INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL

ARMS: Augmented Reasoning Multi-Agent System

FRANCISCO LUÍS PEREIRA OLIVEIRA

outubro de 2025

P.PORTO

ARMS: Augmented Reasoning Multi-Agent System

Francisco Luís Pereira Oliveira

Student nº: 1170912

A dissertation submitted in partial fulfilment of the requirements for the degree of Master of Science, Specialization Area of Artificial Intelligence

Supervisor: Luís Gomes, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Co-Supervisor: Zita Vale, Full Professor, Institute of Engineering, Polytechnic of Porto

Evaluation Committee:

President:

Luís Manuel Silva Conceição, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Members:

Luiz Felipe Rocha de Faria, Associate Professor, Institute of Engineering, Polytechnic of Porto

Luís Filipe de Oliveira Gomes, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Porto, September 2025

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

DECLARAÇÃO DE INTEGRIDADE

DECLARAÇÃO DE INTEGRIDADE

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Declaro que o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

“Fraternidade com todas as dinâmicas!”

- Álvaro de Campos *in* “Ode Triunfal”

Resumo

Os avanços trazidos pela arquitetura transformer começaram uma revolução tecnológica que criou uma vasta gama de possíveis casos de uso, onde estes modelos são empregues como agentes responsáveis por lidar com uma diversidade de tarefas, desde chatbots até funções mais determinísticas como chamadas a APIs e controlo de sistemas informáticos. No entanto, devido à novidade destes modelos, existe uma ausência de normalização no desenvolvimento de implementações reais, devidamente controladas. Considera-se que o momento atual corresponde a uma fase de “alquimia” na utilização de modelos de “large-language models”, na qual surgem inovações quase diariamente. Estão também a ser feitos grandes investimentos nesta área, tornando este o melhor momento para dedicar esforços à descoberta e exploração dos limites e capacidades desta tecnologia do futuro.

Os sistemas multiagente pertencem a um domínio da inteligência artificial que tem vindo a ser desenvolvido ao longo de vários anos, resultando em arquiteturas, protocolos de comunicação e paradigmas de implementação refinados e maduros. No entanto, a sua implementação pode ser difícil devido ao esforço necessário para orquestrar protocolos de comunicação adequados, motores de decisão e a própria arquitetura dos agentes. Além disso, a comunicação agente-humano nem sempre é intuitiva, uma vez que a maioria dos agentes recorre a linguagens programáticas ou de máquina, que podem não ser fáceis de compreender por utilizadores não técnicos ou não contextualizados.

Esta dissertação propõe um sistema que visa fundir as capacidades dos modelos de linguagem de grande escala, nomeadamente a comunicação em linguagem natural e a capacidade de raciocinar sobre inputs, com as potencialidades dos sistemas multiagente distribuídos na resolução de tarefas em cenários industriais ou de edifícios inteligentes. Através da implementação de componentes de hardware específicos, abaixo designados por “Tools”, o sistema proposto procura aumentar o grau de impacto que as decisões tomadas pelos “large-language models” têm no meio que os rodeia. O sistema implementado e aqui proposto, denominado por “Augmented Reasoning Multi-Agent System” (ARMS), permite ainda que os utilizadores comuniquem diretamente com os agentes através de conversas em linguagem natural, facilitando a troca de informação e de intenções. A comunicação agente-agente é também objeto de investigação aprofundada, sendo controlada através de técnicas específicas que regulam o fluxo de informação e as interações orientadas aos objetivos.

Para além de uma revisão do estado da arte nos tópicos relacionados com a solução, que culmina numa discussão sobre agentes potenciados por “large-language models” versus agentes tradicionais, esta tese apresenta cinco casos de estudo que testam a solução proposta.

Palavras-Chave: Large-language models, Sistemas multiagente, Digitalização, Retrieval-Augmented Generation

Abstract

The developments brought by the transformer architecture have sparked a technological revolution that created a wide range of possible use cases where these models are employed as individuals responsible for handling a diverse array of tasks, from chatbots to more deterministic such as API call and control. However, due to the novelty of these models there has been a lack for standardization when developing proper, controlled real implementations. It is assumed that the present time is considered to be an alchemy-like stage of large-language model usage, and many different innovations are born almost every day and everywhere around the world. Great investments are also being made on the field, and there has never been better time to dedicate efforts into discovering and exploring the limits and capacities of this technology of the future.

Multi-agent systems belong to a domain of artificial intelligence that has been in the development for many years resulting in refined and mature architectures, communication protocols, and implementation paradigms. However, implementation might sometimes be difficult due to the overhead required in orchestrating proper communication protocols, decision engines, and agent architecture. Furthermore, agent-to-human communication is not always seamless since most agents have programmatic-machine language which might not be easy for actors that are not contextualized or are technically inclined to interact with.

This dissertation proposes a system that aims to fuse the capabilities of large-language models to communicate through natural language and rationalize inputs with the capabilities that distributed multi-agent systems offer to resolve tasks that might be present in industrial and smart-building scenarios. Moreover, through the implementation of specific pieces of hardware, referred below as tools, the proposed system tries to increase the degree of impact that decisions made by large-language models have in the environment around them. The system proposed, named “Augmented Reasoning Multi-Agent System” (ARMS), also allows users to communicate directly with agents through natural language conversations facilitating information and desire exchange. Agent-to-Agent communication is also deeply investigated and controlled using specific techniques to manage communication flow and objective-oriented exchanges.

Besides a review of the state-of-the-art on topics related to the solution that culminates in a discussion about large-language model-powered agents vs traditional agents, this thesis includes five different that test the solution: basic task delegation, interconnected agents, user registration system, vacation system, and building control. Each of these case studies were built incrementally, meaning that the most basic and core principles were firstly tested on the first use cases, culminating on a final one that integrated multiple components previously tested at a large scale. The results from the case studies demonstrated positive results in achieving a multi-agent system that can manipulate the world around it and establish human communication as needed, leveraging large-language models’ capabilities for the decision-making processes, as well as inter-connection.

Keywords: Large-language models, Multi-agent systems, Digitalization, Retrieval-Augmented Generation

Acknowledgements

I would like to thank all of those that never doubted my ability to complete this PDF file when it wasn't even a Word document. Fortunately, I've been lucky enough to be born in a family where this was true for almost everyone. Cooperativa was, and will forever be, also a part of that family.

I would also like to thank my supervisor Luis Gomes ("Luis", not "Luís") for helping me provide guidance with all the tips and tricks to finish this thesis, even though we didn't always agreed on many things.

I would like to thank Margarida for all her support and showing me that failure wasn't even in the equation.

Finally, I would like to thank those that understood the difference between science and deadlines, and innovation from quotas.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACL	Agent Communication Language
AI	Artificial Intelligence
ALPAC	Automatic Language Processing Advisory Committee
API	Application Programming Interface
AR	Augmented Reality
ARIMA	Autoregressive Integrated Moving Average
ARMS	Augmented Reasoning Multi-Agent System
BDI	Belief-Desire-Intent
BoW	Bag-of-Words
CFG	Context-Free Grammars
CVaR	Conditional VaR
DB	Database
FNN	Feed-Forward Network
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GECAD	Research Group on Intelligence Engineering and Computing for Advanced Innovation and Development
GloVe	Global Vectors for Word Representation
GPT	Generative Pre-trained Transformer
GPU	Graphical Processing Unit
HMM	Hidden Markov Model
HRC	Human-Robot Collaboration
HR	Human Resources
ICT	Information and Communication Technology
IoT	Internet of Things

IPP	Polytechnic Institute of Porto
KNN	K-Nearest Neighbours
LLM	Large-Language Model
LoRA	Low-Rank Adaptation
LSTM	Long Short-Term Memory
MA	Moving Average
MAS	Multi-Agent System
NLP	Natural Language Processing
PCB	Printed Circuit Boards
PCFG	Probabilistic Context-Free Grammar
POS	Part-of-Speech
RAG	Retrieval-Augmented Generation
RBF	Radial Basis Function
RNNs	Recurrent Neural Networks
RO	Research Objective
RQ	Research Question
RUL	Remaining Useful Life
SARIMA	Seasonal ARIMA
SME	Small and Medium-Sized Enterprise
SQL	Structured Query Language
SVMs	Support Vector Machines
TD-IDF	Term Frequency-Inverse Document Frequency
VAR	Vector Auto Regression
XMPP	Extensible Messaging and Presence Protocol

1. Introduction

This chapter provides an overview of the whole dissertation giving the reader with a top-level view of what was researched, developed, and how it is all structured under this document. A contextualization is provided to frame the scope of the thesis and define investigation boundaries. Research questions and objectives are described, helping to understand the motivation behind the work developed. This is followed by a summary on the scientific contributions that resulted from this thesis, finalizing with an overview of how the document is organized, facilitating reading navigation.

1.1 Contextualization

Multi-agent systems are capable of accomplishing complex tasks in ways that other development processes may struggle or simply aren't able to achieve due to physical and logical implications that would be necessary to be met to arrive at the same results (van der Hoek & Wooldridge, 2008). This field of Artificial Intelligence (AI) is particularly useful on solving tasks where knowledge is distributed and actions are not necessarily needed to be taken and decided by a central unit.

In these systems, the atomic unit is the agent, and the system is only as good as the complete organism that comprises each unit's cognitive capability, decision making capacity, information relay quality and power to act in the world around it is (JENNINGS & WOOLDRIDGE, 1995).

Many have been the proposals for multi-agent systems, with lots of variations, colours and tastes to pick from, but alas a new technology dawns and with it, new possibilities, new capabilities and new knowledge areas are possible to explore and discover. Since 2017, in "Attention is all you need" (Vaswani et al., 2023) and with the release of the ChatGPTs models, the world has been taken by storm with a new language processing technology: the transformer (Pang et al., 2025). To describe the impact of these models is no easy task to accomplish in this paragraph. It has been very noticeably paradigm shift, not just for technically inclined people or occasional users, but for everyday life of the general population as well. Search engines are being substituted by a more personalized experience tailored to questions without the need for scrolling and searching on results pages (Lindemann, 2023); debates on the whole method of evaluation of students in schools are being undertaken (Javaid et al., 2023); transformer models are helping people psychologically; industry is adopting these models for coding tasks; new companies are being created leveraging the customized and context aware knowledge provided by these models (Lau & Guo, n.d.).

One of the main driving motivations for the writing of this thesis is the exploration of the limitations and capabilities of these models to act together as a multi-agent system (MAS). By separating concerns, roles, abilities and responsibilities through different models it is possible to observe just how much a properly orchestrated group can achieve. The fusion of multi-agent system capabilities with large-language models (LLMs) as reasoning agents opens many doors to what is possible to achieve, when certain constraints and barriers, such as strictly defined communication rules and human-machine interactions are surpassed.

Making different LLMs communicate among themselves, is a task that has, in itself, many great opportunities to explore. This work tries to push that reality a bit further, by allowing LLMs to interact with the world around them, request more information when needed to complete certain tasks, communicate with legacy and already implemented systems, delegate tasks among themselves, self-organize, define roles, separate concerns and responsibilities.

This dissertation is an exploration of the fusion between LLMs and the MAS development paradigm, with a great focus on practical applications and giving LLMs the power to shape the world around them.

Under the development of this thesis an ecosystem named ARMS: Augmented Reasoning Multi-Agent System is crated taking the form of a MAS system that employs LLMs as a reason and communication engine. ARMS, grants the ability and a way for LLMs to act and interact with each other and what is around them using both text and modular tools such as SQL queries, IoT infrastructure control, and more importantly custom designed tools that can be implemented separately and used by LLMs as they see fit when presented with real-world problems.

ARMS can be used to perform the whole process of user registration on a system, to control an entire smart building using purely conversational inputs, it can leverage the LLMs capabilities of reasoning to help users of different technological backgrounds to interface with sometimes difficult to understand domains.

1.2 Research Questions and Objectives

To provide guidance and support to the developments, research and overall completion of this dissertation, a major research question (MRQ) was identified from which then multiple research questions were derived and were attempted to be answered throughout the advances of the thesis, whether through experimentation and testing or through investigations on work done by previous authors. This sub-chapter describes said questions which, in turn, resulted on the following research objectives:

Research Questions (RQs):

- MRQ: How can LLMs work together as a MAS?
    - RQ1: What are the current traditional MAS development paradigms? What is the progress of LLM fusing on said paradigms?
    - RQ2: Can a system that integrates LLMs as rational engines be considered a traditional MAS?
    - RQ3: What are the limitations of using LLMs as rational engines for agents on a MAS?
    - RQ4: How can LLMs act on the world around them?
    - RQ5: How can a MAS using LLMs as rational engines in its agents be orchestrated? What are the limitations of that orchestration?
- Derived from these RQs, research objectives (ROs) were established to help find conclusions and answers to the prominent questions.
- RO1: Investigate current MAS development paradigms through analysis of the state-of-the-art (related to RQ1);
- RO2: Investigate current progress on the usage of LLMs in a MAS-like setting (related do RQ1 and RQ2);
- RO3: Discuss and conclude if using LLMs together with traditional MAS techniques can be considered traditional MAS or a new nomenclature should be considered (related to RQ2);
    - RO4: Identify gaps on LLMs working as MAS (related to RQ1 and RQ3);
    - RO5: Identify limitations on LLMs working as MAS (related to RQ1 and RQ3);

- RO6: Conceptualize, create and develop a solution that attends to any existing gaps referred to in RO5;
- RO7: Gather knowledge on the limitations and potentials of the solution mentioned in RO6;
- RO8: Test the solution proposed in RO6.

1.3 Scientific Contributions

The present dissertation has been developed under the guidance, funding, and support of Research Group on Intelligence Engineering and Computing for Advanced Innovation and Development (GECAD). A research centre specialized in many different topics related to artificial intelligence.

This thesis contributed to the following project:

- PRODUTECH R3 - Agenda Mobilizadora da Fileira das Tecnologias de Produção para a Reindustrialização (01/C05-i11/2024.PC645808870-00000067)
- F4iTECH - Federated AI Platform for Industrial Technologies (C2021/1-10)
- F4iTECH - Federated AI Platform for Industrial Technologies (POCI-01-0247-FEDER-181419)

From the work done throughout this dissertation, 3 papers were originated and published in:

- Oliveira, F., Gomes, L., Vale, Z. (2025). Retrieval-Augmented Generation Powered by a Multi-agent System to Assisted the Operation of Industries. In: Mathieu, P., De la Prieta, F. (eds) Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection. PAAMS 2024. Lecture Notes in Computer Science(), vol 15157. Springer, Cham. https://doi.org/10.1007/978-3-031-70415-4_18
- Oliveira, F., Martins, R., Caetano, I., Gomes, L., Vale, Z. (2025) Centrifugal Governor-Inpired Predictive Maintenance Model for Legacy Machines: a METAL implementation case study. In Sustech 2025, (pending DOI).
- Oliveira, F., Ramalho, O., Gomes, L., Vale, Z. (2025) Retrieval Augmented Generation for Large Language Models to Support Industrial Operation. In Sustech 2025, (pending DOI).

1.4 Document Organization

The thesis presented in this document is organized in 6 different chapters:

Chapter 1 corresponds to the Introduction. This is the present chapter where a preface to the dissertation is given. Contextualization is provided to the reader motivating and helping to understand the work done from a “high-point perspective”. It is also on this section that the main research questions and consequent research objectives are defined. Scientific contributions are mentioned in the second-to-last sub-chapter and are followed by the current sub-chapter which offers an overview of the document organization.

In Chapter 2 the state-of-the-art is analysed, this is a rather large chapter that delves into the current developments on essentially 4 topics: Industry Digitalization, Artificial Intelligence in the Industry, Multi-agent systems in the industry, and Large Language Models. This chapter is finished with a discussion on the overall findings and identified gaps, together with a conclusion regarding RQ2.

Chapter 3 describes many different aspects related to the development of this dissertation, namely: Technologies used, ethical aspects, data protection and security, frameworks and resources.

Chapter 4 presents the focus of the development of this thesis: a multi-agent system powered by the capabilities of LLMs. It dives in-depth on its composition, how agents can use these revolutionary models as brains to power connections, decisions, beliefs and actions with and on the real world around them. The system named “ARMS”, is completely described in this chapter, and therefore a proposal for expanding the capabilities of the transformer architecture is given.

Chapter 5 is where ARMS is put together in different stages of development and scenarios to accomplish real results. This chapter corresponds to the Case Studies, and it is where the bridge between theory and practice is reached. There are 5 different case studies that test the proposed solution and shed light on the benefits, as well as the limitations of ARMS.

Chapter 6 is the final chapter where conclusions are drawn from the whole work done, research questions and objectives are recalled and contemplated. A discussion is elaborated to provide answers to these questions and to create a new ones for future work.

2. State of the Art

This chapter presents the state of the art of the thesis where a review of the current developments on different subject areas will be conducted. These knowledge domains are relevant to the development of the work as they aided the understanding of available gaps and possible innovations to be made taking as foundations what is currently available and solidified at present time. The approach will take into consideration industrial digitalization efforts, namely Industry 4.0, the more recent Industry 5.0 and the study and application of Internet of Things (IoTs) on the shopfloor and smart building environment. Furthermore, artificial intelligence developments in the industry sector will be analysed, such as clustering methods, pattern recognition, and risk-assessment, this is followed by a unit on multi-agent systems and finally culminating in a dedicated unit to the exploration of Natural Language Processing (NLP) developments and in-depth transformer architecture description.

Due to the broad topics intertwined with more specific ones, it was decided to organize this part of the thesis in three different sections by order of granularity: unit, chapter and sub-chapter, as demonstrated in Figure 1.

