant facts in the retrieved context that are actually extracted and used in the generated response. The calculation yields an overall average of 0,507 using the Formula (3.7).

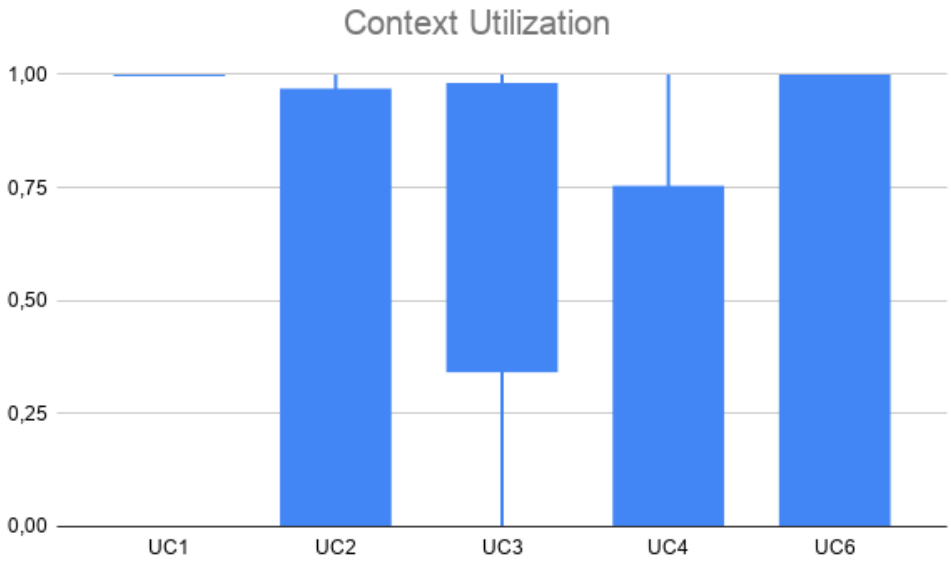


Figure 4.16: Context Utilization

Referring to Figure 4.16, UC 1 performs the best as all the tasks within it exhibit Context Utilisation as 1. This shows how effectively the AI chatbot extracts and uses the correct facts retrieved from the manuals.

For UC 2, UC 3, UC 4, and UC 6, a high range of values is seen, from 0 to 1. For UC 2, UC 3, and UC 6, the upper quartile also overlaps closely with 1, which is a very good indicator. However, the lower quartile at 0 indicates that the information retrieved in some tasks is incorrect. The interquartile ranges for UC 2 and UC 6 indicate the wide variance in the results obtained.

Noise Sensitivity in Relevant/ Irrelevant

The Noise Sensitivity in Relevant/Irrelevant features the generator's tendency to include incorrect or irrelevant facts from the context. The overall averages obtained are 0,263 and 0,222 through the Formulae (3.8) and (3.9) for Noise in Relevant and Irrelevant, respectively. The lower the Noise Sensitivity value, the better. The Noise Sensitivity in Relevant shows that the chatbot was given the correct manual page, but that page contains extra text. The chatbot gets distracted and includes

these extra, unneeded details in the final answer. The Noise Sensitivity in Irrelevant signifies that the search engine made a mistake and retrieved a completely irrelevant manual page. A highly sensitive chatbot will start explaining this wrong page in its answer, whereas a noise-resistant bot will ignore the irrelevant page.

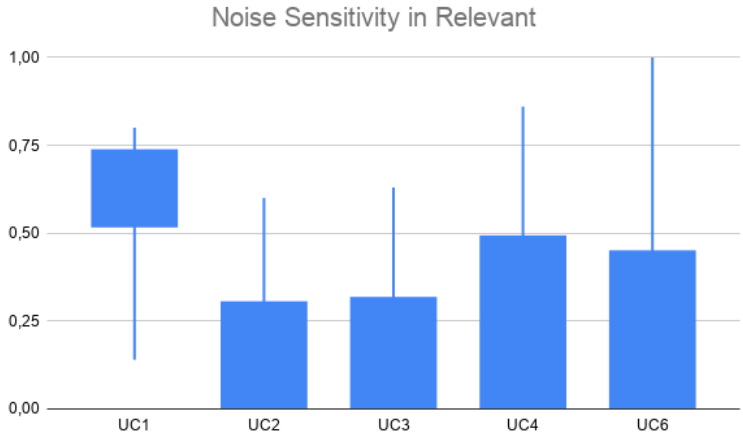


Figure 4.17: Noise Sensitivity in Relevant

For the Noise Sensitivity in Relevant, as seen in Figure 4.17 UC 2, UC 3, UC 4, and UC 6 exhibit the lowest minimum and lower quartile, hence showing how a significant number of tasks had 0 in this parameter. UC 6 exhibits the maximum possible value as well; although the upper quartile is comparatively lower at 0,45, the task UC 6.9 yields the maximum value of 1. Although the correct answer was present, leading to a faithfulness of 1, it was drowned out by excess irrelevant information. Lastly, UC 1 presents high values overall, as although the correct answers were present, there was also an excess of unnecessary information.