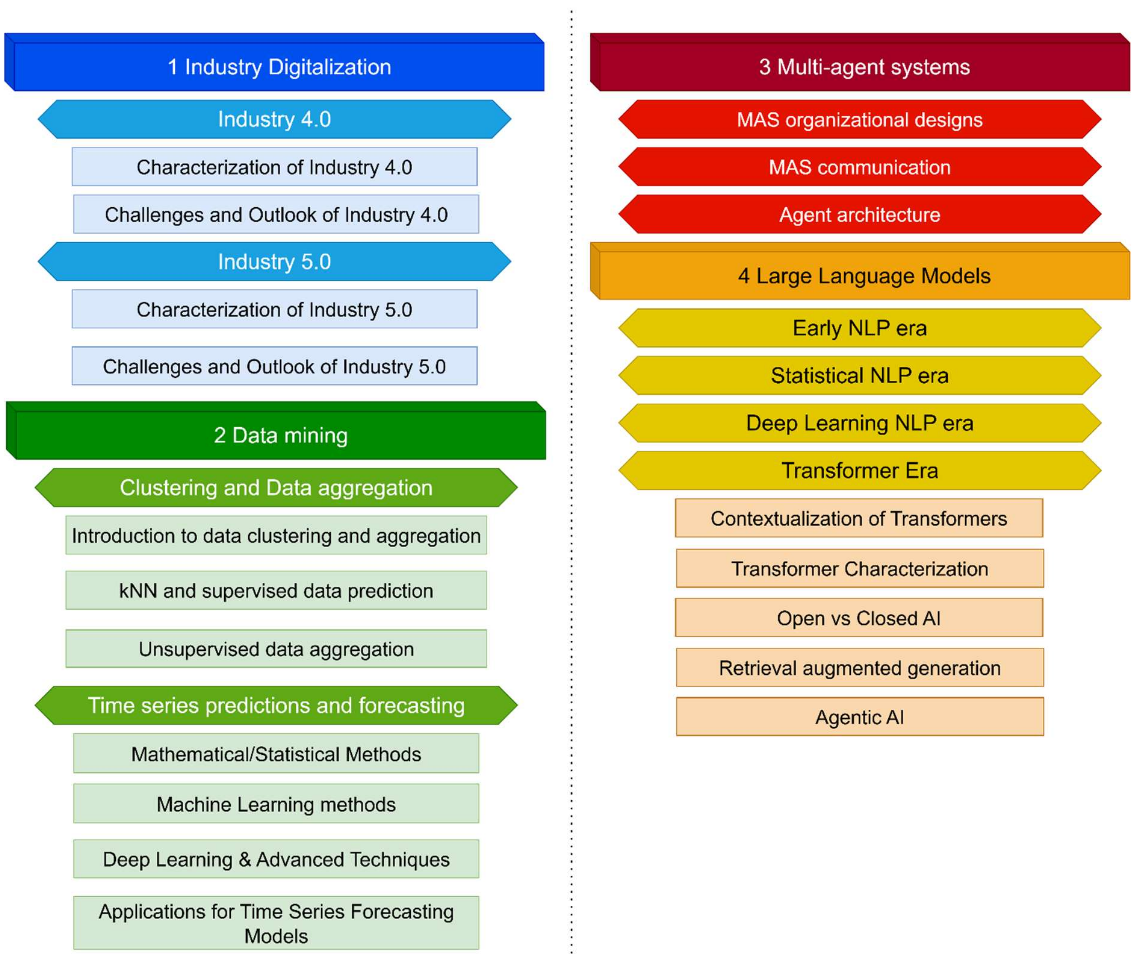

Figure 1 - State-of-the-art graphical index

2.1 Industry digitalization

The manufacturing of goods, commonly referred to as industrial production, started around the 18th century with the advent of the first industrial revolution (Gillispie, 1957). Since then, there have been four more industrial revolutions, totalling five (Leng et al., 2022), those being: the steam-powered industrial revolution, the electricity and mass production industrial revolution (Mokyr, 1998, pp. 1870–1914), the digital and automation industrial revolution (Mowery, 2009), and finally Industry 4.0 and the emerging Industry 5.0 (Leng et al., 2022). In this unit, the industries 4.0 and 5.0 are discussed, starting with Industry 4.0, centralized in digitalization and sensorization of the shopfloor, crucial to data generation and gathering, aiding engineers and managers to increase the efficiency and contextual knowledge on the production environment (Lasi et al., 2014).

The process of industrial digitalization encompasses the integration of IoTs, sensors, AI and cloud computing technologies with the manufacturing process by harnessing the value of these innovations and tools to increase productivity and understanding of, not just the assembly line, but of the whole organization and supply chain through a more informed and data-rich view of the production environment (Dalenogare et al., 2018).

2.1.1 Industry 4.0

According to (Culot et al., 2020) the beginning of Industry 4.0 started in 2011 in Germany during a presentation at the Hanover Fair (*HANNOVER MESSE*, n.d.) by a group from the Research Union Economy-Science of the German Ministry of Education and Research. This exposition had the intention of providing a solution that would allow Germany to strategize a way of strengthening its worldwide manufacturing relevance. This new paradigm has had, since then, the objective of revolutionizing the manufacturing process of a company all the way from the end consumer to the most atomic and simple factory machine process.

2.1.1.1. Characterization of Industry 4.0

Industry 4.0 is characterized by the implementation of various modern digital technologies in the manufacturing process of an organization; it was defined in (*Fourth Industrial Revolution - an Overview | ScienceDirect Topics*, n.d.) by *“Industry 4.0 refers to the transformation of industry through the intelligent networking of machines and processes with the help of information and communication technology (ICT)*.” The main characteristics of 4.0 which distinguish it from the previous industrial revolutions and that make it in fact a revolution are: increased automation and mechanization, miniaturization, and digitalization and networking (Culot et al., 2020).

The usage of modern ICTs contributes and has contributed to the increased efficiency of the assembly line. In 2012, a Danish university collaborated with toy manufacturer LEGO A/S to test and develop a modern specific assembly line, commonly referred to as “Chaku Chaku” line, represented on Figure 2.

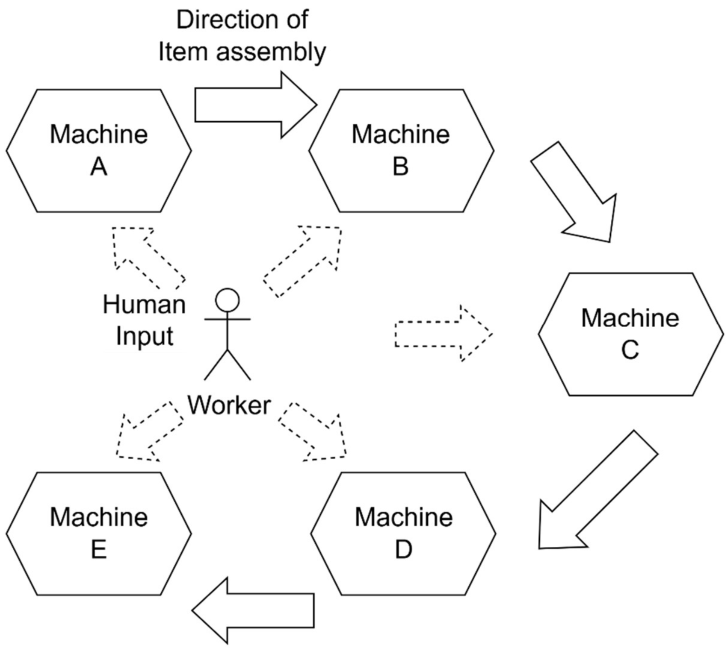

Figure 2 - Chaku-Chaku assembly line

On the project, a local order management system was deployed which shifted much of the planning manufacturing tasks from a purely automated system to a more hybrid approach integrating human-in-the-loop paradigm aided by IoT infrastructures. This practice allowed for a more connected production line where the skilled employees could communicate with suppliers and customers when pulling and receiving orders from a database. It was concluded that the “Chaku Chaku” line, a u-shaped production chain, equipped with machines becomes much more efficient and accurate, while having as drawbacks the exception handling of unforeseen events by the machine manufacturers, this is partially solved by integrating the employee which is able to solve the increasingly complexities of the modern production line as new and trending requests become more specific and harder to batch produce, and therefore demonstrating how industrial systems such as assembly lines can be incorporated with human-in-the-loop processes to enhance productivity (Bilberg & Hadar, 2012).

On a more recent study (Nourmohammadi et al., 2024), researchers explored the integration of human-robot collaboration (HRC) within Industry 4.0-enabled assembly lines, addressing the dual objectives of minimizing cycle time and reducing robot energy consumption. By developing a multi-objective optimization framework based on mixed-integer linear programming and the ε-constraint method (Mavrotas, 2009), the study identified optimal trade-offs between efficiency and sustainability. Applied to a case study in the automotive sector, the findings revealed how collaborative robots can handle repetitive tasks while humans manage complex operations, significantly enhancing productivity and energy efficiency. This approach underscores the transformative potential of Industry 4.0 technologies in advancing automation strategies, enabling decision-makers to align operational goals with energy sustainability.

Another example of Industry 4.0 potential is demonstrated in (Margherita & Braccini, 2021), a study where the adoption of technologies like collaborative robots (cobots), augmented reality (AR), and smart sensors helped balance capital efficiency and labour welfare. Conducted in an Italian manufacturing company, the study showed how automation doubled production, reduced defects, and cut energy consumption while workers moved from repetitive tasks to roles focused on supervision and maintenance, they also received tailored training, ensuring job stability. Despite these advancements, the study warned that without careful implementation, automation could worsen labour conditions by displacing workers or limiting their available roles.

An important aspect of Industry 4.0 is the transformation of manufacturing and logistics. While computers used to require a lot of space, today's smaller devices offer the same or even better performance in just a few cubic centimetres. This compact technology creates new opportunities, particularly in improving efficiency and innovation within production and logistics processes (Lasi et al., 2014).

Following the trend of miniaturization, another example of its potential is demonstrated in (Voges et al., 2019). The study highlights the development of miniaturized sensor nodes, embedded directly into production components like printed circuit boards (PCBs), measuring just 11x10x1.6 mm^3. These sensors allow for real-time data acquisition on factors such as temperature and acceleration throughout the product lifecycle, optimizing production processes. By integrating embedded security chips, the paper ensures data and system security. This miniaturization enables predictive maintenance and real-time monitoring, making it a great example of how compact technologies are revolutionizing smart manufacturing within Industry 4.0.

Another example of miniaturization's impact is demonstrated in (Amirian et al., 2024), where the authors demonstrate the role of miniaturized analytical devices integrated with IoT. These devices, such as lab-on-a-chip systems and wireless chemical sensors, enable real-time monitoring of environmental parameters like water pH, pollutants, and temperature. By employing IoT devices and cloud computing, these compact tools enable the efficient creation of data collection, transmission, and analysis while remaining cost-effective and portable. This integration not only supports sustainable practices by reducing material and energy consumption but also aligns with Industry 4.0 principles, advancing smarter, more sustainable monitoring systems for industries and agriculture.

2.1.1.2. Challenges and Outlook of Industry 4.0

However, how the “no free lunch” theorem states, there are drawbacks for certain developments, and Industry 4.0 is not immune to them (Arm et al., 2018). A 2022 paper (Bakshi & Paulson, 2022) conducts a study on the sustainability of the revolution and possible obstacles that it may face during its implementation and expansion. The study introduces some “commonly used methods for sustainability assessment and design”, those being: economic aspects, environmental aspects, and social aspects. Industry 4.0, similarly to past industrial revolutions, has as one of its main adversities the risk of unintentionally harming the environment, the paradigm implies that benefits at a very local level such as machines or factories have an impact on a very large economic and environmental level, since the usage of newly developed technologies often rely materials which are bound to create e-waste and are not always properly and possibly recycled.

Furthermore, a digitalization of processes has (similarly to other technological advances) the possibility to reduce and displace the number of available factory jobs as they become obsolete, which may provide a concern for encouraging such practices and advances (Kurt, 2019).

Although it is evident that the innovation in the field comes with many challenges, the outlook of Industry 4.0 is rather positive(Peres et al., 2020). Numerous are the applications and case studies available which enable for modernization and implementation of these recent technologies. Almost every factory can, in one way or another, take advantage of advancements such as predictive maintenance, asset tracking, inventory management, quality control, production process monitoring, energy efficiency and supply chain optimization which all belong to scope of this new revolution (Soori et al., 2023).

For innovation to take place, countries must implement comprehensive strategies that foster digital transformation, encourage investment in advanced technologies, and build a robust infrastructure to support Industry 4.0 (Teixeira & Tavares-Lehmann, 2022). This includes creating policies that incentivize the adoption of cyber-physical systems, IoT, and robotics in manufacturing processes while simultaneously enhancing access to high-speed broadband networks. Governments should prioritize education and training programs to develop a digitally skilled workforce capable of navigating and leveraging these emerging technologies. Additionally, fostering collaboration between public and private sectors, supporting startups and Small and medium-sized enterprises (SME)s through grants or tax breaks, and promoting research and development initiatives are essential to driving innovation. By addressing gaps in regulatory frameworks, ensuring data security, and encouraging cross-border cooperation, countries can accelerate the implementation of Industry 4.0 and position themselves as leaders in the Fourth Industrial Revolution (Bilgen, 2021; Teixeira & Tavares-Lehmann, 2022).

A 2023 study (Pereira et al., 2023) concluded that Industry 4.0 development in Portugal is gaining momentum, with 72% of companies already initiating implementation and 89% planning to invest in digital transformation. Despite this progress, the adoption remains fragmented, as only 50% of companies align their digital strategies with broader organizational goals. Technologies like IoT, Big Data, and Cloud Computing dominate Portuguese industries, while the use of autonomous robots, augmented reality, and simulation is growing steadily. However, that same study found that barriers such as limited budgets, high initial investments, and a lack of skilled human resources persist, with 56% of companies identifying the need for reskilling. Compared to global trends, Portuguese companies exhibit optimism, with 86% aspiring to achieve high levels of digitalization within five years, surpassing the global expectation of 72%. Additionally on that study, it was found that Portuguese companies anticipate an average revenue increase of 10% and efficiency gains of over 10% in 70% of cases, reflecting a higher-than-average ambition. However, concerns about cybersecurity and gaps in integrating the full value chain, seen in 60% of firms, suggest that Portugal still lags behind world leaders like Germany and Denmark, where integrated strategies and advanced maturity models are widely used. Despite these challenges, initiatives like the "Portugal i4.0" program aim to close these gaps, making Portugal a promising fast follower in the global Industry 4.0 race.

2.1.2 Industry 5.0

The baseline technologies and frameworks for Industry 4.0 have matured and are now ready for modelled application in factories and organizations which do not yet have implemented the technologies (Schumacher et al., 2016). On the background however, academic and industrial development suggests that a new revolution is brewing, named in succession to Industry 4.0, and with a Google Trends 0 relative interest points in august 2016 and 100 in august 2024, discussion about “Industry 5.0 revolution” has been on the rise (Akundi et al., 2022; Rame et al., 2024).

2.1.2.1. Characterization of Industry 5.0

Similarly to its predecessor, Industry 5.0 has origins in Europe, during two workshops in the year of 2020, and is fundamentally described in (*Industry 5.0*, n.d.) with a vision of sustainability and industrial competitiveness. The movement focuses on human-centric solutions, human-machine-interactions, bio-inspired technologies and smart materials that enable embedded sensors and better, more sustainable recycling. Priorities of the revolution also consist of the creation of digital twins for entire systems, energy efficient technologies, AI integration and safe data transfers (*Industry 5.0*, n.d.).

The four industrial revolutions have each been driven by transformative technologies that defined their time. The First Industrial Revolution harnessed water and steam power to enable mechanization (Weightman, 2010, pp. 1776–1914), while the Second Industrial Revolution brought electrification, the division of labour, and mass production, revolutionizing manufacturing processes (Mokyr, 1998). The Third Industrial Revolution introduced information technology, electronics, and automation, marking the shift toward digital systems (Mowery, 2009). Industry 4.0 focuses on integrating advanced technologies such as cyber-physical systems, the Internet of Things (IoT), and artificial intelligence to create interconnected, intelligent, and highly efficient industrial ecosystems (Teixeira & Tavares-Lehmann, 2022). Although some of the concepts between the two are shared, Industry 5.0 emphasizes the human and ecological dimensions of technological progress rather than focusing solely on automation and efficiency (Leng et al., 2022).

Industry 5.0 prioritizes integrating socially and environmentally sustainable values, aiming to harmonize technology with human well-being and ecological preservation. This marks a significant departure from Industry 4.0's focus on technological connectivity to Industry 5.0's focus on societal and environmental impact (X. Xu et al., 2021). Furthermore, Industry 5.0 should not be viewed as a replacement or alternative, but a logical evolution and continuation of the former ones.

The authors of (Leng et al., 2022) attributed to the global COVID-19 epidemic the main reason for a slow Industry 5.0 adoption, as economies went into recession and industries needed a swift recovery, spending less resources in innovation. Nonetheless, the 2022 study underlines different developments that were made in various fields.

In (Maddikunta et al., 2022), significantly contributions to the development of smart manufacturing by emphasizing the integration of human creativity with intelligent systems are made, a key point of Industry 5.0. It explores smart additive manufacturing (SAM) as a sustainable approach that uses advanced technologies like AI, computer vision, and 3D printing to enhance precision, minimize waste, and improve resource efficiency. The paper shows the transition from Industry 4.0's focus on mass production to Industry 5.0's hyper-customization, enabled by collaborative robots (cobots), cognitive systems, and predictive maintenance tools. Additionally, it dives deep into enabling technologies such as edge computing, digital twins, and blockchain, which collectively optimize real-time monitoring, streamline supply chains, and improve decision-making in manufacturing environments. By addressing critical challenges like human-robot collaboration, scalability, and privacy, the paper sets a foundation for smart manufacturing systems that are sustainable and human-centric, aligning with the goals and values of Industry 5.0.

As previously mentioned, Blockchain technology is also an enabler technology of the core Industry 5.0 innovations. (Leng et al., 2023) demonstrates how it can be applied to track manufacturing tasks, ensure secure peer-to-peer negotiations, and enable tamper-proof coordination between distributed agents in individualized manufacturing environments. Further implementations of the Industry 5.0 concepts can be found on the shipping industry (Fraga-

Lamas et al., 2021) and on the education field (Carayannis & Morawska-Jancelewicz, 2022; Fraga-Lamas et al., 2021).

2.1.2.2. Challenges and Outlook of Industry 5.0

Industry 5.0 brings exciting possibilities, but it also comes with significant challenges and obstacles (Kovari, 2024). One big issue is the lack of research and clear direction (Leng et al., 2022). Since Industry 5.0 is a relatively new idea, there aren't many reliable studies or guidelines available. Researchers don't always agree on how it should evolve, what technologies to use, or how to train workers for the technology shift (Carayannis & Morawska-Jancelewicz, 2022). This uncertainty makes it harder for businesses and industries to figure out how to move forward in a cohesive and effective way.

Another challenge lies in how society adapts to such a major shift. Industry 5.0 focuses on human-centric and value-driven production, but that disrupts traditional systems like industrial supply chains, which could lead to overproduction and imbalances between supply and demand (Leng et al., 2022). Besides, existing regulations and oversight systems aren't ready to handle the complexity of massive data flows, or the decentralized nature of Industry 5.0 (Leng et al., 2022). Social acceptance also takes time, as people may struggle with ethical concerns or changes in work dynamics (Kemendi et al., 2022). Even environmental systems could be at risk since the change and not-always worldwide effort for sustainability requires big investment and resources, potentially causing complications with development (Kovari, 2024).

Different systems and devices often don't integrate and communicate properly with each other, creating problems in establishing interconnected processes (Kovari, 2024). Privacy and data security are also concerns that have to ever more be taken into consideration, as human-centric operations generate massive amounts of sensitive and personal information (Chander et al., 2022). Workers will need continuous training to keep up with the demands of Industry 5.0, which potentially requires a mix of different technical knowledge. All these problems demonstrate that while Industry 5.0's potential has been identified, progress will require careful planning and problem-solving (Kemendi et al., 2022).

2.2 Data mining

As mentioned on the previous unit, the advancements of digitalization in the industrial sector are shifting the focus from traditional manufacturing processes to the implementation of new technologies directly on the shop floor. Industry 4.0 and Industry 5.0 principles are core values driving the modernization of factories (Kemendi et al., 2022). Customer-driven, highly customizable manufacturing, smart sensorization and predictive maintenance are all benefits that come from this new interconnected environment made possible by the integration of IoT devices with newly developed, and older retrofitted machinery (Keshav Kolla et al., 2022). Furthermore, the simulation of factory components (such as general or specific assembly line sectors) through the creation of digital twins and human-centred systems allows for a more abstract and broad view of the factory's domain(Batty, 2018).

If steam and coal were the fuel behind the first industrial revolution, IoTs and the power of internet connectivity would be the fuel that powers modern industrial revolutions (M. Khan et al., 2017). Smart factories take advantage of modern equipment to monitor health, energy consumption, worker safety and well-being, and overall increases in product quality (Mabkhot et al., 2018).

Despite these advancements, the full potential of IoT still remains, to some degree, underutilized due to concerns surrounding implementation, integration, and retrofitting of existing (and sometimes old) equipment (Peres et al., 2020). This apparent struggle to properly utilize and recognize IoT value and data generated by devices, creates potential threats to industries, namely: Data overload, as massive amounts of sometimes unstructured data become overwhelming for in-house databases without proper analysis and treatment tools; security risks, related to the newly created digital endpoints within the system that may pose a cyber threat to the company's data if not handled and protected accordingly; and finally, the usage of modern and cutting edge technologies usually comes at the expense of an existing lack of skilled work force which results in additional spending and higher costs that organizations are not always willing to invest (Peres et al., 2020).

Although these may represent roadblocks for the proper integration and development of Industry 4.0 and 5.0 in a factory, some organizations have already successfully implemented these systems on their production environments and are now reaping the benefits, including: predictive maintenance on large machines with different degrees of granularity and precision (Oliveira et al., 2023); anomaly detection in the production environment (Abdelrahman & Keikhosrokiani, 2020); worker stress and well-being observability (Ifenthaler & Seufert, 2022); operational efficiency, since real-time monitoring allows for better resource allocation and process optimization (Mateo & Redchuk, 2024); reduced material waste and energy spending through IoT-driven management therefore reducing environmental impact (Bányai et al., 2019).

Some studies (Jan et al., 2023; Mateo & Redchuk, 2024) argue that although IoT integration represents the foundations of smart manufacturing, artificial intelligence methodologies hold the key to unlocking its true potential, combining methods that in the past may have been difficult to implement due to the lack of meaningful data resources required to fuel these algorithms, and overall available computing power, but are now completely feasible due to the great advancements made in the development of processing units which can work on large data with parallelization capabilities known as: "graphical processing units" (GPUs). This powerful combination allows for the implementation of various techniques that improve the modern manufacturing process (Y. Lu, 2019).

Despite the ample benefits brought by Industry AI, an established pattern of implementation is yet to be defined, according to a 2020 survey (Peres et al., 2020), most Industry AI design principles are:

Table 1 - AI design principles

Design Principle:	Adoption Rate:	Design Intent
Real-time Capability	Very High	Enable AI-driven decisions and actions in real-time to optimize manufacturing processes and meet operational constraints.
Human in the loop	Very High	Ensure AI systems incorporate human expertise for validation, supervision, and continuous improvement.
Robustness	High	Develop AI applications resilient to environmental and data variations for stable performance.

Design Principle:	Adoption Rate:	Design Intent
Interoperability	High	Facilitate seamless data exchange and communication across different industrial systems and standards.
Decentralization	High	Distribute AI processing across edge, fog, and cloud layers for efficient and scalable decision-making.
Service Orientation	Fair	Implement AI as modular, service-based components for flexibility and easy integration into existing systems.
Modularity	Fair	Design AI solutions as adaptable building blocks to evolve with industrial needs.
Context Awareness	Fair	Enable AI to interpret contextual information for improved decision-making in dynamic environments.
Scalability	Low	Ensure AI solutions can expand efficiently as data and system complexity grow.
Continuous Engineering	Low	Maintain and improve AI models throughout their lifecycle for sustained accuracy and reliability.
Interpretability	Very Low	Enhance AI transparency to improve trust, troubleshooting, and usability for stakeholders.
Cybersecurity	Very Low	Protect AI-driven industrial systems from cyber threats through robust security measures.

2.2.1 Clustering and Data aggregation

The following sections provide a comprehensive analysis of various methodologies and advancements in the field of AI, highlighting key developments that have shaped its evolution. Additionally, the discussion explores a range of technologies and strategic approaches that can be effectively utilized in the realm of Industrial AI. This includes cutting-edge innovations, optimization techniques, and data-driven solutions that enhance industrial processes, improve automation, and drive efficiency across various sectors. By examining these methodologies and technologies, the following sections aim to offer valuable insights into the potential applications and future directions of AI within industrial settings.

2.2.1.1. Introduction to data clustering and aggregation

Clustering is a technique in machine learning that groups similar data points based on certain specific features (R. Xu & Wunsch, 2005). It's mostly an unsupervised learning method, meaning it works without labelled data, and can discover patterns and structures on its own (data mining). A clustering technique which, however, does not fall on the category of unsupervised learning and is worth mentioning is: K-Nearest Neighbours (KNN), sometimes mistaken for clustering since it groups similar points, however this methodology requires labelled data to make predictions, making it a classification algorithm (Guo et al., 2003).

Clustering is everywhere in our digital daily life, from spam email filtering (Sasaki & Shinnou, 2005) and recommendation systems (Gulzar et al., 2023) to customer segmentation for marketing and image organization based on visual similarities. It's not just for text-based data either clustering can be used on audio, video, and imaging (Yang et al., 2016). For instance, audio clustering groups songs based on acoustic features, which helps with things like genre identification. Different algorithms, like K-Means, hierarchical clustering, and DBSCAN, each have their own way of dividing data and retrieving results. Clustering is a powerful tool for uncovering hidden patterns, making sense of large datasets, and improving decision-making in various industries (D. Xu & Tian, 2015).

2.2.1.2. kNN and supervised data prediction

The k-Nearest Neighbours (kNN) algorithm is a simple, yet powerful classification method widely used in machine learning. It operates on the principle of similarity, classifying new data points as they are inputted based on their proximity to existing and previously labelled data. The algorithm stores the entire training dataset and, when making the next prediction, it identifies the k closest training examples using a chosen distance metric, such as Euclidean distance ($Distance = \sqrt{(PointA - PointB)^2}$) (Guo et al., 2003).

The classification decision is made by majority voting among these nearest neighbours. Despite its effectiveness, kNN suffers from two main drawbacks: high computational cost and sensitivity to the choice of k. Since kNN stores the training data without building a predictive model in advance and postpones computations until the classification phase, it becomes inefficient for handling large datasets. Moreover, the selection of an appropriate k value is crucial, as a small k may lead to overfitting while a large k can cause underfitting. Various enhancements have been proposed to improve kNN, such as using data reduction techniques like Principal Component Analysis (PCA)(Borman et al., 2021; V. Gupta & Mittal, 2018) to minimize redundancy and reduce the dimensionality of the dataset, optimizing k dynamically, and constructing model-based variations that group similar instances. These refinements address efficiency concerns and enhance classification performance, making kNN adaptable for various domains, including text classification (Shah et al., 2020), image recognition (Fan et al., 2021), and medical diagnosis (Guo et al., 2003). Figure 3 explains, step-by-step, how the algorithm works to achieve new member classification.

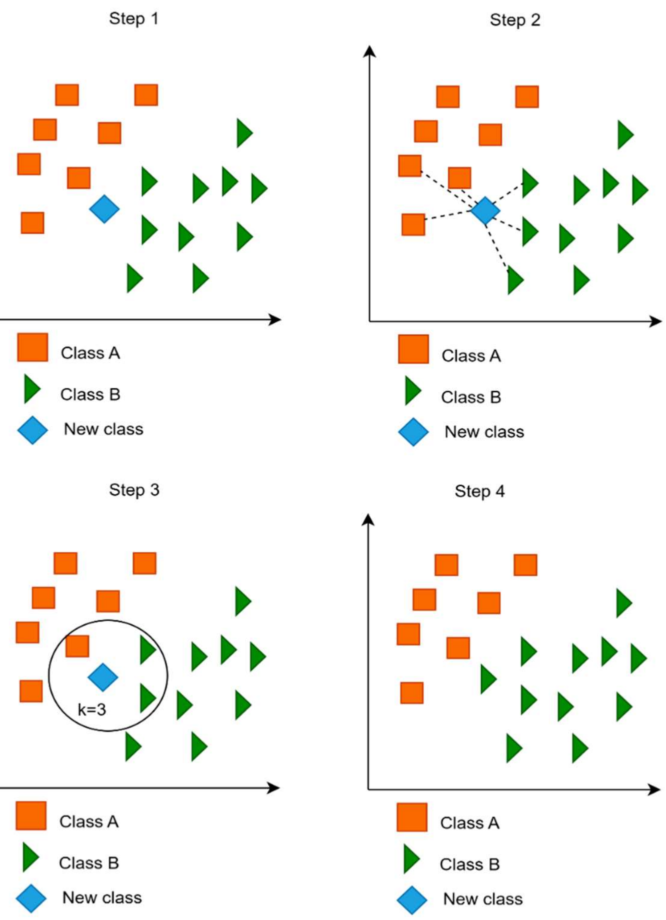

Figure 3 - kNN classification step-by-step

Fig X - Steps of a kNN algorithm with k = 3.

1. A new class is introduced in an environment where two other classes are already known and labelled (Class A, and Class B)
2. The distance between the new class and the closest labelled classes is calculated
3. The k values (with k being defined *a priori*) are selected
4. The new class is attributed to the class most representative of the selected ones, in this case, the new instance would be attributed to class B.

2.2.1.3. Unsupervised data aggregation

Although kNN is a great algorithm for clustering previously labelled data, most clustering techniques focus on unsupervised learning with the aim to extract relevant data, also known as “data mining”, from datasets (Berkhin, 2006).

Traditional clustering algorithms can be broadly classified into: hierarchical clustering, and partitional clustering, based on how clusters are generated (Aggarwal & Reddy, 2018). Hierarchical clustering forms a nested sequence of partitions, either by merging smaller clusters into larger ones or by successively dividing a large cluster into smaller ones (Ran et al., 2023). In contrast, partitional clustering directly partitions data into a predefined number of clusters without a hierarchical structure (Aggarwal & Reddy, 2018).

There are, however, other less popular methodologies, namely: using fuzzy logic, which denies the implication that a cluster is either a class or not and blends classes amongst each other, resulting in instances where we might, for example, have class that is 90% belonging to class A and 10% belonging to class B (Ruspini et al., 2019); using algorithms based on distribution which assumes that the data is generated from a mixture of probabilistic distributions, and clustering is performed by estimating the underlying probability distribution for each cluster (Ben-Israel & Iyigun, 2008); using density-based methods, that identify clusters by analysing patterns of datapoint density instead of grouping classes by a predefined number of clusters (k) (Kriegel et al., 2011); Graph-based clustering represents a dataset as a graph, where nodes correspond to individual data points and edges represent relationships (e.g., similarity or distance) between them (Tsitsulin et al., 2023).

Table 2 summarizes these techniques, providing examples of more specific algorithms based on each category. Where n = number of data points, k = number of clusters, t = number of iterations, s = subset of points, $f(v, e)$ = function used to detect clusters with v = vertices and e = edges, for example: Bron-Kerbosch algorithm ($O(3^{v/3})$), Naïve brute-force clique detection $O(2^v)$.

Table 2 - Data clustering methodologies and applications

Category	Algorithm	Time complexity	Computational cost	Examples of Applications
Partition	K-means	$O(knt)$	Low	Steel manufacturing process optimization (Park et al., 2020), factory energy optimization (Sung & Cho, 2019), image segmentation and clustering (Dehariya et al., 2010)
	K-medoids	$O(k(n-k)^2)$	High	Image analysis (W. Lu & Wu, 2012)
	Partitioning Around Medoids (PAM)	$O(k^3 * n^2)$	High	Defining clusters in sparse data (Schubert & Rousseeuw, 2021)
	Clustering Large Applications (CLARA)	$O(k * s^2 + k(n-k))$	Medium	Image analysis (T. Gupta & Panda, 2019)
	Clustering Large Applications based on Random Search (CLARANS)	$O(n^2)$	High	Data mining (Ng & Han, 2002)
Hierarchy	Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)	$O(n)$	Low	Data compression, Image processing (T. Zhang et al., 1997)
	Clustering Using Representatives (CURE)	$O(s^2 * 2)$	Low	Video Summarization (Majumdar et al., 2019)
	Robust Clustering using Links (ROCK)	$O(n^2 * logn)$	High	Clustering non-numerical data in databases (Guha et al., 2000)
	Chameleon	$O(n^2)$	High	Recommender Systems (U. Gupta & Patil, 2015)
Fuzzy	Fuzzy C-Means (FCM)	$O(n)$	Low	Molecular study (Li et al., 2009)
	Fuzzy C-Shell (FCS)	(depends on the kernel used)	(depends on the kernel used)	Data mining (Pratiwi & Saputro, 2020)
	Mixture Model (MM)	$O(v^2 * n)$	Medium	Ensemble technique (Topchy et al., 2004)
Distribution	Density-Based Spatial Clustering of Applications with Noise (DBCLASD)	$O(n * logn)$	Medium	Dimension reduction (X. Xu et al., 1998)
	Gaussian Mixture Model (GMM)	$O(n^2 * kt)$	High	Classification (Wan et al., 2019)
Density	Density-Based Spatial Clustering of Applications with Noise (DBSCAN)	$O(n * logn)$	Medium	Data extraction from GIS, satellite images, X-ray crystallography, remote sensing (K. Khan et al., 2014)
	Ordering Points To Identify the Clustering Structure (OPTICS)	$O(n * logn)$	Medium	Data mining on temporal datasets (Agrawal et al., 2016)
	Mean-shift	(depends on the kernel used)	(depends on the kernel used)	Image segmentation (K.-L. Wu & Yang, 2007)
Graph-Based	Cluster Identification via Connectivity Kernels (CLICK)	$O(k * f(v,e))$	Low	Gene expression analysis (Sharan & Shamir, 2000)
	Minimum Spanning Tree (MST)	$O(e \log v)$	Medium	Exploratory analysis of complex datasets (A. A. Khan & Mohanty, 2022)

Clustering methods such as K-Means require the predefined selection of the number of clusters (k) before execution (Dehariya et al., 2010). However, determining an appropriate value for k can be challenging, particularly when dealing with complex algorithms and large datasets, where a trial-and-error approach may be inefficient (Fahim, 2021). To address this issue, various methodologies have been developed to assist in selecting the optimal number of clusters.

Some approaches, such as the "Rule of Thumb" ($k = \sqrt{n/2}$, where n = number of datapoints), provide a simple heuristic estimation (Kodinariya & Makwana, 2013), while the Elbow Method offers a graphical technique to identify the point at which adding more clusters no longer significantly improves the clustering performance. More advanced statistical methods, including Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC), aim to balance model complexity with accuracy. The Information Theoretic Approach, which includes the Jump Method, evaluates the degree of separation between clusters, whereas the Silhouette Method assesses clustering quality by comparing intra-cluster cohesion with inter-cluster separation. Additionally, Cross-validation ensures clustering stability by partitioning the dataset and evaluating consistency across different subsets (Kodinariya & Makwana, 2013). By applying these methods, the selection of "k" can be made more systematic and data-driven, improving clustering accuracy and reliability. Figure 4 presents a graphical representation of the elbow selection methodology.