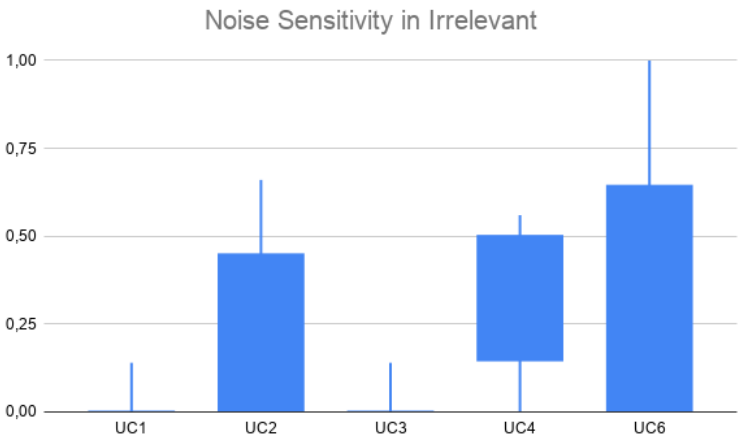


Figure 4.18: Noise Sensitivity in Irrelevant

Likewise, Noise Sensitivity in Irrelevant, as seen in Figure 4.18, UC 1 and UC 3 perform the best with the lowest values overall, and even the maximum value is 0,14 for both of them, meaning that, when working on these cases, PrivateGPT rarely went to the wrong page in the PDFs. The minimum values for all use cases are 0, whilst only UC 6, specifically task UC 6.12, arrives at the maximum value of 1. This anomalous result is due to the word "argument" appearing in the task wording, which the chatbot misinterpreted throughout the research experiment.

Hallucination

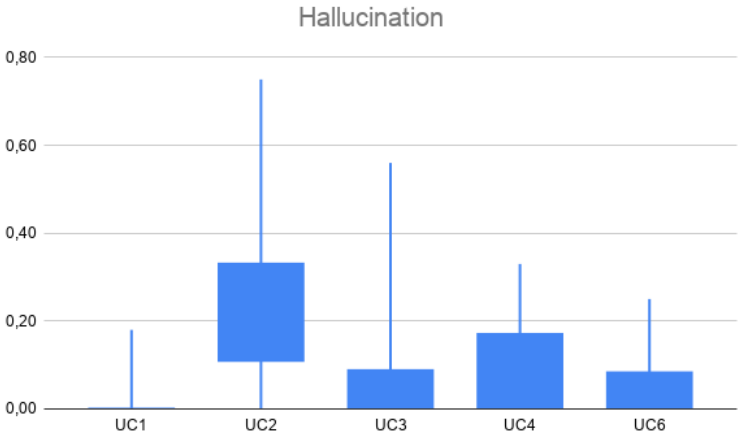


Figure 4.19: Hallucination

Hallucination gives the proportion of generated claims that are incorrect and not found anywhere in the retrieved context. The calculations are performed using the Formula (3.10), resulting in an overall average of 0,121, which is excellent, considering that one of the primary objectives is to minimise hallucination.

Focusing on the graph generated, Figure 4.19, UC 1 is seen to be performing the best with the minimum value, and lower and upper quartiles all at 0, with only one task, UC 1.6, showing slight hallucination at 0,18, which is an anomaly and is due to the task type requiring the speed of movement to be changed. Moreover, UC 3, UC 4, and UC 6 have minimum and lower quartiles at 0, indicating that most queries exhibited 0 hallucination. Only UC 2 showed some high values of hallucination, specifically with task UC 2.12 obtaining a value of 0,75 which is the highest overall. This was due to the task requiring a simple argument to be assigned, but PrivateGPT was unable to locate the specific wording in the manual, resulting in higher hallucination rates overall.

Self-knowledge

Self-knowledge, highlighting the proportion of correct generated claims that are not present in the retrieved context. This measures the reliance on pre-trained weights rather than context. The Formula used is (3.11), which provides an overall average of 0,063.

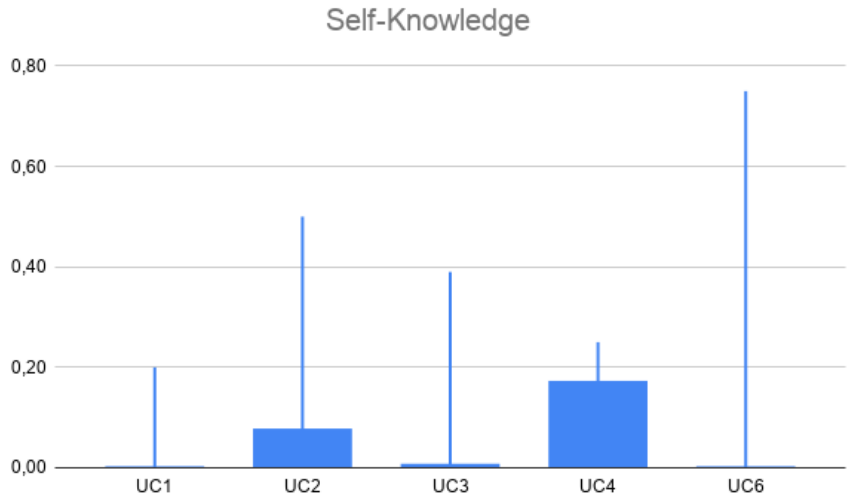


Figure 4.20: Self-knowledge

Looking at Figure 4.20, it can be seen that for UC 1 and UC 6, the minimum, lower quartile, and upper quartile are all at 0, which puts emphasis on PrivateGPT

entirely using the manuals to answer the questions rather than using its own pre-trained memory, which is a brilliant performance indicator. Additionally, the other three use cases, UC 2, UC 3, and UC 4, also have minimum values and lower quartiles at 0 whilst having slightly higher upper quartiles at 0,08, 0,01, and 0,17, which are not as high. UC 6 has a high maximum value, which can be attributed to a specific task in UC 6.1 that required a step-by-step response to transmit a job.

Faithfulness

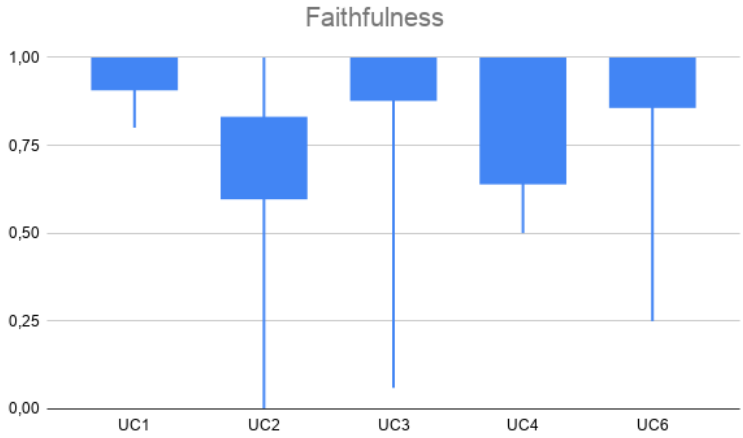


Figure 4.21: Faithfulness

Lastly, Faithfulness includes the proportion of claims in the generated response that are supported by the retrieved context. Using the Formula (3.12), the overall average obtained is 0,817 which is a very high value.

Figure 4.21 highlights the strength of the results obtained by PrivateGPT, with the upper quartiles and the maximum values for 4 of the 5 use cases overlapping at 1, the highest possible value. This shows that the performance was very efficient. Even UC 2 has a maximum value of 1 whilst the upper quartile is 0,83. The lowest value of 0 overall is observed only in UC 2, specifically in task UC 2.12, where the task was not correctly interpreted by the chatbot despite different wording. Overall, the high Faithfulness is a direct indicator that the statements and steps generated by the chatbot were directly rooted in the provided YASKAWA manuals.

4.2 Discussion of the Results

The overall performance of the PrivateGPT exhibits a high success rate of 91,67%, meaning that, in 9 out of 10 cases, the answers provided correctly guide the

user to complete the robot programming task. However, the RAG F1 score of 31.70% indicates that, despite the high completion rate, the very first answers are rarely complete or perfect, meaning they either miss some steps (recall) or include irrelevant information (precision). The Instruction Success Rate (ISR) is high at 70,99%, indicating that roughly 7 out of 10 steps generated by the PrivateGPT were immediately correct and executable on the physical YASKAWA robot.

Since a higher Claim Recall of 0,415 is observed compared to the F1 Score, it implies that the search engine did successfully find a large chunk of the facts, but the LLM generator failed to read or write them down. The lower overall Claim Recall could also imply that the search was too restrictive.

In the end, the PrivateGPT framework was extremely useful, especially for a novice user, as it spared them from sifting through hundreds of pages of PDF manuals to locate specific instructions or seek assistance with operations. PrivateGPT found the correct sections, making it highly useful as a guide and lab assistant. However, the bot required an average of 4,34 interaction turns and a refusal rate of 1,02 per task, meaning the user still had to engage in a dialogue with the interface to obtain the most useful solution. The interface required rephrasing of queries and follow-up questions to extract the correct steps hidden in the PDF clutter. The bot achieved a high Faithfulness rate of 81,66%, meaning 81,66% of the statements and steps generated by the chatbot were directly taken from the provided YASKAWA manuals. This shows that PrivateGPT is highly disciplined and generally avoids drawing on unrelated internet search data, making sure that the instructions correspond to the official manuals. Also, it exhibits a low Hallucination rate of 12.10% . This rate, though low, requires some supervision to evade any safety-related issues with the cobot.

The chatbot shows a high reliance on source material, with a low Self-Knowledge of 6.37%. This means only 6.37% of the correct facts in the chatbot's answers came from its general training memory. Over 93.6% of the correct information was extracted directly from the retrieved manual pages. This is great, as in robot programming one does not want the model to guess or rely on general knowledge. The extremely low self-knowledge score indicates that the bot relies exclusively on the specific YASKAWA manuals uploaded to PrivateGPT.

High resilience to irrelevant information is seen as 73.61% Noise Resistance. This means that when the retriever accidentally fetched irrelevant information, the generator successfully ignored that noise 73,61% of the time (Noise Sensitivity in Relevant Context was only 26.39%), showing high filtering capabilities. Also, in Noise Sensitivity in Irrelevant Context, a low percentage of 22,25% is seen. This shows that when the retriever accidentally fetched irrelevant chapters or unrelated page clutter, the generator successfully ignored that noise.

Despite the overall great performance, there were some areas of the system and its behaviour that were lacking. Possible reasons include context preservation

in large chunks, which can lead the system to retrieve large, raw sections of the original PDFs. For simple sequential tasks, this is highly beneficial because all the steps are clearly highlighted. However, the drawback is that the user must interact with the bot multiple times to obtain the most accurate answer. Possible remedies and future work are discussed in the next Chapter 5.