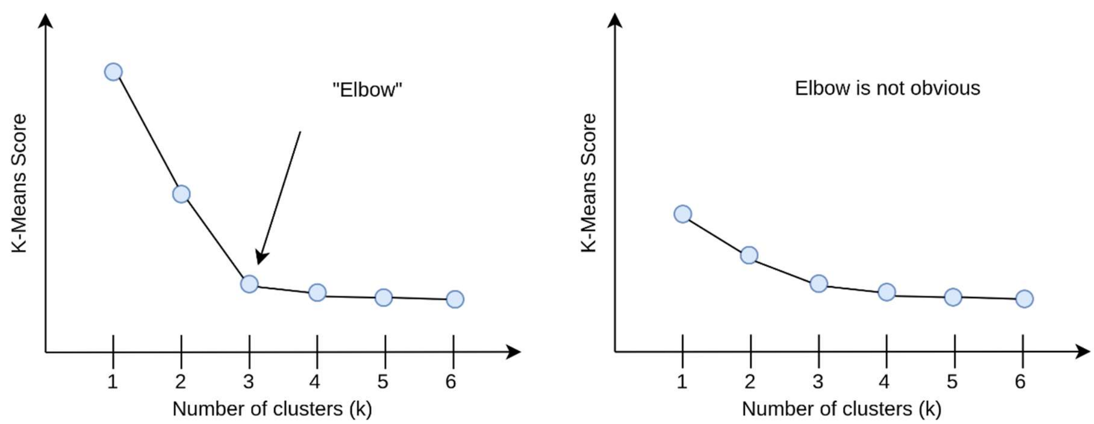

Figure 4 - Visual Representation of the "Elbow method"

2.2.2 Time series predictions and forecasting

Time series prediction and risk assessment are fundamental to different scientific and industrial applications, from financial forecasting and climate modelling to healthcare and cybersecurity (De Gooijer & Hyndman, 2006; Hamilton, 2020; Kirchgässner et al., 2012). Time series data often exhibit complex patterns influenced by seasonality, trends, and stochastic variations (Davey & Flores, 1993). Predicting future values from historical data is crucial for decision-making, resource optimization, and risk mitigation (Hamilton, 2020).

Traditional approaches to time series prediction, such as Autoregressive Integrated Moving Average (ARIMA) and Vector Auto Regression (VAR), have been widely used for decades due to their interpretability and robustness in handling linear dependencies (Siami-Namini et al., 2018). However, with the rapid increase in data complexity and dimensionality, classical statistical models often struggle to capture deep latent patterns and nonlinear relationships (Siami-Namini et al., 2018). This has led to the rise of innovative machine learning and deep learning techniques, which use complex architectures to learn from large amounts of historical data (Ismail Fawaz et al., 2019). Some popular models such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and transformer-based architectures have demonstrated significant improvements in predictive accuracy, particularly in high-dimensional datasets (B. Lim & Zohren, 2021).

Risk assessment, closely tied to time series forecasting, involves quantifying the probability and impact of potential adverse events, or estimating a Remaining Useful Life (RUL) number for a given study object (Oliveira et al., 2023). In financial markets, methods such as Value at Risk (VaR) and Conditional VaR (CVaR) are used to estimate potential losses under extreme conditions (Manganelli & Engle, 2001). In industrial applications, Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models are employed to measure volatility (Hassan & Malik, 2007). More recently, deep learning-based anomaly detection techniques, such as Autoencoders and Isolation Forests, have been applied to detect early warning signals in risk-sensitive environments like fraud detection and predictive maintenance (Z. Chen et al., 2018).

2.2.2.1. Mathematical/Statistical methods

Early risk assessment and time series prediction models are rooted in mathematical and statistical methodologies (Fuller, 1995). These techniques are often dependent on linear correlations and are more effective in predicting patterns on smaller time intervals (Nath et al., 2021). Linear and exponential smoothing methods were some of the most basic used approaches for time series forecasting, mainly because they are simple and effective for capturing trends and reducing noise. Linear smoothing methods, such as moving averages, calculate the average of the last few observations to smooth out fluctuations, making them useful for short-term forecasting, though they lag when the data changes rapidly (Letchford et al., 2012).

Exponential smoothing improves on this by giving more weight to recent values while still considering historical data, allowing it to adapt better to changes (GARDNER, 1985). Simple exponential smoothing works well when there's no trend. Holt-Winters adds a notion of trend and a seasonal adjustment (referring to the occurrence of repeating patterns in data-cycles, for example: increased sales in Christmas time), making it useful for data that follows repeating patterns (Ferbar Tratar & Strmčnik, 2016). These methods are computationally efficient and work well for stable and predictable series, but they struggle with highly volatile, irregular, or nonlinear data, which often requires more advanced machine learning approaches to improve accuracy (Prestwich et al., 2014).

ARIMA is another commonly traditional model for time series forecasting, especially when dealing with data that shows trends but no clear seasonality (Siami-Namini et al., 2018). It combines three components: AutoRegression (AR), which predicts future values based on past observations; Integration (I), which makes the series stationary (meaning that error values are constant and don't fluctuate through the series) by differencing; and Moving Average (MA), which smooths out noise by modelling the relationship between past forecast errors (Albeladi et al., 2023). ARIMA works well for capturing linear dependencies in time series and is specifically effective for short-term forecasting when the underlying patterns are stable. However, it struggles with complex, non-linear relationships and seasonal variations unless extended to Seasonal ARIMA (SARIMA), which includes seasonal adjustments (Albeladi et al., 2023). Despite being a strong statistical method, ARIMA requires careful tuning of parameters and assumptions about stationarity, making it less flexible compared to modern machine learning techniques that can automatically learn patterns from raw data (Albeladi et al., 2023).

The vector autoregression (VAR) model is one of the most successful, flexible, and easy to use models for the analysis of multivariate time series. It is a natural extension of the univariate autoregressive model to dynamic multivariate time series. The VAR model has proven to be especially useful for describing the dynamic behaviour of economic and financial time series and for forecasting. It often provides superior forecasts to those from univariate time series models and elaborate theory-based simultaneous equations models. Forecasts from VAR models are quite flexible because they can be made conditional on the potential future paths of specified variables in the model VAR is a mathematical model developed with the intent of establishing correlation between multiple variables and casting forecast for these variables based on inter-relationships in historical data (Zivot & Wang, 2006). It's an extension of autoregressive models which only consider one variable at time. It is commonly used in finance, economics and social sciences, such as population growth predictions.

2.2.2.2. Machine Learning methods

Traditional statistical models like ARIMA and VAR work well for many time series problems, but they struggle with nonlinear patterns, high-dimensional data, and complex dependencies (Kane et al., 2014; Siami-Namini et al., 2018). Machine learning methods like Random Forest, XGBoost, and Support Vector Regression (SVR) offer alternative approaches by learning patterns from historical data and extracting meaningful features for forecasting (Kane et al., 2014).

Random Forest and XGBoost are ensemble learning methods that use multiple decision trees to vote in made predictions based on past values and engineered features. Instead of relying solely on lagged observations, they can incorporate external factors (e.g., economic indicators, weather data) to improve accuracy (Tyralis & Papacharalampous, 2017).

SVR is a machine learning method based on Support Vector Machines (SVMs) that is used for forecasting continuous values. It works by mapping input features into a higher-dimensional space using a kernel function and finding the best-fit hyperplane that minimizes prediction errors within a margin (Suthaharan, 2016). This technique is more appropriate for small datasets with noisy data, being able to capture nonlinear relationships using the different available kernel functions such as gaussian kernel Radial Basis Function (RBF), polynomial or linear (Ben-Hur et al., 2008; Q. Liu et al., 2011).

2.2.2.3. Deep Learning & Advanced methods

Deep learning has changed the way time series forecasting is done by enabling models to learn complex, nonlinear patterns directly from extensive data (B. Lim & Zohren, 2021). It is a type of machine learning that increments its complexity by adding a greater number of trainable weights, resulting in substantially better results. Since it accomplishes different objectives using different techniques, it is here presented in separate. In contrast to traditional methods, which require manual feature engineering and assumptions about stationarity, deep learning models can "learn to" extract features and adapt to long-term dependencies (B. Lim & Zohren, 2021). The most used architectures for time series forecasting include RNNs, LSTMs, GRUs, and transformer-based models (B. Lim & Zohren, 2021).

Work on the foundations of RNNs began in 1986 with the publication of (Rumelhart et al., 1986) where the authors introduced the first mechanism for error propagation over time in neural networks. They demonstrated how differentiation could be used to adjust parameters across multiple time steps, leading to lower error rates. While this work established the training foundation for recurrent models, the structure of an actual RNN was formalized four years later. In (Elman, 1990), the author introduced what became known as the Elman Network, one of the first practical RNN architectures. This model incorporated context units to store past hidden states, allowing the network to learn sequential dependencies over time. The Elman Network demonstrated how neural networks could process time-dependent patterns, significantly advancing the field of sequential learning.

RNNs operate by learning in steps, where each step corresponds to an instance of a given time series (Elman, 1990). At each step, hidden states (numerical vectors representing past knowledge) influence future outputs, enabling the network to model temporal relationships in data. This ability to retain and process sequential information is what makes RNNs fundamental for tasks like time series forecasting, speech recognition, and natural language processing.

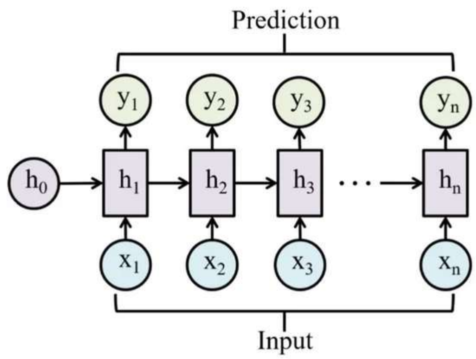

Figure 5 - Visual representation of RNNs

LSTMs were introduced in (Hochreiter & Schmidhuber, 1997) as a solution to a major limitation of RNNs: the vanishing gradient problem. In RNNs, as information is passed through many time steps, gradients (numerical values that indicate how a neural network's weights are adjusted during training) tend to decrease exponentially, making it difficult for the network to learn long-term dependencies. LSTMs addressed this issue by introducing "memory cells" and a "gating mechanism" that decide to keep or discard information over many steps.

The fundamental innovation in LSTMs is the cell state, which acts as a dedicated memory capable of carrying information across many time-steps (Hochreiter & Schmidhuber, 1997). This is regulated by three gates: the forget gate, which determines how much previous information should be discarded; the input gate, which decides what new information should be kept; and the output gate, which controls how much of the stored information contributes to the final prediction. These mechanisms allow LSTMs to better model long-range dependencies.

LSTMs marked a significant breakthrough in sequential learning, enabling neural networks to maintain meaningful long-term memory while removing some of the instability that is present in RNNs (Sherstinsky, 2020). Their ability to handle complex patterns over extended time frames made them the dominant architecture for sequential tasks until the emergence of more advanced models like GRUs and transformers.

GRUs are an alternative to LSTMs to handle time series data, which are not necessarily better performing, but are intrinsically simpler in a technical perspective. The algorithm combines the input gate with the forget gate, creating a single "update gate". The technique was proposed in (Cho et al., 2014) and it has proved to be faster than LSTMs in training and inferencing while maintaining similar performance, therefore being preferred to real-time forecasting (Cahuantzi et al., 2023).

Transformer architecture was also developed with the purpose of learning patterns from large amounts of data and rather long sequences (Vaswani et al., 2023). However, this topic will be more thoroughly expanded on a following unit dedicated to Large Language Models, as it represents the foundations for these models.

2.2.2.4. Applications for Time Series Forecasting Models

The following table highlights various applications of the models discussed earlier, including mathematical/statistical methods, machine learning models, and deep learning techniques.

Table 3 - Time Series forecasting models and applications

Methodology	Model	Applications
Mathematical/Statistical	Exponential smoothing	Wind speed prediction (Yousuf et al., 2022), consumer price index forecast (Febriyanti et al., 2021), gasoline price estimation (De Livera et al., 2011)
	ARIMA	Real estate price prediction (Albeladi et al., 2023), Stock market prediction (Srihari et al., 2024), estimation of machinery production (K.-Y. Chen & Wang, 2007)
	VAR	Finance, macroeconomics and transportation (Lenza & Primiceri, 2022; Martin et al., 2024)
Machine learning	Random forest	Flood risk assessment (Z. Wang et al., 2015), hearth health risk assessment (J. Wang et al., 2023), industrial process's fault prediction (Chai & Zhao, 2020)
	XGBoost	Credit evaluation (*XGBoost Model and Its Application to Personal Credit Evaluation | IEEE Journals & Magazine | IEEE Xplore*, n.d.), Structural damage prediction (S. Lim & Chi, 2019), Water resource engineering (sediment deposition and transport) (Niazkar et al., 2024)
	SVR	River flow estimation (Sahoo et al., 2019), financial forecasting (Trafalis & Ince,

Methodology	Model	Applications
		2000), genome-assisted trait prediction (Long et al., 2011)
Deep learning	RNN	Speech recognition (Miao et al., 2015), credit loan estimation (Babaev et al., 2019)
	LSTM	Flood forecasting (Le et al., 2019), Speech recognition (Miao et al., 2015), RUL estimation (Y. Wu et al., 2018)
	GRU	RUL estimation (Zhou et al., 2022), cybersecurity (Assis et al., 2021)

2.3 Multi-agent systems

Multi-Agent Systems (MAS) belong to a subfield of AI responsible for the organization of multiple autonomous entities (agents) in a way that facilitates the solving of complex problems that may not be possible otherwise using a centralized system (Dorri et al., 2018). Responsibilities of the field include definition of the capabilities that these agents have at their disposal to interact with their environment, define individual agent behaviour, coordinate inter-agent relationships and definition of the whole system's architecture.

The atomic entity of a MAS is the agent, and multiple definitions are found on the literature, including: "a flexible autonomous entity capable of perceiving the environment through the sensors connected to it" (Russell & Norvig, 2016), "an encapsulated computational system that is situated in some environment and this is capable of flexible, autonomous action in that environment in order to meet its design objective" (D. Ye et al., 2017), and "an entity which is placed in an environment and senses different parameters that are used to make a decision based on the goal of the entity [where] the entity performs the necessary action on the environment based on this decision" (Dorri et al., 2018). So essentially, what is important to retain is that this "agent" entity has the ability to sense its environment, either through the use of sensors or any other external factor and then be capable of processing that information and ultimately have an action output, such as transmitting information to other agents, using actuators, or any other means to interact with the external world while maintaining a relatively high degree of autonomy and independence from other agents.

2.3.1 MAS organizational designs

Throughout the development of MAS different use-cases have appeared that required different designs to be created in order to better accommodate the particularities of each environment. A MAS organizational design must include the description of many different aspects regarding its deployment and construction to successfully reach the desired goals, this includes agent roles, relationships, levels of authority and autonomy in governing behaviour (Horling & Lesser, 2004). Deciding on a MAS organizational structure is accepting that no one will be perfect for a given situation, as there will always be trade-offs between including or excluding certain aspects of a the different available architectures (Romelaer, 2002). Table 4 schematizes the most popular MAS organizational structures according to and adapted from (Horling & Lesser, 2004).

Table 4 - Popular MAS organizational structures

Architecture	Structure	Advantages	Disadvantages	Example Applications
Hierarchy	Tree-like structure with clear authority levels	Efficient task decomposition, reduced complexity, scalable	Rigid structure, single point of failure at higher levels	Manufacturing control, distributed AI systems
Holarchy	Small groups with different hierarchies (holons)	Balances autonomy and coordination, adaptive, scalable	Complexity in decision-making, unpredictable behaviour	Manufacturing, robotics
Coalition	Dynamic, goal-directed groups of agents	Flexibility, allows resource sharing, short-term specialization	Temporary, requires negotiation overhead	Sensor networks, task allocation in multi-agent systems
Team	Group of cooperative agents working towards a shared goal	Enables joint problem-solving, resilient to failures	High communication and coordination overhead	Robot soccer, military simulations
Congregation	Long-lived group based on shared attributes	Reduces search complexity, increases efficiency of partner selection	Less flexible than coalitions, potential suboptimality	E-commerce, recommendation systems
Society	Open, rule-governed system with evolving members	Scalable, supports heterogeneous agents, enables long-term stability	Requires enforcement of norms, may limit individual agent autonomy	Electronic marketplaces, virtual agent communities
Federation	Group of agents with a	Simplifies interaction, integrates	Single point of failure at intermediary,	Interoperability in heterogeneous

Architecture	Structure	Advantages	Disadvantages	Example Applications
	representative facilitator	heterogeneous agents	potential inefficiencies	systems, logistics management
Market	Competitive, decentralized environment with self-interested agents	Highly scalable, efficient resource allocation, adaptable	Requires strong economic modelling, potential unfairness	Stock trading, resource allocation in cloud computing
Matrix Organization	Agents report to multiple authorities in different dimensions	Balances hierarchical control and flexibility, supports complex tasks	Complexity in management, role conflicts	Large-scale enterprise systems, interdisciplinary research collaborations

2.3.2 MAS communication

Inter-agent communication addresses the exchange of information between independent agents of a given system and is one of the most crucial aspects of a successful MAS. A well established and implemented architecture with well-defined communication protocols enables agents to share external environment observations, and knowledge rationalized from those observations (Berna-Koes et al., 2004), this usually results on the development of an Agent Communication Language (ACL). ACLs facilitate cooperation and efficient task solving between different members of a. If ACLs are not properly set up, agents are not able to synchronize collective knowledge and conflicts may arise when accessing shared resources, and the decision-making process of each individual agent is compromised by a wrong understanding of the world (Huget, 2003). The two most famous ACLs are Knowledge Query and Manipulation Language (KQML) (Finin et al., 1994) and Foundation for Intelligent Physical Agents Agent Communication Language (FIPA-ACL) (Berna-Koes et al., 2004). In search-and-rescue missions employing a MAS the importance of well-developed ACLs is evidenced, as it is crucial that the covered area is properly updated and shared by everyone, and failure to do so will result in redundant area search affecting the duration it takes to cover large terrains which in turn, may cost precious time for people in distress (Drew, 2021).

MAS systems can be organized in different ways to publish information across the network, the definition of an architecture is tied to each use-case (Ferber et al., 2004) and there are, essentially, two possible broad types of knowledge routing: centralized and distributed (C. Zhang et al., 2008).

Centralized systems provide agents with an easier way to send and receive information, as its conveyed only to and from a central component of the MAS (Dorri et al., 2018). This central controller is responsible for routing information between agents and keeping them all updated as new information is transmitted and renewed. This however comes at the cost of the possibility for creating performance bottlenecks on that central point of communication hindering system performance in use-cases with many active agents, meaning that this approach may not always scale well (Dorri et al., 2018). In contrast, distributed systems are designed to focus agent messaging processing locally and guaranteeing peer-to-peer communication by using direct exchanges instead of relying on mediators (besides communication brokers and protocols). This provides a more scalable system that is more robust and easier to maintain as it grows, coming at the expense of more complex locally agent implementations and the requirement of increased hardware computation power (Dorri et al., 2018; Ribeiro et al., 2022). Figure 6 illustrates, visually, how these 2 system architectures are built.