Chapter 5

Investigation for Future Works

The Chapter details the investigation for document parsing whilst highlighting the work done and the restraints within it, opening up future possibilities for further analysis.

5.1 Document Parsing

Document parsing is the process of converting a raw, unstructured document (such as a PDF, Word document, or scanned image) into a clean, structured, machine-readable format (such as plain text, Markdown, or JSON).

A simple analogy is reading a restaurant menu that is unparsed: it has a visual layout with prices aligned to the right, columns separating starters from mains, and decorative borders, section titles, and page numbers.

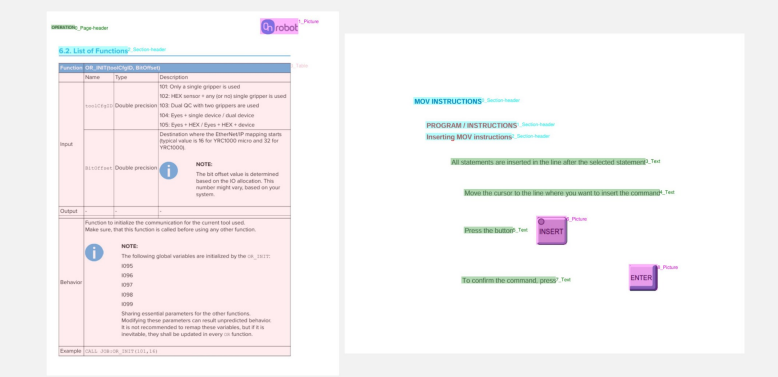


Figure 5.1: Parsed File Example

Document Parsing is a parsing tool that reads the menu and converts it into a structured list. It removes the visual borders and page headers, keeping only the raw, organised data as seen in Fig. 5.1. The importance of document parsing is evident in RAG systems, such as the YASKAWA PDF manuals, which are written for humans and contain visual elements that can be difficult for computers to read, including multiple columns, tables, junk text, and images. The parser reorganises multiple columns so that, instead of a computer reading a raw PDF straight across the page, blending two columns together, it reads down column 1, then down column 2. With tables, a parser converts visual grid lines into structured rows so the AI understands which setting belongs to which parameter. For junk text, a parser removes repeated page headers, footers, page numbers, and margin notes to avoid clutter. In short, document parsing is the clean-up step that translates human-friendly layouts into AI-friendly data.

5.2 Parsing and Testing the Manuals

The manuals were parsed using the dots.mocr program [27] in order to clear up the data to make it easy for the AI to read and search information through.

These parsed manuals were then uploaded to the PrivateGPT platform instead of the original PDFs. Furthermore, the tests were rerun to assess possible improvements in the system. It was anticipated that, since the clutter was removed and the data was more organised and accessible on the platform, the answers retrieved would have higher overall values.

Using the same queries, PrivateGPT was utilised with these new files in their markdown version. However, even with correct parsing, the overall results of the parsed version were slightly worse, despite several attempts.

As seen in the Table 5.1, the results show how all of the parameters performed worse. For example, Precision was 13,11% lower. The only improvements seen were in the Noise Sensitivity in Relevant/Irrelevant. Furthermore, even the Self-Knowledge applied increased by 8,57%.

This issue can be traced back to several factors and possibilities and can serve as the basis for future work. Starting with, even though the parsed text was merged back into single files to match the exact page structure of the original PDFs, changing the file extension from '.pdf' to '.md' alters how the RAG database cuts, reads, and searches the files.

The first issue that arises with the dots.mocr program converts images to base64-encoded strings, which LLMs cannot understand, making information retrieval difficult. Moreover, this causes issues with chunking strategies, leading to reduced performance and accuracy; for example, even if the required text was in the .md file, PrivateGPT was unable to retrieve the answer. Whilst tables were read correctly,

	Percentage Difference
Precision	13,11%
Recall	9,05%
F1 SCORE	3,45%
Claim recall	17,31%
Context Precision	22,12%
Context Utilization	8,93%
Noise Sensitivity in Relevant	5,09%
Noise Sensitivity in Irrelevant	8,57%
Hallucination	-22,17%
Self-knowledge	-8,57%
Faithfulness	6,07%

Table 5.1: Percetage Difference of Results in PDF vs. Markdown Files

images were not. Instead of extracting text from images, base64-encoded strings were inserted, which not only increased the file size but also introduced further performance issues as the chunking was performed incorrectly. Due to this, there is a need for a new multimedia OCR parser that can read image text rather than convert it to this format.

A possibility is that the plain-text chunking cuts the sentences in the wrong places. For instance, for PDFs, the system uses a specialised PDF reader that splits the text based on the visual page boundaries and layout structure. However, for Markdown files, the system treats them as fat text files and splits them at character limits or double newlines. This poses a problem, as the reader does not understand the page layout, which splits the text in arbitrary places, often cutting a table or step-by-step instruction list at the wrong points, making it impossible for the reader to retrieve the whole answer.

Another possibility could be keyword depletion. In Markdown files, clutter such as page numbers, document titles, and chapter names at the top and bottom of every page is stripped away. In the unparsed PDF version, every page still has this information clutter, but it acts as a search tag. Meanwhile,, for the parsed version of the file, since there is keyword depletion and repeated labels are removed, the search engine has a harder time matching a generic user query to the correct page chunk.