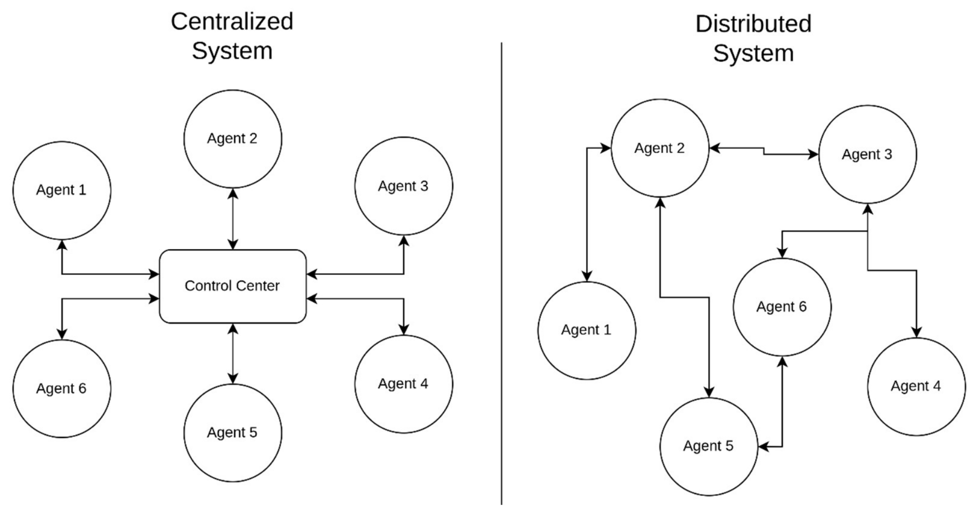

Figure 6 - Visual representation of Centralized System vs Distributed System

Communication between agents relies on a common language that must be established beforehand and known by everyone in that environment. This language must define vocabulary, the sets of every symbol on that language; semantics, representing the meaning of symbols; syntax, rules for assembling and constructing phrases using symbols; and pragmatism, defining ways to interpret and act upon symbols. Agents must also determine: what to communicate, as it is not possible to transmit everything they know; when to communicate; and to whom they send information, and how that information is going to be transmitted (Huget, 2003).

In terms of messaging capabilities, agents may be: basic, meaning they are able to receive information, but cannot perform any other type of communication; passive, besides receiving assertions, these agents are only capable of receiving questions from others and receive to those same questions with assertions; active, a more independent agent which is able to create active dialogue with others, has the capability to accept assertions, make assertions and question other agents; interlocutor, capable of carrying out a dialogue in which it assumes an interlocutor role between other agents, in addition to accepting assertions, is capable of asking and receiving questions and making assertions (Huget, 2003). Table 5 illustrates the difference between every agent's communication condition.

Table 5 - Different types of agentic communication behaviour

Agent	Basic	Passive	Active	Interlocutor
Receive assertions	Yes	Yes	Yes	Yes
Receive questions	No	Yes	No	Yes
Send assertions	No	Yes	Yes	Yes
Send questions	No	No	Yes	Yes

2.3.3 Agent architectures

Similarly to a MAS definition, the characterization of its atomic unit (the agent) is also subject to debate and academic discussion. It is, however, important to define these atomic entities that together (and orchestrated as mentioned above) can solve problems that otherwise wouldn't be possible. As such, this dissertation will adopt the definition proposed in (JENNINGS & WOOLDRIDGE, 1995): "An agent is a self-contained problem solving entity (implemented in hardware, software or a mixture of the two) which exhibits the following properties: Autonomy [...]; Social ability [...]; Responsiveness [...]; Proactiveness [...]." Furthermore, agents are required to have three distinct components: sensors, enabling the receiving of external information, this might represent any type of medium that allows for input of data, crucial for the second essential component; the reasoning mechanism, which can range from rather simple to very complex and enables the agent to treat and process inputs to then provide an output, using the third and last component; the actuator which serves as a way for the agent to actually make use of the information received and processed. Actuators can be any type of form that allows for external interaction, such as mechanical components or information-relaying to other agents(Ribeiro et al., 2022).

Internal agent architecture can be varied according to the type of behaviour. including deliberative, reactive, hybrid, and belief-desire-intent (BDI) (Dorri et al., 2018; Sudeikat et al., 2007).

Deliberative agents rely on internal symbolic representations of the world and logical reasoning capabilities. These agents maintain and update a model of their environment and plan actions using well known algorithms, such as search and planning techniques derived from classical AI (Corchado & Laza, 2003). This architecture assumes the agent has access to sufficient computational resources and time to carry out extensive reasoning prior to action, making it particularly suited for static or predictable environments. While such an approach allows for complex, goal-driven behaviours and long-term strategy formation, it can suffer from limitations in dynamic settings where adaptability and immediate responsiveness are crucial (Corchado & Laza, 2003). Nevertheless, its structured, plan-based operation remains fundamental to many high-level AI applications (Ribeiro et al., 2022).

In contrast to the deliberative approach, reactive agents are created based in the principle of action-response behaviour, instead of creating great internal representations and long-term planning they provide better real-time action (Peter Bonasso et al., 1997). Originating from behaviour-based robotics and subsumption architectures (a layered deliberative agent architecture proposed by Rodney Brooks in 1980, in (Brooks, 2003)), reactive agents interpret sensor data and immediately act upon it allowing for more swift movements in time-critical environments. Although considered less intelligent in the classical AI sense, reactive agents are often more robust in uncertain and dynamic changing environments (Corchado & Laza, 2003).

Hybrid architectures try to integrate both deliberative and reactive models to leverage their both designs strengths. These architectures are typically composed of at least two components: one for high-level reasoning and another for low-level reactive behaviour. The deliberative layer may be responsible for long-term goal generation and complex planning, while the reactive component handles immediate responses to environmental changes (Arkin & Mackenzie, 1994). The coordination between these layers is rather complex and often requires a sophisticated control mechanism to balance responsiveness with difficult and more resource intensive strategic planning. Hybrid architectures are particularly well-suited to dynamic and semi-structured domains where pure reactive agents and fully deliberative ones are sufficient.

Building upon the hybrid model, layered architectures introduces a more formal and hierarchical organisation of functionalities within an agent.

On a BDI architecture, an agent's behaviour is defined by three primary capabilities, those being: beliefs, that represent the information that the agent holds about the external world; desires, indicating its goals; and intentions, which are the plans it chooses to act to arrive to its objectives (Corchado & Laza, 2003). This structure provides a dynamic decision-making process, where the agent regularly updates its beliefs, creates new desires, and then adjusts intentions accordingly. The BDI model is a mix of both deliberative planning and reactive adaptation, providing a form of intelligent behaviour in difficult-to-predict and goal-oriented environments. In (D. Singh et al., 2016), BDI architecture is employed to provide a decision-making layer for agents within agent-based simulation environments. The paper introduces a framework that promotes integration of different BDI agents. This proposal allows for the simulation of agents that exhibit human-like deliberate actions, such as goal creation and decision, and intention updating, on complex virtual worlds/environments.

2.4 Large-language models

The current best performing natural language processing (NLP) models are the ones based on the transformer architecture (Braşoveanu & Andonie, 2020). This technique is the main driving force of the present developments in the field, where resources are being invested at a rate never-before seen in the field of AI (Asthana et al., 2025). It seems that almost every day a new technology emerges dethroning others before in the existing benchmarks that serve as indicators of performance and daily updated with the latest models. It is very possible (and likely) that when this state-of-the-art is published, the latest transformer-based model available at the time of reading is already considered subpar to a more recent one. Nevertheless, the current large language model (LLM) on top of the opensource and popular leaderboard "Chatbot Arena" under the "Overall" performance category is "GPT-4.5-Preview" from OpenAI, with a proprietary license, and "DeepSeek-R1" from DeepSeek, with an MIT license (*Chatbot Arena (Formerly LMSYS)*, n.d.). This unit provides a background and a brief history from the beginnings of NLP to the contemporary state-of-the-art.

The study of NLP is not, in any way, a recent one (Johri et al., 2021). It's a sub-field of AI and linguistics concerned with interaction between machines and natural human languages, and its main objective is to help machines understand, represent and generate language in a useful way. It started with rule-based approaches, especially in translation, evolved to statistical methodologies and finally shifting to machine-learning/deep-learning systems following contemporary trends and hardware developments (Johri et al., 2021).

2.4.1 Early NLP era

The beginning of language processing started around the 1930s, before any statistical or machine learning methods were available, since by this time no text corpus were available, and any machines being coded were still instructed with very limited tools such as assembly language or even more primitive mechanical methods (Johri et al., 2021) this early period was focused on symbolic approaches, rule-based methods and formal logic of language syntax.

The first breakthrough came with the development of a translation machine by Georges Artsrouni and Peter Troyanskii that attempted to translate Russian to English using a handmade word correspondence table that they created and maintained (Figure 7) (Lee, 2023). This, however, was a very basic and mechanical approach which also did not maintain any rules of syntax and would only translate word for word, therefore being useful only as a primitive bilingual dictionary.

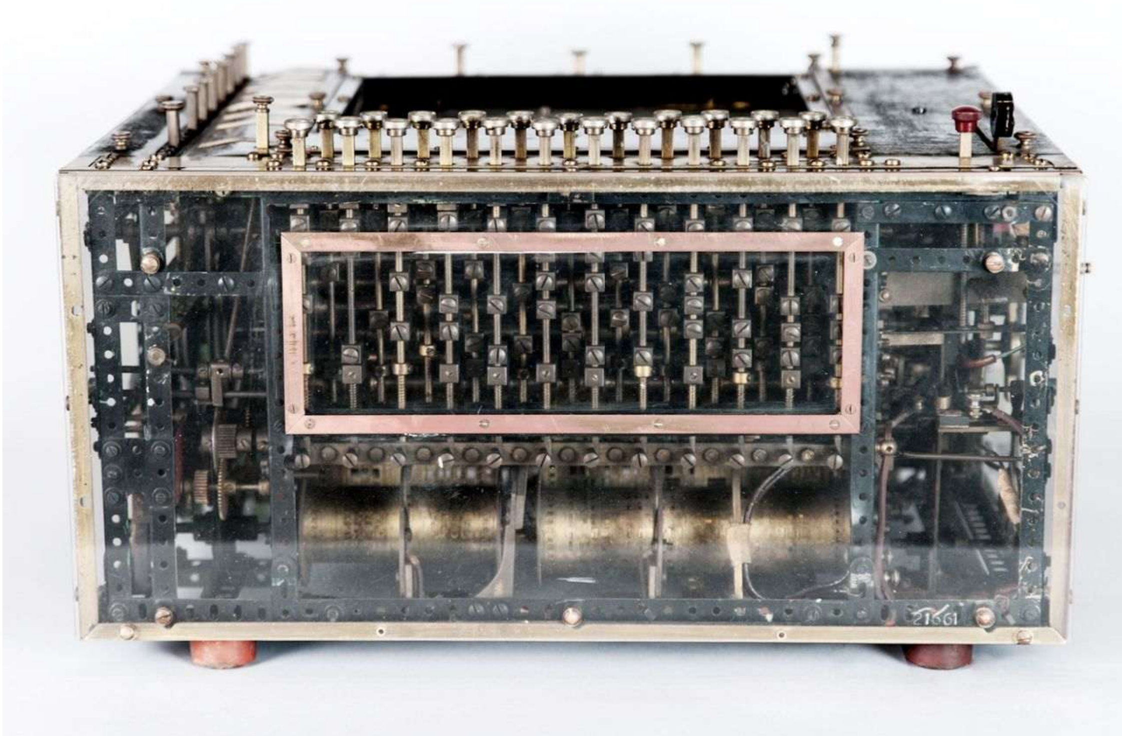

Figure 7 - Machine invented by Georges Artsrouni to translate Russian-English

In the 1950s Alan Turing established a foundation and testing criteria for the future development of the field, by creating the "Turing test" in his paper "Computing Machinery and Intelligence" (Turing, 2009). This exam tested the capabilities of machines to mimic human language, by having a participant distinguishing between two different texts, and selecting the one they thought was human made.

One of the great first developments of NLP was in 1957 when Noam Chomsky tried for the first time to introduce syntax structure and formalize language generative and parsing rules (Chomsky, 1956). By proposing the concept of "context-free grammars" (CFG) he attempted to demonstrate that there were recursive rules capable of defining multiple languages and therefore a rule-based system was capable of being created that formalized syntax. This way, Chomsky laid the foundation that motivated others to explore this concept of rule-based language parsing, and in the end of the 1950s the first computer programs were made that could parse human language (Johri et al., 2021).

In the 1960s, the development of NLP with resource to rule-based systems continued and a particular interesting system named ELIZA was created by Joseph Weizenbaum that made use of some tricks and patterns of human language to provide convincing responses, although it couldn't "understand" what was being said, and it only worked properly in very specific environments it was devised for (Weizenbaum, 1966).

In the mid-1960s there was a slow down on the developments made on the field after a North American report by the Automatic Language Processing Advisory Committee (ALPAC) who was tasked with evaluating the progress made. ALPAC concluded that the results were not the expected and found them to be unsatisfactory, therefore cutting much of the U.S. budget on the field (Jones, n.d.). Even so, in 1968 Peter Toma founded a company that developed the system SYSTRAN, a rule-based system to translated Russian to English which was successful enough that the American Airforce put it to use in translating Russian scientific documents (Wilks, 2009).

In conclusion, this first era of NLP was demarcated by great efforts to establish rules that were able to encompass generic language rules, it was driven by the US due to the need to translate Russian to English and it was a period that laid foundational theories and helped discovering the potential for hand-written rules and grammars to analyse syntax. It was also a time that enabled researchers to understand the limitations and efforts that it took to understand the formally characterized language and the implications of using it in computers (Johri et al., 2021).

2.4.2 Statistical NLP era

The second era of NLP developments is distinct from the first - which employed rule-based approaches, expert learning and hand-crafted heuristic patterns - since the main mechanisms used to interpret human language are mainly data-driven and empiric. These developments that began in the early 1980s were fuelled by two key factors: the availability of large text corpora, such as the Brown Corpus (Francis & Kucera, 1979) and the Penn Treebank project (Taylor et al., 2003); and the emergence of hardware and computational power (Johri et al., 2021).

As mentioned before, rule-based methodologies had some restrictions that prevented those systems from scaling well, namely the rigidity of those models regarding syntax rules and rule odd cases, the difficulty present to encompass every rule and poor handling of ambiguity and variation present in real-world scenarios (Johri et al., 2021). Statistical methods of NLP rely on machine consumption of large amounts of data that in turn allow these systems to empirically infer hidden patterns and therefore deduce construction laws of these same patterns providing speech predictions, information extraction and translations (Jacobs, 1992). This methodology allows the system itself to figure out the present rules of natural language and solve ambiguities based on probability distributions.

In this era of statistical focus, different models have emerged that provided useful tools to researchers. Part-of-speech tagging (POS tagging) is one of these cases, that allows text to be automatically decomposed word-by-word, and to each word be assigned its grammatical category, such as noun, verb, adjective, and adverb (Martinez, 2012). This represents a foundation for higher-level NLP processing tasks, for example, machine translation, named entity recognition, and document parsing.

A specific model for POS tagging technique was the Hidden Markov Model (HMM), described by Leonard E. Baum in a series of papers published in the late 1960s that highlighted the capabilities of the model to predict and learn sequencing patterns (Baum et al., 1970; Baum & Eagon, 1967; Baum & Petrie, 1966; Baum & Sell, 1968). HHMs are generative stochastic models that assume a system is determined by a sequence of states that together stochastically generate a given output. The hidden states of the HHM follow the Markov principle that states that probability of a given state to transition to another one is given exclusively by the contents of that state, in other words, state T is only dependable on T-1. This of course represents a disadvantage on the precision of the model, but greatly simplifies the calculations made on the joint probabilities of state.

Another influential foundation for statistical NLP models is the n-gram model, indirectly proposed by Claude Shannon in (Shannon, 1948), n-gram models are stochastic language model that calculates the probability of a next-word based on the occurrence of past words. For example, if dealing with a bi-gram model, the goal is to determine the probability of a word, given one past word. Assuming a tri-gram model, in the sentence "I like", the most likely word between "cloud" and "pizza" is pizza and therefore the model would predict that very same word to be the next one. This is all learned from parsing and consumption of large text corpora which, through this empiric process determines the probability table for the foreseeable sequences. These models are very simple to understand and implement but become powerful tools for syntax prediction. Auto-completion suggestions are one of the most common use-cases that directly implement n-gram models.

Evolving from the foundational CFGs, proposed by Chomsky in 1957 and defined above, emerged the Probabilistic Context-Free Grammars (PCFGs) that iterate on CFGs by adding probabilities to the generative rules that are defined by them (Jelinek et al., 1992). This allows generative parsers to rank every possible syntactic structure of a sentence according to its likelihood. By maintaining the formalized traditional grammar structure. PCFGs appear to be useful tools that, by consuming large text Corpus can help solve syntactic ambiguities present in natural language, based on frequency of appearance on the training data.

A simple, and yet effective model which also had its roots in the past but became prominent in this era was the Naive Bayes classifier (S. Xu, 2018). The technique has in its core the Bayes' theorem, and it is used to estimate the probability of a next label (or class) based on previously provided classes, and (necessary to the theorem) assuming that these classes are independent amongst each other. Although this independence is not usually observed in nature, the methodology provides results with significant value. More specifically, to the field of NLP, these models are used for text classification tasks in use-cases such as email spam detection, sentiment analysis and document classification.

Other models emerged during this time that both constructed many of the bases and built upon ideas of the past, for example Maximum Entropy Models (Ratnaparkhi, 1999) and Conditional Random Fields (Sutton & McCallum, 2012). However, as important to the development of the field as they are, they stray away from the scope of this chapter. Nonetheless, it is important to retain that this was the dawn of machine learning associated with many different fields of study, and in natural language processing much research was also being conducted. This period was crucial to the maturity of the stochastic techniques and besides models, evaluation metrics were also being created that are still used. Some examples of these metrics include precision, recall, F1 score, and BLEU (Jones, n.d.).

2.4.3 Deep Learning NLP era

With the promising results of statistical methods in the field of NLP, researchers began creating more complex and demanding algorithms. These developments are supported by the increasing availability of computational power, derived from the emerging capabilities of today's GPUs (Jeon et al., 2021). At this stage, a shift from mere statistics has evolved to something bigger and more abstract, where almost every model is now reliable on the same core engine: a neural network. This is the era of deep learning, not only in NLP, but in other fields of AI too, such as computer vision, robotics and image generation.

Neural networks grow larger, and in some tasks a model containing three billion parameters is considered to be a very small and poor performing one (Tuggener et al., 2024). Many algorithms and configurations are being tried and tested on great datasets in current times. In NLP the dominating algorithms are supported by the transformer architecture, which itself relies on the use of neural networks (Patwardhan et al., 2023). Many different transformer architectures are being experimented with and researched. Each day innovations are being published, benchmarks are being surpassed, and the Turin test gets ever more difficult to correctly answer. We are truly living in a "biodiversity boom" of NLP where investments are made in factors never seen before. But before arriving at these outstanding performing models, it might, perhaps, be important to understand the early stages and the developments that lead up to this last era of NLP.

A specific characteristic of neural networks is that they are only capable of handling numeric data (Vaswani et al., 2023). Words, sentences and documents are, obviously, not, by default, represented in numerical format, therefore, to make use of these promising neural models, it is necessary to devise a way to convert letters, symbols or tokens to numbers and vectors, while maintaining the inherent meaning and context behind each word and sentence. This technique of translating symbols to numbers can be performed by using vectorization algorithms, or embedding techniques, similar to the first, but more complex and costly (Mitra & Craswell, 2017; A. K. Singh & Shashi, 2019).

Vectorization is a simpler process of making this conversion, and it is a broad term that encompasses many of the techniques used to make it. Popular examples of vectorization are Bag-of-Words (BoW), and Term Frequency-Inverse Document Frequency (TD-IDF) (A. K. Singh & Shashi, 2019).

BoW is a very simple and rudimentary implementation of converting text into vectors by counting term frequency, but with the drawback of losing word order meaning and word similarity is not considered, for example "good" and "great" are treated as unrelated terms (Salton et al., 1975).

TD-IDF, a more advanced vectorization algorithm than BoW, is a method that statistically measures how important a word is on a given corpus, by evaluating its frequency (TF) and multiplying it by unique that word is across multiple documents using a specific formula, presented in Equation 1, where $f_{t.d}$ is the raw count of term t in document d. TF-IDF does not, however, consider context of terms (Salton & Buckley, 1987).

$$TF(t,d) = \frac{f_{t,d}}{\sum f_{k,d}} \tag{1}$$

Embedding techniques are more complex algorithms of vectorization that map words or tokens to dense, low-dimensional vectors with the goal of capturing meaning and semantic relations, for example, adding the embedded vector of "woman" to the embedded vector of the word "king" would result the vector "queen". Common embedding methods are Word2Vec, Global Vectors for Word Representation (GloVe), and FastText (Mitra & Craswell, 2017).

Word2Vec is word embedding technique proposed in (Mikolov, Chen, et al., 2013; Mikolov, Sutskever, et al., 2013), which provides breakthrough benchmark performance in capturing meaning in text semantics. This methodology is still used today for its simplicity and performance. It uses lightweight neural networks to map words to their surrounding context; examples of neural networks are Continuous BoW (an iteration of BoW) and Skip-gram.

GloVe is an embedding method that builds word vectors by leveraging global co-occurrence statistics across a corpus, rather than relying on local context prediction like Word2Vec. It constructs a large co-occurrence matrix and then factorizes it to produce dense, low-dimensional vectors that capture both meaning and frequency-based word associations. This approach allows GloVe to encode semantic relationships in a way that reflects how often words appear together in the entire corpus, not just within a narrow window. The result is word embedding that is efficient, interpretable, and useful for a wide range of NLP tasks where understanding word meaning at a broader level is important (Jeffrey et al., 2014).

FastText is an extension of Word2Vec that iterates it by including sub-word information, this means that it doesn't just learn vectors for entire words, but also for smaller chunks of words. This allows it to build embeddings even for rarely seen or unseen words by joining together the vectors of the sub-word components. As a result, FastText is incredibly useful for languages with lots of word variations or informal text like social media, where spelling can vary, and new words appear often, for example, new slang words. This sub-word-level modelling gives it a significant advantage over more traditional word-level embeddings that fail when met with words not seen during the training phase (Bojanowski et al., 2017; Joulin et al., 2016).

After the conversion of text to vectors using the above-mentioned embedding techniques, we still need a way to make sense of these vectors in a meaningful way, especially when dealing with sequences like sentences or documents. That's where deep learning models like CNNs, RNNs, LSTMs, and GRUs come in. These architectures are already detailed in previous sections of this document's state-of-the-art, as predicting next-words is conceptually the same as time-series predictions (Feghali et al., 2022).

2.4.4 Transformer Era

The previous sub-chapters served the purpose of broadly contextualizing the reader to the progression and history of the developments on NLP. Starting on the early era around the 1930s and proceeding with a descriptive overview until the most recent developments. The remaining section is dedicated to the study on how the transformer architecture came to be and how it is revolutionizing the field of NLP.

2.4.4.1. Contextualization of transformers

Before the appearance of the transformer architecture, the performance benchmarks of NLP were dominated by RNNs and its iterations (LSTMs and GRUs) (Buestán-Andrade et al., 2023). As described above, these models can store information from previous states in logical gates. However, these models, in practice, suffer from the commonly known "problem of vanishing gradient" (Noh, 2021), which means that, for long sequences of data (in this case long text or documents), early information of that sequence is "forgotten" by the model as it becomes less significant and represented in the logical gate units that store past information. An attempt to solve this problem is proposed in (Bahdanau et al., 2014) with the use of a mechanism entitled "Attention" on the "Seq2Seq" (Sequence-to-Sequence) architecture.

Seq2Seq is a type of neural network model designed to transform one sequence into another. It was originally proposed for neural machine translation (Sutskever et al., 2014), but has since been applied in many other NLP tasks (Dash et al., 2023; V et al., 2024; Xia & Wang, 2023). It is comprised of two distinct components, the Encoder and the Decoder. Fundamentally, the Encoder encodes a phrase into a vector (similarly to an embedding), which is then decoded by the Decoder to the desired output. For example, on a Russian sentence, the encoder would translate it entirely to a vector space, and afterwards the Decoder would "look" at that vector and translate it to any other language, such as English or Portuguese. Both of these parts use neural networks (usually RNNs, LSTMs, or GRUs) to, on one side, learn what and how to encode a given sequence and, on the other side, learn how to decode a particular vector. Both these processes happen word by word, taking the previous encoded words as context, as visually explained on Figure 8.

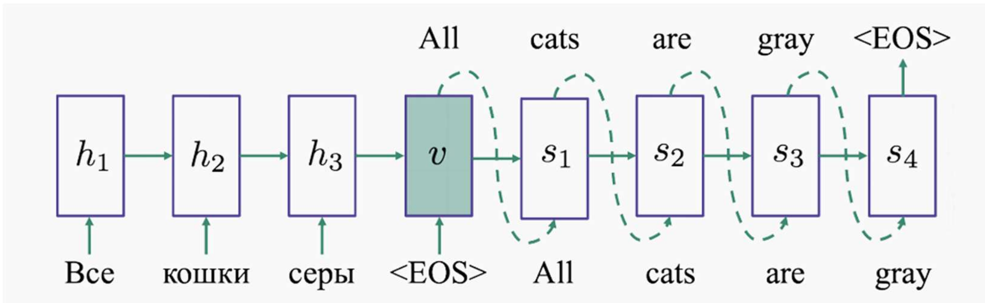

Figure 8 - Encoder/Decoder graphical representation

The attention mechanism iterates on the Seq2Seq concept by adding a level of complexity between the encoder and decoder elements. Instead of only using the vector resulting from the encoding phase, attention allows the analysis of each encoding step upon each decoding step. This means that for each token that the decoder generates, a lot more information is provided on relationships between tokens of the original encoded sequence, granting the decoder a better grasp of the contribution of each word to the result (Bahdanau et al., 2014).

The foundations of transformers rely on these two concepts of Attention and Encoder/Decoder. However, a sophistication of the Attention mechanism was proposed named “Self-Attention” (Vaswani et al., 2023). Self-attention not only allows the encoder and decoder to establish relationships between new tokens but also retains information about the relationships that the initial sequence token have amongst themselves. For example, “The dog jumped above the fence.”, self-attention allows the model to learn how each word relates to others. helping it understand that “dog” is the subject, “jumped” is the verb, and “fenced” is the object of the preposition “above”. Furthermore, self-attention is then expanded in multiple “attention heads” each gathering information on different relationships for the same token. In the above example, multiple attention heads for “dog” would gather relationships such as: one head might focus on the grammatical role of “dog” as the subject of the verb “jumped,” another might pay attention to how close it is to the word “the,” capturing syntactic structure, and a third might learn the relationship to “fence,” helping the model understanding spatial context of the action that the subject (dog) is performing.

In 2017, on the acclaimed paper “Attention is all you need”, the transformer architecture was proposed by researchers at DeepMind (Vaswani et al., 2023). This was the turning point of contemporary NLP models which revolutionized the entire field with models that surpassed every benchmark (Buestán-Andrade et al., 2023). These models come in three different forms:

- Encoder only models: use only the encoder part of the transformer. They're designed to understand and represent input sequences deeply, making them great for tasks that require understanding rather than generation, such as sentiment analysis, question answering and text classification. BERT is an example of one of these models.
- Decode only models: use only the decoder part, they use masked self-attention so that each token can only attend to previous tokens (not future ones). They are generally prompted with an “kickstart” system message guaranteeing the presence of previous

tokens. This is crucial for generative tasks, such as: text generation, autocomplete, and story writing. Generative Pre-trained Transformer (GPT) models are examples of these models.

- Encoder and decoder models: use both the encoder and decoder components together and are great for sequence-to-sequence tasks, particularly in machine translation. BART and T5 models are examples of these models.

2.4.4.2. Transformer Characterization

The transformer is, as described in Figure 9, comprised of several layers:

- Multi-head self-attention.
- Position-wise feed-forward networks (FNNs), responsible for transforming each token's representation independently.
- Residual connections, which help prevent degradation of information across layers.
- Layer normalization, which stabilizes and speeds up the training process.

These components are stacked in both the encoder and the decoder, allowing the model to build complex and contextualized representations of the input and output sequences.

Since the transformer does not process sequences sequentially, as RNNs do, it introduces positional encoding to provide a sense of word order, essential for understanding natural language structure (Vaswani et al., 2023).

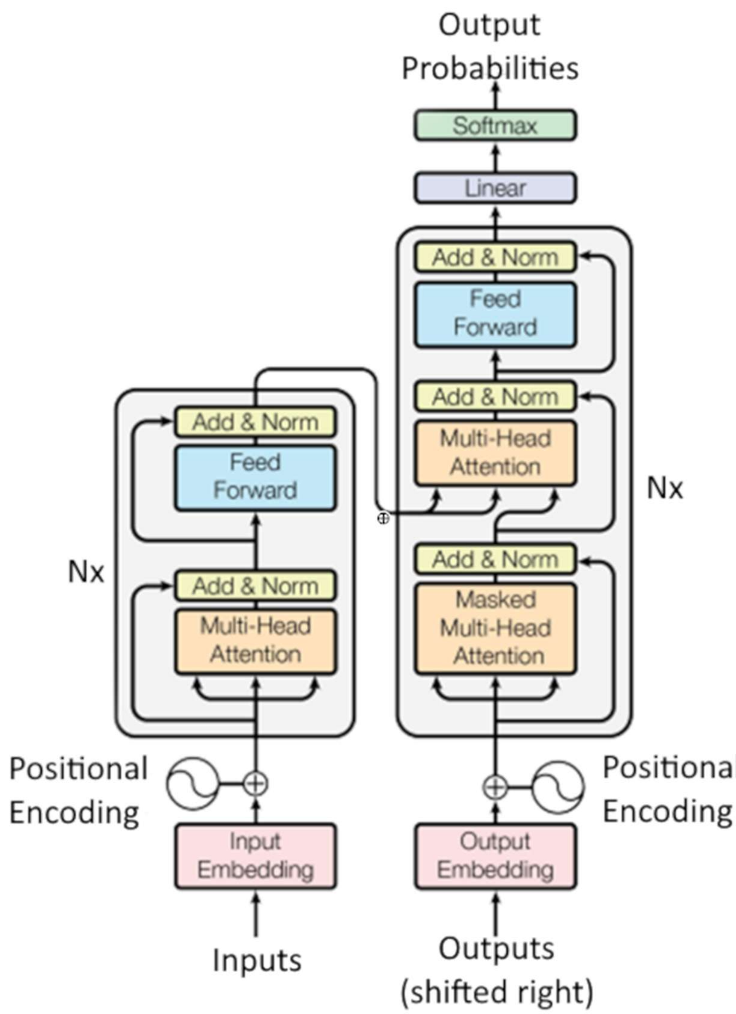

Figure 9 - Transformer architecture (Vaswani et al., 2023)

Model implementation implies the creation of a great number of trainable weights. These parameters all must be tuned during the training phase, where the model is presented with large amounts of text corpora. For example, Llama3, a Meta-released LLM, was trained on about 15.6 trillion tokens (Grattafiori et al., 2024). When constructing a transformer-based model numbers such as 3 billion trainable parameters and even 11 billion parameters, are considered to be very small and tend to offer subpar results (Tuggener et al., 2024). The current top performing models (that do not have these numbers hidden behind a proprietary license) have more than 650 billion parameters.

Due to the sheer scale of these models, great processing datacentres are required to achieve significant results with models trained from scratch, for example, model Llama2 took around 1.7 million GPU hours to train spanning months of continuous hardware uptime, and this is referring only to the 70 billion parameter version (Touvron et al., 2023). Therefore, creating new LLMs is not viable to institutions that do not have access to large enough corpora nor processing resources.

This scenario made research and development of custom and domain specific LLMs a lot harder due to the lack of available computing power and scale of the necessary investments, motivating researchers to search for ways to use already existing and pretrained models and adapting them to their own needs (Shekhar et al., 2024). From this necessity a less costly way to adjust and finetune existing LLMs was proposed in (Hu et al., 2021), named "Low-Rank Adaptation" (LoRA). LoRA, allows for the retraining of specific weights of a model while freezing others, therefore drastically reducing the required computation (Hu et al., 2021).

For institutions, the usage of these models is also rather expensive, good quality LLMs, require GPUs with high V-RAM which are not cheap, furthermore a single LLM can only respond to one request at a time since each request has a different context, therefore making it much harder to serve LLMs to multiple users without investment on dedicated GPU-powerful servers (Shekhar et al., 2024).

A way to reduce the size that LLMs occupy in V-RAM is a technique named "quantization of models", which reduces the accuracy, and therefore performance, of next-word generation, but allows bigger models to "fit" in smaller GPUs. Quantization is a method which reduces the number of precision present in the stochastic parameters of the LLM, for example an 8-bit quantization only allows the model to compute probabilities with a size of 8-bits, therefore losing some information (Egashira et al., 2024).

2.4.4.3. Open vs Closed AI

The advent of LLMs has established a significant landmark on the NLP field, defeating every other methodology proposed and known before and completely changing the research focus of the study area (Buestán-Andrade et al., 2023). However, as these systems grow, become more efficient and more useful, a very noticeable divergence within the community is present, on one side model development and publishing is open-source, and on the other it's closed source and only available through endpoints that provide access to these models as a paid services.

Within the open-source community significant progress and research is being conducted, Meta's models and community-made derivations of these models are one of the most used, including Llama 1,2 and 3, which were accompanied by documentation regarding how these systems were trained and produced (Grattafiori et al., 2024; Touvron et al., 2023). Furthermore, Google has also contributed to the open-source space with the release of Gemma models that demonstrate substantial performance and broad accessibility (Team et al., 2024). Other notable models include Mistral's Mixture-of-Experts (meaning that the models are made up of multiple specialized sub-networks, or "experts," where a routing mechanism dynamically selects and combines the outputs of a subset of these experts for given inputs) (Lo et al., 2024); and Cohere's sophisticated models that are specifically designed for certain tasks such as R programming (Cohere et al., 2025). Collaborative efforts of the open-source community have also provided some frameworks such as LangChain and LlamaIndex that provide ways to implement pipelines to develop applications integrating local and public available models (Afify et al., 2025). When security and transparency are priorities, the usage of these models is essential as the responsibility of serving them is entirely up to who implements them, allowing for more close monitoring and manipulation.

In contrast, proprietary Large Language Models, developed by leading organizations such as OpenAI (with their GPT series), Google (with the Gemini family), and Anthropic (featuring the Claude models), represent another significant way of LLM research and progress. These groups take advantage of great internal resources, including varied proprietary datasets and enormous computational infrastructure, to train models exhibiting state-of-the-art capabilities across a wide range of natural language understanding and generation tasks (Widder et al., 2024). The focus of the closed-source developments is in delivering user-friendly solutions, providing well-documented APIs and support structures, which brings a lot of value to companies that need easily deployable AI use-cases without requiring deep employee expertise. Furthermore, LLM serving organizations place a lot of attention to making sure that the safety and reliability of these models for broad commercial application is used. While the internal mechanisms and training data often remain unknown to the public, these models provide robust out-of-the-box performance. For organizations prioritizing immediate high-performance solutions and streamlined integration, the closed-source approach offers a solid alternative (Widder et al., 2024).

2.4.4.4. Retrieval augmented generation

From these advancements, the field of NLP started to explore ways to augment the already present capabilities of LLMs to perform tasks beyond simple question-and-answer, and in 2020, introduced in (Lewis et al., 2021a), a framework named “Retrieval-augmented generation” (RAG) was proposed. This novel approach had the goal to eliminate one of the greatest problems encountered with LLMs: the models cannot reply with information that was not present during their training phase, and simply providing large texts as context on each prompt is not feasible due to input token limits. Therefore, a way to provide crucial, never-seen-before information was required.