Furthermore, the table syntax could also be a potential problem. In the PDFs, the tables, though they can be misread, can still be read as simple text alignments. However, in the Markdown version, the formatting is changed to pipes "|" and hyphens "-". The search engine is trained on natural language, and in the parsed tables, these add semantic noise that lowers precision due to syntax formatting.

Additionally, the parsed files present the possible problem of loss of visual geometry. The PDF parsed uses the coordinate boundaries to group headers and body paragraphs, but once parsed, the document is just words. The retriever loses the ability to recognise section blocks based on visual layout, further contributing to poor document ranking.

Conclusion

The thesis was started from elaboration on the background knowledge and state of the art of RAG, LLMs, AI, cobots, and their collaboration to improve the end results.

The thesis successfully exhibited the viability and the optimisation of the localised Retrieval-Augmented Generation framework for collaborative robot programming. By deploying a localised PrivateGPT grounded in official YASKAWA manuals, the work established a secure, low-latency, and highly usable interface that shifts the operator's role from low-level manual scripting to high-level process supervision.

The thesis then moved to the elaboration of the PrivateGPT and how the usage of this framework assisted in using the YASKAWA robot with minimal assistance and external intervention, hence, promoting a novice user to learn how to use the robot quickly and to be able to use the robot efficiently through the PrivateGPT platform by asking relevant queries till the task is achieved.

In comparison to simply using AI to solve queries, the thesis saw an extremely high Task Success rate of 91,67% which highlights the number of tasks that were successfully achieved through this methodology. Furthermore, an average value of only 0,475 per task required intervention whilst the rest were solved without this.

The work exhibited strong grounding and safety compliance which is essential on the shop floor as the system exhibited 81,76% Faithfulness and only 12,10% Hallucination rate whilst also demonstrating a very low Self-Knowledge rate of 6,26%, hence, proving that the AI is not guessing instructions but strictly using verified data from the uploaded Yaskawa manuals.

Rather than evaluating a black boxed version of the chatbot, the thesis evaluated the human usability metrics and RAG metrics. This helped produce a complete work for auditing AI assistants in industrial settings.

Furthermore, the thesis does not just focus on theoretical work but has a direct real world application by the implementation of the chatbot's instructions on the YASKAWA cobot. It successfully tested operations ranging from basic motion commands to coordinate calculations as well as operations based on picking and placing objects using the RG6, robotic gripper, and MG10, Magnetic Gripper.

Lastly, the beginning of the investigation was performed for some pre-processing strategies. Files were parsed using the dots.mocr framework, however, there were some bottlenecks, such as the parser converting images into base64 strings which disallowed PrivateGPT from being able to read the .md files correctly due to possible incorrect chunking leading to lower performance metrics. For this, a further analysis on the file parsing should be performed and the specific obstacles should be removed so as to improve the data retrieval from the files and improve the overall system efficiency. Alternatively, a different parsing tool should be used entirely to avoid this issue.

To conclude, the thesis observes that the implementation of the tool, PrivateGPT, is extremely useful in not just this particular case but it could also be utilised for other robots and in other manufacturing scenarios.

Appendix A

Use Case Tasks and Queries

Task ID	Use Case Task	Use Case Query
UC 1.1	CREATE A NEW JOB	how to create a new job
UC 1.2	CHOOSE THE RIGHT MOTION TYPE AND CHANGE	the correct type of motion and how to change it on the robot
UC 1.3	inserting a Move instruction	how to add the motion to move the robot
UC 1.4	create reference point	to create a reference point for the robot
UC 1.5	CHANGE EXISTING POSITION	to change the existing position of a move instruction
UC 1.6	CHANGE THE SPEED OF MOVJ	focusing on how to change the speed of MOVJ which is a motion type
UC 1.7	delete operations	how to delete a particular instruction that has been programmed into the robot pendant

Table A.1: Use Case 1

Task ID	**Use Case Task**	**Use Case Query**
UC 2.1	safety assurance	how to have a safety assurance before the robot is used
UC 2.2	CREATE A NEW JOB	how to create a new job
UC 2.3	CHOOSE THE RIGHT Coordinates	how to select the correct coordinates for the robot
UC 2.4	Main Initialization	steps to do the initialization of the rg6 before setting other parameters
UC 2.5	RESET GRIPPER	how to power reset the rg6 gripper
UC 2.6	CHOOSE THE RIGHT MOTION TYPE	how to choose the right motion type and change it
UC 2.7	Move to the safe area before picking up	how to move the robot to the correct position (over the object) before it can be picked up
UC 2.8	Set Function Call for pick	the call function for picking up the object using RG6
UC 2.9	set the first argument and understand the meaning	give step by step instructions on how to set the ARG(1):instance and what does it mean
UC 2.10	set the second argument and understand the meaning	how to choose the ARG(2):Width and what does it mean
UC 2.11	set the third argument and understand the meaning	step by step instructions on how to set the third argument wait and what does it mean
UC 2.12	set the fourth argument and understand the meaning	step by step instructions on how to set the fourth argument wait and what does it mean
UC 2.13	Move to the placement point	how to move to the placement point?
UC 2.14	Set Function Call for place	what is the call function for placing after picking for rg6
UC 2.15	set right argument for place	what is the function for placing after picking for rg6 and how to change the width argument
UC 2.16	Lift up	how to move the robot in an upwards direction after picking up the object
UC 2.17	Return to the starting position	how to move robot back to the starting point

Table A.2: Use Case 2

Task ID	**Use Case Task**	**Use Case Query**
UC 3.1	CREATE A NEW JOB	how to create a new job?
UC 3.2	CHANGE LD variables	how to change the local double precision variables in job header?
UC 3.3	UNDERSTAND THE MEANING OF ARG AND HOW TO SET ARG	what is the meaning of ARG and how to set ARG as an LD variable?
UC 3.4	UNDERSTAND THE MEANING of WHILEEXP loop AND HOW TO SET WHILEEXP loop	MEANING of WHILEEXP loop AND HOW TO perform the WHILEEXP command?
UC 3.5	understand Data Flow in Loops	what is data flow in loops?
UC 3.6	UNDERSTAND THE MEANING AND HOW TO SET "IFTHENEXP"	what is THE MEANING of IFTHENEXP AND HOW TO SET "IFTHENEXP"
UC 3.7	CREATE OR_SETU16	full instructions for OR_SETU16
UC 3.8	CREATE OR_SETCU16	full instructions for OR_SETCU16
UC 3.9	CREATE OR_INIT	commands for OR_INIT
UC 3.10	CREATE OR_RG_RESETPOWER	full instructions for OR_RG_RESETPOWER
UC 3.11	CREATE OR_RG_MOVE	commands for OR_RG_MOVE

Table A.3: Use Case 3

Task ID	**Use Case Task**	**Use Case Query**
UC 4.1	understand the meaning of TCP(why we need to set TCP after the change of gripper)	understand the meaning of TCP(why we need to set TCP after the change of gripper)?
UC 4.2	understand the Geometry of TCP	understand the Geometry of TCP
UC 4.3	understand the Physical Parameter of TCP	TCP physical parameters?
UC 4.4	understand the Tool Load of TCP	TCP tool load?
UC 4.5	change security mode	change security mode
UC 4.6	SET TCP	how to set tool load information of tcp for mg10
UC 4.7	TOOL CALIBRATION	steps for tool calibration
UC 4.8	CHECK TCP	CHECK TCP

Table A.4: Use Case 4

Task ID	Use Case Task	Use Case Query
UC 6.1	Select the appropriate mode to transmit the job	how to copy the whole job to an external source?
UC 6.2	Set Management Mode	Set Management Mode
UC 6.3	transfer external data to pendant(new jobs from MG10)	instructions on how to transfer external data to pendant(new jobs)?
UC 6.4	create new job	how to create new job?
UC 6.5	check position[R1] to reset position	robot to second home position?
UC 6.6	Main Initialization	steps to do the initialization of the mg10 before setting other parameters?
UC 6.7	SET ARGUMENT OF OR_INIT	what are the values in the table to set argument of OR_INIT
UC 6.8	CHOOSE THE RIGHT Coordinates	how to select the right coordinates?
UC 6.9	CHOOSE THE RIGHT MOTION TYPE	how to choose the right motion type and change?
UC 6.10	Move to the safe area before picking up.	how to move robot to the object?
UC 6.11	Set Function Call for pick	what is the call function for picking for mg10?
UC 6.12	SET ARGUMENT OF instance	how is ""ARG(1):Instance" retrieved
UC 6.13	SET ARGUMENT OF target_strength for pick	SET ARGUMENT OF target_strength for pick
UC 6.14	Move to the placement point	Move to the placement point
UC 6.15	Set Function Call to release	what is the call function for releasing for mg10?
UC 6.16	SET ARGUMENT OF target_strength for release	how to SET ARGUMENT OF target_strength for release
UC 6.17	Return to the starting position	steps to return robot to its initial position?

Table A.5: Use Case 6

Bibliography

[1] Amr Adel. «Future of industry 5.0 in society: human-centric solutions, challenges and prospective research areas». In: *Journal of Cloud Computing* 11.1 (2022). DOI: 10.1186/s13677-022-00314-5. URL: https://doi.org/10.1186/s13677-022-00314-5 (cit. on pp. 3, 10, 15).

[2] Hamed Fazlollahtabar. «Human-robot interaction using retrieval-augmented generation and fine-tuning with transformer neural networks in industry 5.0». In: *Scientific Reports* 15.1 (2025). DOI: 10.1038/s41598-025-12742-9. URL: https://doi.org/10.1038/s41598-025-12742-9 (cit. on pp. 3, 5, 10, 13, 15).

[3] Pantelis Karapanagiotis, Kolitha Kottagaha W.M., Diego Rovere, Jos A.C. Bokhorst, Andrea Valdata, and Christos Emmanouilidis. «Enabling interoperable human–AI teaming for automation in construction and manufacturing via Digital Twins and Sliding Work Sharing ontologies». In: *Journal of Industrial Information Integration* 48 (2025), p. 100962. DOI: 10.1016/j.jii.2025.100962. URL: https://doi.org/10.1016/j.jii.2025.100962 (cit. on pp. 3, 10, 12, 16).