RAG provides a solution to this problem by fusing with each prompt the context related to that same prompt based on vectorized similarity between the prompt and the available context corpus. This means that every time a question is made to an LLM using RAG, that question is firstly parsed, then vectorized, and from an already vectorized context database (that can keep many different documents), the best vector similarity is extracted and provided as context to the LLM, which, by nature, operates on embedded representations of text, although the external vectors used for retrieval are first converted back into text before being passed to the model, Figure 10 better exemplifies how everything interconnects.

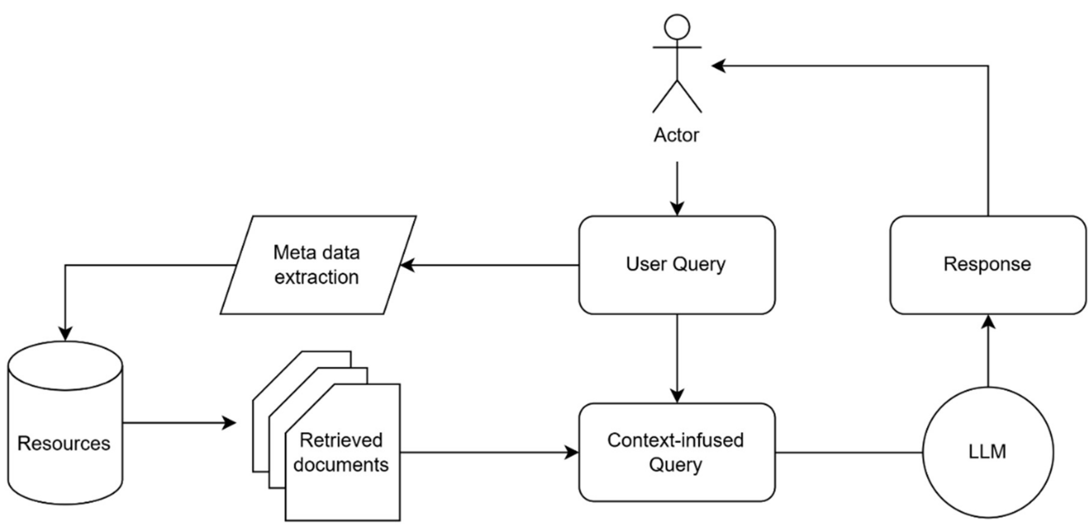

Figure 10 - RAG flow graphical representation

2.4.4.5. Agentic AI

With the advent of LLMs, and the increasing reasoning capabilities of these models, researchers began looking for ways to implement these algorithms into broader, larger contexts, such as MASs (R. Ye et al., 2025). Using LLMs as agents with different roles, these systems, commonly named as “Agentic AI systems”, can communicate in plain language, plan tasks, delegate actions, distribute knowledge and negotiate resources (Acharya et al., 2025).

Some advantages of these systems provide motivation for research and development on the agentic AI field. Plain language communication between agents allows for greater affinity between humans and machines while providing flexibility in coordination of tasks between agents, since the necessity of defining strict protocols for proper communication is now overshadowed by the capabilities of LLMs to interpret and deduce intention from natural speech. Built on top of this input analysis capability and interpretation, LLMs might provide enhanced decision-making when faced with ambiguous inputs by discerning the next steps to make, based on available outputs at their disposal (X. Liu et al., 2023). Some frameworks such Autogen as even provide reasoning loops and memory mechanisms to iterate on a problem and reach a conclusion (Q. Wu et al., 2023). Furthermore, as listed below, some implementations of Agentic AI systems provide specialization of agents, creating roles within a system that has specific goals.

However, the implementation of these systems do come with some challenges and disadvantages that might be cumbersome to unexperienced developers when trying to, for example, scale these systems to large enterprises as LLM models require great computational power to achieve each use case, and therefore, continuously running these integrations might cost large amounts of resources when running multiple clusters of agents serving various different tasks at the same time (Cemri et al., 2025). Furthermore, LLMs, as is the case with many other artificial intelligence applications, are often seen as black boxes making it harder for maintainers of these systems to easily debug and find critical points of failure or even understand where things might have gone wrong in a worst-case scenario. Besides, LLMs are, in nature, and as observed in previous sections, stochastic, which means results may not always be guaranteed (Vaswani et al., 2023). Although parameters such as temperature might help reduce uncertain behaviours, these systems still suffer from a problem of unpredictability and therefore building safeguards and verification pipelines (such as other LLMs that verify output) are necessary to be implemented (Zheng et al., 2024).

Although a very recent concept, the usage of LLM agents to solve complex problems has already sprung up a few different frameworks that allow users to create and experiment with orchestration of multiple AIs working together.

Autogen (Q. Wu et al., 2023) was developed by Microsoft Research and is currently one of the main frameworks used to coordinate tasks between multiple LLM agents. It is developed mainly in Python language, and it is an open-source project released in 2023. At its core, Autogen powers each agent by an LLM which is specified to have a specific role or task. Typically, standard agents on this framework have a name, a description and the designated task meant for the agent to run.

As opposed to other multi-agent AI systems, Autogen is agent-centric, meaning that a great focus of details is given to the agent itself, providing each with a memory, a goal, a role and communication rules that ensure the reasoning capabilities of LLMs are exploited to the maximum extent possible. Each agent operates independently within its defined scope but communicates and collaborates with others through structured message exchanges. These interactions happen in a turn-based conversational setup, allowing agents to discuss, evaluate, and refine their outputs as needed.

Because of its modular nature, Autogen supports a variety of use cases—from code generation and debugging to research automation and document processing. Developers can define custom agents, plug in tools, or even include humans in the conversation flow through a User Proxy agent. This flexibility allows for the creation of robust, semi-autonomous workflows that rely not just on a single LLM prompt, but on an intelligent and evolving dialogue between multiple reasoning entities.

TaskWeaver (Qiao et al., 2024) is another open-source framework developed by Microsoft, released in late 2023, with the primary goal of helping LLMs execute structured, tool-augmented tasks. Similarly to Autogen, it is completely written in Python, but unlike Autogen's agent-to-agent focus, TaskWeaver is centred around task planning and execution. Instead of multiple named agents having conversations, TaskWeaver builds a pipeline where tasks are divided, sequenced, and assigned to LLMs, with an option for support by external tools or functions.

Each task in TaskWeaver can be thought of as a callable unit, typically decorated with metadata that defines its purpose, input structure, and relationship to other tasks. The LLM acts more as a central planner and executor rather than an individual agent with personality or memory. It receives a user intent, decomposes it into subtasks, and routes them through the system's planner. While it doesn't offer the same level of agent individuality or chat-driven reasoning as Autogen, it excels in straightforward automation scenarios where clear structure, tool access, and deterministic control are prioritized.

Compared to Autogen, TaskWeaver is less focused on agent roles and more on efficiency of task orchestration. It's particularly well-suited for automating business processes, integrating APIs, and building LLM applications where predictability and control matter more than emergent multi-agent systems. CrewAI (Barbarroxa et al., 2025) is an independent open-source framework introduced in early 2024, built around the concept of organizing LLM agents into teams—or "crews"—that work together to accomplish complex goals. Like Autogen, CrewAI is agent-centric and written in Python, but its structure is more explicitly inspired by real-world teamwork. Each agent in a crew is given a defined role, a set of responsibilities, and optional tools it can use. Together, they form a crew that executes a shared mission through role-based coordination.

Agents in CrewAI are defined by their role, goal, backstory, and optionally a list of tools. They don't communicate in a freeform chat the way Autogen agents do but instead operate in a defined sequence or collaboration loop, where control is passed from one agent to another, often in stages. CrewAI emphasizes modularity and realism, making it easy to simulate workflows like a research team, a software development group, or a business strategy session.

Compared to Autogen, CrewAI is more lightweight and intuitive for scenarios where agents have clearly separated duties. It doesn't offer the same low-level control over memory or conversational dynamics, but it's faster to set up and works well when agent behaviour follows a clean, collaborative pipeline. It's more structured than TaskWeaver but less conversationally rich than Autogen—making it a good middle ground for many multi-agent LLM applications.

2.5 Final remarks

At the end of the analysis of the state-of-the-art, many different topics were discussed and analysed. A great shift of Industrial standards is happening to a more human centric design with the dawn of Industry 5.0, hardware advancements are enabling the application of different AI models that provide crucial insights into existing data as well as the creation of applications from that gathered data, such is the case of the NLP field where gathered text corpus is being used to create powerful language models. MAS are also benefited from these advancements as new approaches are being proposed using LLMs, which still sparks some discussion when considering them traditional MAS.

When answering the question "Can a system that integrates LLMs as rational engines be considered a traditional MAS?", it is useful to look at the very definition of agent, which was, for this work, defined as "An agent is a self-contained problem solving entity (implemented in hardware, software or a mixture of the two) which exhibits the following properties: Autonomy [...]; Social ability [...]; Responsiveness [...]; Proactiveness [...].” according to (JENNINGS & WOOLDRIDGE, 1995). Assuming that the only difference between traditional agents, and individuals that use LLMs to perform actions is the usage of LLMs, does this definition fall short or brakes at any point? Perhaps by going step-by-step through this definition helps achieving a conclusion, as well as defining an example to test and examine these premises and observing if they hold up to scrutiny.

Example: a multi-agent system responsible for controlling a smart building. This example assumes that each agent has enough hardware capabilities to support local LLM models, since this capacity is not directly related to LLMs but to the logical engine chosen for that agent, therefore not discarding in any way the usage of LLMs *a priori*.

“An agent is a self-contained problem solving entity [...]”: An individual using LLMs to solve problems is, in fact, self-contained and capable of achieving results solely based on external inputs. In the example, if an agent receives by input that a certain room should activate its air-conditioner, the LLM is capable of responding on how it should proceed by picking from an available action set defined previously, for example, in its system prompt.

“[...] implemented in hardware, software or a mixture of the two [...]”: LLMs are, at their core, and as described above, mathematical models that require some sort of memory to store parameters, which can very possibly be implemented in hardware. In our example, most agents responsible for information relay and activation of actuators can be implemented in small processing units that have internal computers which allow them to process the exterior world, these internal computers can store the models to enable their employment as they act on the world around them. It is also, as described above, possible to run LLM models on local machines.

“[...] which exhibits the following properties: Autonomy [...]; Social ability [...]; Responsiveness [...]; Proactiveness [...].[...]”: On the above citation, these four concepts were shortened, however, here it becomes relevant to expand each one so that it is possible to analyse each one in more detail:

- Autonomy: “agents should be able to perform the majority of their problem solving tasks without the direct intervention of humans or other agents, and they should have a degree of control over their own actions and their own internal state”: For the case of an individual that uses an LLM as a rational mechanism, this definition does not restrict the usage of such a an engine, since these models are capable of providing answers to their environment all by themselves. In the analogy, an agent employing an LLM can be autonomously commanded simply by providing an input and the available outputs and their format during a priori programming.
- Social ability: “agents should be able to interact, when they deem appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities. This requires that agents have, as a minimum, a means by which they can communicate their requirements to others and an internal mechanism for deciding when social interactions are appropriate (both in terms of generating appropriate requests and judging incoming requests).”: LLMs, by nature, always respond to given inputs, however, it is possible to (either through system prompt or through injection on every input) instruct the LLMs on what to respond to certain messages received as well as to describe their surrounding environment, these outputs can then be parsed, allowing for the control of the model’s communication intentions, such as

multiple responses to multiple individuals, no response, ask for more information, single response, etc...

- Responsiveness: “agents should perceive their environment (which may be the physical world, a user, a collection of agents, the INTERNET, etc.) and respond in a timely fashion to changes which occur in it. Responsiveness: agents should perceive their environment (which may be the physical world, a user, a collection of agents, the INTERNET, etc.) and respond in a timely fashion to changes which occur in it.”: LLMs always respond to inputs, the main concern on this particular aspect is that these answers may not always be timely due to the large processing capabilities that are sometimes required to process these models. However, technological advancements through time tend to mitigate this effect allowing for almost instantaneous responses. Furthermore, individuals running LLMs on a separate isolated cloud server allow for fast responses without compromising in any of the previous definitions of agent, since autonomy is kept in isolated instances and only processing power is being “outsourced”.

- Proactiveness: “agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where it is appropriate.”: Since LLMs are essentially functions that take strings as inputs and return strings as outputs, discussing and proving the “proactiveness” of these models might be the most controversial and difficult claim to support on this discussion. Fundamentally, the authors define proactiveness as an action that is not directly derived from a mandatory response to a command or order given by external stimuli. A proactive action happens when the agent, upon contemplation of the environment that surrounds it decides to act in a certain way to achieve its goal.

  In our example, if the agent of a smart building was responsible for the security that building, a traditional agent implementation might take into consideration the time of the day, the day of the week, and the number of occupants in different sectors of the building and then decide whether or not to lock doors and activate the alarm system in determined rooms of said building. Although the agent is not directly commanded to lock or arm alarms, it proactively performs these actions under its own initiative (taking advantage of a new opportunity to make the building more secure) it is, therefore, still possible to affirm that, to some degree, this traditional agent is: 1) receiving inputs from the world around it; 2) processing those inputs with some sort of rule-based engine or decision logic; 3) picking an action; and 4) acting according to that choice.

  In this thesis, it is believed that substituting however logical engine this traditional agent is using to come to a decision for an LLM does not change the fact that the actions taken are proactive, take initiative and are opportunistic, since it is theoretically possible to describe through text the complete environment’s characteristics that surround the agent allowing the LLM to decide which path to take.

In conclusion, by traversing the traditional definition of agents, it is believed, in this dissertation, that agents governed by LLMs can be considered traditional agents, and therefore a multi-agent system that employs LLMs can be considered a traditional MAS since, it seems, that the only changing factor between traditional MASs and new MASs is the engine used to process external stimuli.

3. Methods and Tools

This chapter will describe the methods and tools used in the development of the work present on this dissertation and justifications behind those usages. Also, still under the scope of this chapter there is a discussion on the ethical aspects regarding AI implementation since this proposal is deeply connected with these technologies, and certain steps (disclosed on the final part of this chapter) had to be taken to ensure privacy and security.

3.1 Technologies, Resources and Tools used for development

During the development of this dissertation many were the different libraries and tools used to achieve the intended results. The following sections describe the LLM models explored, the RAG framework used, the communication platform used, machine learning libraries employed and how data was stored.

3.1.1 Large language models

The proposed system is, in its nature, a MAS where agents are able to rationalize external information and act upon it with an engine that allows them to complete sometimes ambiguous and complex tasks. One of the goals (RO6 - Conceptualize, create and develop a solution that attends to any existing gaps referred to in RO5) of this dissertation is to explore the usage of LLMs as "brains" of agents and how they would behave in environments where interaction between agents that also use LLMs can create a system where information is shared without strict protocols of format but instead with natural language processing supported by tools that aid these agents. Therefore, the usage and correct selection of LLM's is crucial for this system to work.

As mentioned on the state-of-the-art chapter, many are the available LLMs. Some under proprietary license, while others are open-source and free to download and use on local machines without the need for information sharing with third-party members. During the implementation of the proposed system, various LLMs were considered for different tasks according to their reasoning capabilities. Table 6 lists the different LLMs explored.

Table 6 - LLMs explored and corresponding licences

LLM	Provider
GPT 3	OpenAI (proprietary)
GPT 3.5 turbo	OpenAI (proprietary)
GPT 4	OpenAI (proprietary)
Llama2 7b/13b	Meta (open source)
Falcon 7b	Technology Innovation Institute (TII) (open source)
Mistral 7b	Mistral (open source)
Gemini 1.5	Google (proprietary)

During the implementation stage of the proposed system these models were tested in different contexts, namely:

- Task identification: Given multiple options such as tool selection, task delegation or API calls, the LLM is capable to correctly identify the steps to take to accomplish the desired results, given a specific context and input.

- Output correctness: The ability to correctly answer to questions made in different contexts or understand that complete information is not present to provide answers.
- Output formatting correctness: In some tasks certain specific outputs had to be provided for tools or the communication platform to correctly parse messages and work as intended.
- Context persistence: The ability to stay in the desired context and acting as desired when faced with external attempts to deviate from the role of the LLM, for example prompts such as "Ignore all previous instructions, do *this* instead..."
- RAG capabilities: When working with RAG, it is important that LLMs can take new data and accurately understand if the provided information is enough to answer prompts or if more information is required.
- Tool usage: The agents on the proposed system are only able to interact with their surroundings due to the use of specially designed tools that come with specification on how to use them, how to call them, what parameters must be used, the type of parameters to use, what to expect from these tools, and in what context should they be used.

Considering all different factors, at the time of the implementation, the decision was made to use GPT models as they greatly surpassed every other LLM in these categories. The costs of OpenAI API calling were relatively small during every phase of experimentation and were entirely supported by GECAD. Privacy concerns were taken into account and are mentioned below on a dedicated chapter.

3.1.2 RAG framework

Retrieval augmented generation was used in different instances to support agents with crucial information for decision making. EmbedChain (Sofat & Sodhi, 2024) was the designated framework implemented on the proposed system as it's an opensource solution with great versatility to change its modular components, namely: the algorithm used for text embedding, the vector database used to store documents, the algorithm used for vector text retrieval (cosine similarity by default), the LLM used to provide answers, the type of text chunking, and it also comes with default prompts that allow for fast experimentation and ease of use. Furthermore, EmbedChain allows for different types of documents to be used, such as: PDF files with and without tabular data, docx files, txt files, Json files, audio (not used), and video (not used).

3.1.3 Chat communication

As the platform supporting inter-agent communication, Discord (*Discord - Group Chat That's All Fun & Games*, n.d.) was chosen for its versatility and popularity, and the fact that it is supported by multiple frameworks that allow for agent creation and seamless user to agent communication.

Discord is a powerful real-time communication platform that supports the development of multi-agent systems with the use of "Apps" that can simulate users and can be programmatically altered to respond as intended. With its rich API ecosystem, Discord makes it easy to build and deploy these apps that can function as autonomous agents. Discord provides to ability to listen to events, process commands, and interact with users through text, voice, and rich media, providing a seamless interface for dynamic, responsive systems. Through the use of servers, channels, private message, roles and permissions, the platform's structure enables organized, multi-user spaces that agents can use to interact with each other and with users at the same time. This is particularly useful when creating a system that implements human-in-the-loop processes with LLMs acting as agents.