[4] Miao Hong, Shebing Liu, Kuaiqin Zhao, and Lei Shi. «Modeling and Robust Control for Power Insurance Assessment SCARA Robots Using RAG Retrieval-Augmented Techniques». In: *2025 2nd International Conference on Smart Grid and Artificial Intelligence (SGAI)*. IEEE, 2025, pp. 14–19. DOI: 10.1109/sgai64825.2025.11009412. URL: https://doi.org/10.1109/sgai64825.2025.11009412 (cit. on pp. 3, 4, 10, 12, 16).

[5] Franci Rrapi, Beatrice Portelli, Giuseppe Serra, and Lorenzo Scalera. «From AI foundations to Large Language Models: A survey on challenges and opportunities in collaborative robotics». In: *Robotics and Computer-Integrated Manufacturing* 100 (2026), p. 103269. DOI: 10.1016/j.rcim.2026.103269. URL: https://doi.org/10.1016/j.rcim.2026.103269 (cit. on pp. 3, 4, 10, 15).

[6] Shanhe Lou, Yiran Zhang, Runjia Tan, and Chen Lv. «A human-cyber-physical system enabled sequential disassembly planning approach for a human-robot collaboration cell in Industry 5.0». In: *Robotics and Computer-Integrated Manufacturing* 87 (2024), p. 102706. DOI: 10.1016/j.rcim.2023.102706. URL: https://doi.org/10.1016/j.rcim.2023.102706 (cit. on pp. 3, 6, 10, 16).

[7] M. Arjomandi and T. Mukherjee. «A review of applications of collaborative robot in welding and additive manufacturing». In: *Robotics and Computer-Integrated Manufacturing* 100 (2026), p. 103256. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2026.103256. URL: https://www.sciencedirect.com/science/article/pii/S0736584526000359 (cit. on pp. 3, 10, 12, 14, 15).

[8] Sophie Berretta, Alina Tausch, Greta Ontrup, Björn Gilles, Corinna Peifer, and Annette Kluge. «Defining human-AI teaming the human-centered way: a scoping review and network analysis». In: *Frontiers in Artificial Intelligence* 6 (2023). DOI: 10.3389/frai.2023.1250725. URL: https://doi.org/10.3389/frai.2023.1250725 (cit. on pp. 4, 6, 10, 15).

[9] Qunlong Chen, Yuyi Zhang, Wei Qin, Shouchen Yao, Peixiang Wang, Peng Xu, and Songyun Wang. «Leveraging large and small model collaboration for advanced automation in smart manufacturing». In: *Robotics and Computer-Integrated Manufacturing* 101 (2026), p. 103313. DOI: 10.1016/j.rcim.2026.103313. URL: https://doi.org/10.1016/j.rcim.2026.103313 (cit. on pp. 4–6, 10, 12, 15).

[10] Tianyu Zhou, Lai Zou, Wenxi Wang, and Yun Huang. «Towards intelligent knowledge assistance in abrasive belt grinding via a retrieval-augmented generation chatbot with reliability support». In: *Computers in Industry* 178 (2026), p. 104479. DOI: 10.1016/j.compind.2026.104479. URL: https://doi.org/10.1016/j.compind.2026.104479 (cit. on pp. 4, 11, 13, 15).

[11] Haitao Lu, Dayu Zheng, and Luo Yang. «Mutual information-based retrieval-augmented generation for domain question answering». In: *Knowledge and Information Systems* 68.1 (2026). DOI: 10.1007/s10115-025-02624-x. URL: https://doi.org/10.1007/s10115-025-02624-x (cit. on pp. 4, 11, 13).

[12] Dongyu Ru et al. «RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation». In: *Advances in Neural Information Processing Systems 37*. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2024, pp. 21999–22027. DOI: 10.52202/079017-0692. URL: https://doi.org/10.52202/079017-0692 (cit. on pp. 5, 11, 25–30).

[13] Tian Wang, Junming Fan, and Pai Zheng. «An LLM-based vision and language cobot navigation approach for Human-centric Smart Manufacturing». In: *Journal of Manufacturing Systems* 75 (2024), pp. 299–305. DOI: 10.1016/j.jmsy.2024.04.020. URL: https://doi.org/10.1016/j.jmsy.2024.04.020 (cit. on pp. 5, 6, 11–13, 16).

[14] GeeksforGeeks. *What is Retrieval-Augmented Generation (RAG)?* https://www.geeksforgeeks.org/nlp/what-is-retrieval-augmented-generation-rag/. Accessed: 2026-06-17. 2024 (cit. on p. 5).

[15] T. Shi et al. «Knowledge-driven autonomous materials research via collaborative multi-agent and robotic system». In: *Cell Press* 9 (2 2026). DOI: https://doi.org/10.1016/j.matt.2025.102577. URL: https://www.cell.com/matter/abstract/S2590-2385(25)00620-4?_returnURL=https%3A%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS2590238525006204%3Fshowall%3Dtrue (cit. on pp. 6, 11, 14–16).

[16] Xingwei Zhou, Shijiang Li, Ying Liu, Wanran Yang, Chengyue Xiong, and Liang Hou. «LLM-based domain-specific knowledge graph construction and hybrid retrieval-augmented generation». In: *Journal of Manufacturing Systems* 86 (2026), pp. 1022–1038. DOI: 10.1016/j.jmsy.2026.04.023. URL: https://doi.org/10.1016/j.jmsy.2026.04.023 (cit. on pp. 6, 11, 13).

[17] Hangyu He, Xin Yuan, Kai Wu, and Wei Ni. «Federated Retrieval-Augmented Generation for Cybersecurity in Resource-Constrained IoT and Edge Environments: A Deployment-Oriented Scoping Review». In: *Electronics* 15.7 (2026), p. 1409. DOI: 10.3390/electronics15071409. URL: https://doi.org/10.3390/electronics15071409 (cit. on pp. 6, 11, 13, 15).

[18] Kai Chen and Min Hu. «An Automatic Video Tag Extraction Method based on Large Language Model Text Content Parsing». In: *2024 7th International Conference on Data Science and Information Technology (DSIT)*. IEEE, 2024, pp. 1–4. DOI: 10.1109/dsit61374.2024.10881284. URL: https://doi.org/10.1109/dsit61374.2024.10881284 (cit. on pp. 6, 11).

[19] Zhenyang Guo, Jin Cao, Wanying Ma, Qiulan Xu, Ben Niu, and Hui Li. «LLMQUA:Using LLM Automation to Enhance Risk Quantification in Data Processing for Big Data Platforms». In: *2025 11th IEEE International Conference on Privacy Computing and Data Security (PCDS)*. IEEE, 2025, pp. 345–353. DOI: 10.1109/pcds65695.2025.00054. URL: https://doi.org/10.1109/pcds65695.2025.00054 (cit. on pp. 6, 7, 11).

[20] Yidan Wang and Jiayin Wang. «BioWorkflow: Retrieving comprehensive bioinformatics workflows from publications». In: *Briefings in Bioinformatics* 26.6 (2025). DOI: 10.1093/bib/bbaf571. URL: https://doi.org/10.1093/bib/bbaf571 (cit. on pp. 6, 11).

[21] Rong Hu, Sen Liu, Panpan Qi, Jingyi Liu, and Fengyuan Li. «ICCA-RAG: Intelligent Customs Clearance Assistant Using Retrieval-Augmented Generation (RAG)». In: *IEEE Access* 13 (2025), pp. 39711–39726. DOI: 10.1109/access.2025.3544408. URL: https://doi.org/10.1109/access.2025.3544408 (cit. on pp. 6, 7, 11).

[22] Omkar Salunkhe, Clarissa A. González Chávez, Hao Wang, Anna Syberfeldt, David Romero, and Johan Stahre. *Developing Code Agents for Robot Programming: Technical and Managerial Perspectives*. Springer Nature Switzerland, 2025, pp. 134–147. DOI: 10.1007/978-3-032-03515-8_10. URL: https://doi.org/10.1007/978-3-032-03515-8_10 (cit. on p. 11).

[23] Bogdan Balog, Stelian Brad, Emilia Brad, Vasile-Dragos Bartos, Diana Ticudean, and Alexandru-Gabriel Cirlejan. «AI-Driven Robot Programming Optimization Using TRIZ Principles for Efficient and Seamless Deployment». In: *World Conference of AI-Powered Innovation and TRIZ Methodology*. Ed. by Denis Cavallucci, Stelian Brad, Pavel Livotov, and Rémy Houssin. Cham: Springer Nature Switzerland, 2026, pp. 69–81 (cit. on pp. 12, 14–16).

[24] Abdelrhman Bassiouny, Ahmed H. Elsayed, Zoe Falomir, and Angel P. del Pobil. «UJI-Butler: A Symbolic/Non-symbolic Robotic System that Learns Through Multi-modal Interaction». In: *International Journal of Social Robotics* 17.12 (2025), pp. 2883–2903. DOI: 10.1007/s12369-025-01234-5. URL: https://link.springer.com/article/10.1007/s12369-025-01234-5 (cit. on pp. 12, 15).

[25] T. Tharmilan, A. Henoshan, K. Garthigan, A.G. Buddhika, P. Jayasekara, V. Logeeshan, and Chathura Wanigasekara. «Enabling Conversational Intelligence in Service Robots Through Multimodal Fusion and Resource-Aware Generative Models». In: *2025 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2025, pp. 1087–1092. DOI: 10.1109/aiiot65859.2025.11105269. URL: https://doi.org/10.1109/aiiot65859.2025.11105269 (cit. on pp. 12, 15).

[26] Kabita Choudhary, Sheeba Uruj, Abhinav Pathak, V. Kalaichelvi, Sujala D. Shetty, R. Karthikeyan, Tarek Taha, and Rajkumar Muthusamy. «A Retrieval-Augmented Generation (RAG)-Based LLM for Modern Warehouse Automation and Management». In: *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*. IEEE, 2025, pp. 1688–1694. DOI: 10.1109/case58245.2025.11164106. URL: https://doi.org/10.1109/case58245.2025.11164106 (cit. on pp. 12, 16).

[27] Handong Zheng et al. «Multimodal OCR: Parse Anything from Documents». In: (2026). DOI: 10.2139/ssrn.7008277. URL: https://doi.org/10.2139/ssrn.7008277 (cit. on pp. 12, 21, 54).

[28] Handong Zheng et al. *dots.mocr: Multimodal OCR — Parse Anything from Documents*. `https://github.com/rednote-hilab/dots.mocr`. Accessed: 2026-06-29. 2026 (cit. on p. 21).

[29] Dongyu Ru et al. *RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation*. `https://github.com/amazon-science/RAGChecker`. Accessed: 2026-06-29. 2024 (cit. on p. 25).