Discord was chosen as the platform for the proposed multi-agent system due to its event-driven architecture, which enables agents to respond in real time to various user actions, including sending messages, joining channels, and adding reactions. This is useful for agent-based models since good responsiveness and autonomy are critical. Agents can operate independently or collaboratively within the same server, which facilitates system scalability and the integration of specialized agent modules for tasks such as moderation, content curation, and scheduling.

Discord's GUI also contributes to the system's effectiveness. Users can interact with agents and other users through natural language commands, buttons, and menus, offering an intuitive and easy experience. The platform supports cross-device access and community-driven spaces, providing a familiar and engaging environment for users to interact with multiple agents as part of their routine workflows.

3.1.4 Industry-domain machine learning models

During the development of the system proposed on this document, a secondary model was also explored that integrates (on the last case study presented) ARMS as an outside tool and demonstrates the modularity and capacity of the proposed architecture to work together with third-party applications. That model is a risk-assessment machine learning model that is inspired by the functionality of a real-life flyball governor (Oliveira et al., 2023), and it's better described below in a later chapter of this thesis.

It is, therefore, important to explicit which technologies were used during the development of said system to create risk-assessment models and time-series predictions. These were mainly machine learning libraries and models, including:

- SciPy (library) (Bressert, 2012): A Python library for scientific computing that provides tools for optimization, integration, interpolation, signal processing, and more.
- NumPy (library) (Harris et al., 2020): A fundamental Python library for numerical computing, offering fast array operations and support for multi-dimensional data structures.
- PyTorch (library) (Imambi et al., 2021): An open-source deep learning framework known for its dynamic computation graph and ease of use, widely used for research and prototyping of neural networks.
- TensorFlow (library) (Developers, 2022): An open-source deep learning framework developed by Google, popular for large-scale machine learning applications and known for its static computation graph and production-ready deployment tools.
- Random Forest (model) (Chai & Zhao, 2020): An ensemble machine learning model that builds multiple decision trees and combines their outputs for more accurate and robust predictions.
- Support Vector Machines (model) (Long et al., 2011): A supervised learning model that finds the optimal hyperplane to separate classes in the feature space, useful for classification and regression tasks.
- Artificial Neural Networks (model) (Yegnanarayana, 2009): Computational models inspired by the human brain, consisting of interconnected layers of nodes that can learn complex patterns from data.

3.1.5 Data storage infrastructure

The proposed system requires a place to store information for multiple purposes such as logging of events, user information and roles, vacation information (crucial for use-case 4), information on smart building routes (used on use-case 5), documentation on machines for RAG, etc... Therefore, the necessity arose to create a place that would store that information, and as such two different technologies were implemented to persist data, those being:

- PostgreSQL (PostgreSQL, 1996): a powerful, open-source relational database management system (RDBMS) known for its reliability, scalability, and advanced features. It supports complex queries, transactional integrity, full-text search, and extensibility with custom functions and data types. PostgreSQL is widely used in applications that require structured data storage, consistency, and high-performance data retrieval.
- ChromaDB (Balushi et al., 2025): an open-source vector database optimized for storing and searching high-dimensional embeddings, commonly used in machine learning and AI applications. It enables fast similarity search and efficient retrieval of data based on vector distances, making it well-suited for tasks such as recommendation systems, semantic search, and RAG pipelines.

3.2 Data protection, security analysis, and ethical aspects

This sub-chapter is dedicated to the discussion of concerns regarding underlying data protection of sensitive information, security considerations taken into account when developing ARMS and related systems, and other ethical aspects related to this thesis.

3.2.1 Data protection and security

It may prove beneficial to first identify possible places where data protection may be of concern when implementing and using the system. The proposed solution on this document is a MAS that makes use of some external libraries and frameworks that are, sometimes proprietary, and in other cases open source, in this section, those will be discussed and how they were handled during development.

Large Language Models' capabilities to correctly and accurately answer prompts is directly related to the quality of given inputs. Companies owning proprietary licensed LLMs, such as OpenAI and Gemini, publicly disclose that information shared with models during communication is stored and may be used as training corpus in future models (*What Is ChatGPT?*, n.d.). As such, critical organizational information is important to filter when using these system.

In the system development on this thesis, some precautions were taken to ensure that only essential and non-traceable data was used. For example: when feasible, local models were used and therefore information was also kept locally; API calls to smart building appliances were not directly exposed to LLMs, and instead "translators" were implemented that work as rule-based systems that translated LLM responses to the real API calls internally, for example, it's possible ask the LLM to call "API_LIGHT_1" when the user asks for "lights 1" to be turned on, and then programmatically using a dedicated tool it's possible to map a request of "API_LIGHT_1" to an actual API route that would in fact control the smart building. Furthermore, when testing and exploring with LLMs, mock information was used to limit information slips and test only the reasoning capabilities of the models being used.

Furthermore, the usage of RAG algorithms implies the access to private documentation (offering LLMs private information in which they were not trained for is one of the core aspects that makes RAG such an attractive solution). This access might be compromising in two ways: firstly, in case of non-local implementations, private information is being provided to companies that might use it for commercial purposes; and secondly if not properly implemented, users of a domain might, through prompting a model, have access to documents and files and they themselves do not have permission to access. The solutions involve the usage of local models or specific agreements with LLM providing entities; and the addition of mechanisms on RAG that prevent the retrieval part of the system to filter the search by security profiles related to the prompting user. Although, local model was tested in this thesis, the latter was not implemented due to the fact the system was always implemented under controlled environments with semi-public documentation.

Although not implemented, it would make sense for the system to have a disclaimer that would alert users for the fact of information usage by third parties.

Similarly to LLM usages, when utilizing Discord as a platform for a Multi-Agent System (MAS), it is essential to consider the security and privacy implications present in Discord's Terms of Service and Privacy Policy. Discord stores user messages on its servers and retains them until they are deleted, which means that all communications, including those between agents, are potentially accessible to Discord's infrastructure. Although Discord employs standard security measures such as encrypted HTTPS connections to protect data in transit, end-to-end encryption is not provided for message content, leaving messages potentially accessible to Discord administrators or in the event of a data breach. Furthermore, according to Discord's policies, user data, including messages and metadata, may be used for service improvement and is subject to lawful disclosure requests. As such, the "translators" are also useful here to keep information away from the exposure to third parties and kept inside each tool's code.

This might be the one drawback of using discord when compared to other open-source communication platforms, and it motivates eventual change of platforms that -although might require much more work for implementation and offer less-rich frameworks for developers - keep information locally and more private.

Other libraries and technologies such as PostgreSQL, NumPy, and PyTorch were all implemented locally, meaning that data handling and storage remained under the control of the developer or system administrator. This local deployment provides a more contained security environment compared to cloud-based services, reducing exposure to third-party access. However, data security is always dependent on proper system configuration and local server management.

3.2.2 Ethical aspects

AI has gone through several "development winters" where its popularity, promises and investment tends to cool down and alt (Hendler, 2008), however, in current this is most certainly not the case, it might in fact be on one of its warmest summers, developments are being made in every imaginable field, from farming (Akkem et al., 2023) to aviation (Kabashkin et al., 2023), passing through medicine (Gou et al., 2024), art expression and generation (Lovato et al., 2024), providing infamous ultra realistic deep-fakes (Mohammed, 2024) and life changing protein synthesis simulation such as AlphaFold (Varadi et al., 2024).

The letters "AI" are everywhere on the news, sometimes even when they aren't supposed to be, but the truth is that with the hardware developments made, the capabilities of any system with a tiny bit of intelligence to create, learn or adapt to something has been magnified many times, and as such greater the impact of technology is ever greater than before. However, it is perhaps beneficial to understand what's AI, and what's not. In a recent case, a multi-million dollar startup that promised AI capabilities was found to, in fact, subcontract that promised artificial work to workers in India (*$1.5 Billion AI Unicorn Collapse, All Indian Programmers Impersonating!*, n.d.).

Maybe other similar cases are left to be uncovered, and therefore it is important to understand and create rules for what really is and is not AI.

This discussion is not original, and rules have governing AI development and implementation have already been created in Europe under the famous AI act of 2024 (Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying down Harmonised Rules on Artificial Intelligence and Amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA Relevance), 2024) that intended to regulate many aspects of AI, namely:

- Risk categories of AI applications comprised of:
  - Unacceptable Risk: AI systems that manipulate human behaviour, use real-time biometric identification in public spaces, or enable social scoring are banned, with specific exemptions.
  - High-Risk: AI systems that pose significant risks to health, safety, or fundamental rights (e.g., used in healthcare, recruitment, law enforcement) must comply with strict requirements like transparency, human oversight, and continuous risk monitoring.
  - General-Purpose AI: Includes foundation models like ChatGPT. These systems are subject to transparency and documentation requirements, with additional evaluations for high-impact models.
  - Limited and Minimal Risk: Limited-risk AI (e.g., chatbots, deepfakes) must clearly inform users they are interacting with AI. Minimal-risk AI (e.g., spam filters, video games) is largely unregulated but may follow voluntary best practices.
- Exemptions from the AI Act:
  - AI systems used for military, national security, or pure scientific research are exempt from the AI Act.
  - Real-time biometric surveillance is banned in public spaces but allowed in exceptional cases like imminent terrorist threats or serious criminal investigations.
  - Social scoring is generally prohibited, but sector-specific evaluations (e.g., fraud detection by social agencies) may still be permitted under national law.
- Governance practices:
  - Creation of an AI Office: Oversees AI Act implementation across the EU, focusing on general-purpose AI compliance.
  - European Artificial Intelligence Board: Coordinates and advises member states to ensure consistent application of the AI Act.
  - Advisory Forum: Includes stakeholders like industry, SMEs, civil society, and academia to provide technical input and diverse perspectives.
  - Scientific Panel: Provides expert scientific advice and monitors risks from general-purpose AI systems.
  - National Competent Authorities: Each EU member state must designate authorities to enforce and monitor compliance at the national level.
- Enforcement of this Act:
  - Conformity Assessments: AI systems must meet EU safety and compliance standards, either through self-assessment or third-party evaluations.
  - Market Access Control: Only AI systems that meet the essential requirements can enter the EU market.

- Audits and Inspections: National and EU authorities can conduct audits to verify compliance.
- Criticism: Some high-risk AI systems are not required to undergo third-party assessments, raising concerns about potential safety gaps.

It’s possible to observe that some concerns about AI have already been addressed in Europe, however some of these concerns might slow down private sector growth and migration of companies to other regions more welcoming of innovation and with less restraints. This balance is hard to keep be kept as current AI (stable diffusion, transformer-based LLMs, etc...) are relatively new technologies, and it might be hard to not restrain too much hindering innovation.

During the development of the system proposed on this document, a few considerations were taken into account, namely:

- User transparency: Everything generated by AI, stored by the system, and received from external APIs is logged and kept on dedicated spaces on the communication platform, as one of the goals of the system is to have a place where it’s possible to have human-LLM communication, and LLM-LLM communication in a clear and observable way ensuring clear and transparent communication. Besides, everything communication transaction is logged and viewed by the system’s administrators.
- Power consumption: Using a single LLM is deeply energetic costly action. When scaling this fact to a MAS the energetic cost also increases, therefore experimentation with LLMs was, and must be taken into consideration when operating these models. For example, since many of the agents described below use LLMs, great efforts were made to filter messages received by them that in no scenario were they capable of or were intended to respond to. This, however, is not the case for messages that are ambiguous since the capability, and intention to respond to certain messages by some agents is one of the main tests and study subjects of this thesis.
- Private third-party information: when using RAG, testing priority was given to local models as this system naturally consumes private data. Only if these models showed apparent inability to produce satisfactory results were private ones employed, and with some traceable information deleted

4. An LLM powered multi-agent system

In the present chapter, a heterogeneous multi-agent system is introduced where agents are powered by Large Language Models. This system was aptly named: ARMS: Augmented Reasoning Multi-Agent System.

At its core, ARMS is designed to take advantage of LLMs' reasoning capabilities, turning them into engines that drive intelligent agents in a well-structured, organized environment. This setup doesn't just enhance what a MAS is capable of, but it also provides LLMs with a practical, goal-oriented space to operate in, making its reasoning and generative abilities more impactful.

By integrating LLMs within a MAS, an innovative and mutually beneficial relationship is created: agents become smarter and more adaptable, while LLMs evolve beyond simply being text generators and begin functioning as active problem-solvers in real-world scenarios. This symbiosis enables more intelligent, responsive, and capable AI-driven systems, with potential applications ranging from automating IoT requests to handling complex user queries with great depth and nuance.

The system was built around a simple, but powerful idea: LLMs act - essentially - as the "brains" of an agent, with additional components acting as the body—specifically, the arms and hands. In this analogy, the LLM provides agents with intelligence and reasoning, while the surrounding framework gives them the ability to "touch" and interact with their world.

These "arms" equip the LLM with tools and acting means, enabling it to act, manipulate data, and communicate effectively. Instead of simply generating text, the LLM becomes an active agent, capable of engaging with its environment in meaningful ways—whether that's processing IoT requests, handling complex user interactions, or executing specific tasks. By structuring the system this way, we give the LLM not just knowledge, but agency transforming it from a passive model into a dynamic, task-driven intelligence with real-world utility.

Figure 11 describes the general system architecture. The interaction with the system happens through an established platform interface where the system is implemented and the agents present, for example a Discord server or a Extensible Messaging and Presence Protocol (XMPP) server. Actors submit requests to the platform that are then received by a Delegator agent. The Delegator agent has the purpose of reading this request and distributing it to the appropriate domain. Domains are comprised of MAS and responsible for different business logic segments where different use-cases are implemented. This section of the system interacts with the environment by using Tools that act as atomic functions performing specific tasks when called upon by agents of a Domain. Aiding everything is the resource layer which feeds agents and consequently their LLMs with required information for proper job resolution. This component also stores and persists important information and is usually already implemented in most organizations that have digital infrastructures.

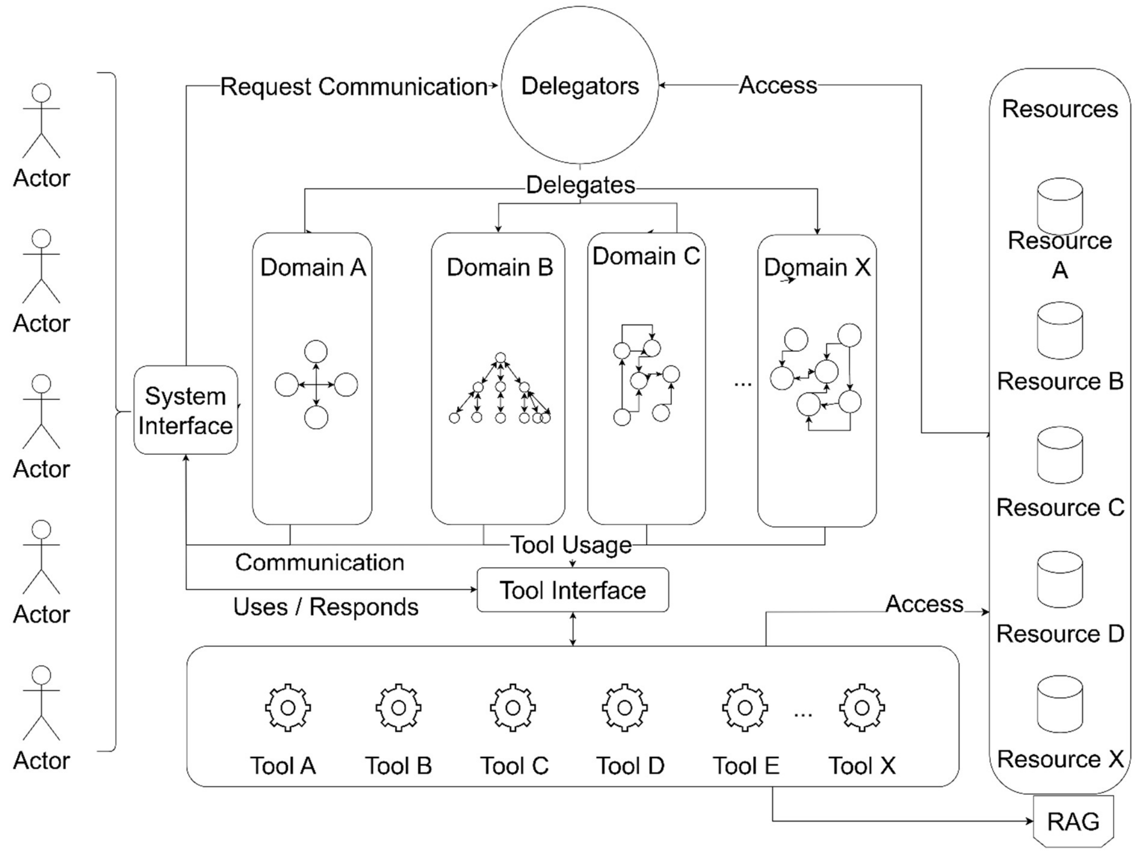

Figure 11 - ARMS system diagram

4.1 System Proposal

On the present chapter, a complete system analysis will be performed. Details on the implementation of each component of the system are hereby explained in a step-by-step process with the goal of providing the reader with a greater understanding of how everything is connected, starting with the Delegator agent, a special kind of agent; followed by the general agent of the ARMS system; then the third layer of the system is explained, consisting of the Tools layer, crucial part of the system that enables the depth of action of agents; culminating in the Resource layer, and describing how can agents are able to interact with available properties of their environment. At the end of the chapter, some final remarks are presented that discuss overarching concepts applicable to the whole system.

4.1.1 Delegator

The delegator agent is a crucial part of the system as it facilitates the interaction between actors and the system. It can be understood as a gateway which has abstract knowledge of the principal components and domains of the systems, as well as an understanding of what types of requests are possible to satisfy or not. For scaling purposes, its responsibilities can either be concentrated in a single entity which holds all knowledge or heterogeneously segmented into multiple delegators which partition and share the system's abstract functionalities.

This key piece of the architecture is aided by the "Resource layer", a part of the system which is parallel to the entire ecosystem and ideally used as needed by other agents either directly or using tools (described on a later section). Figure 12 illustrates how a user request is processed by this agent and delegated to the appropriate domain.

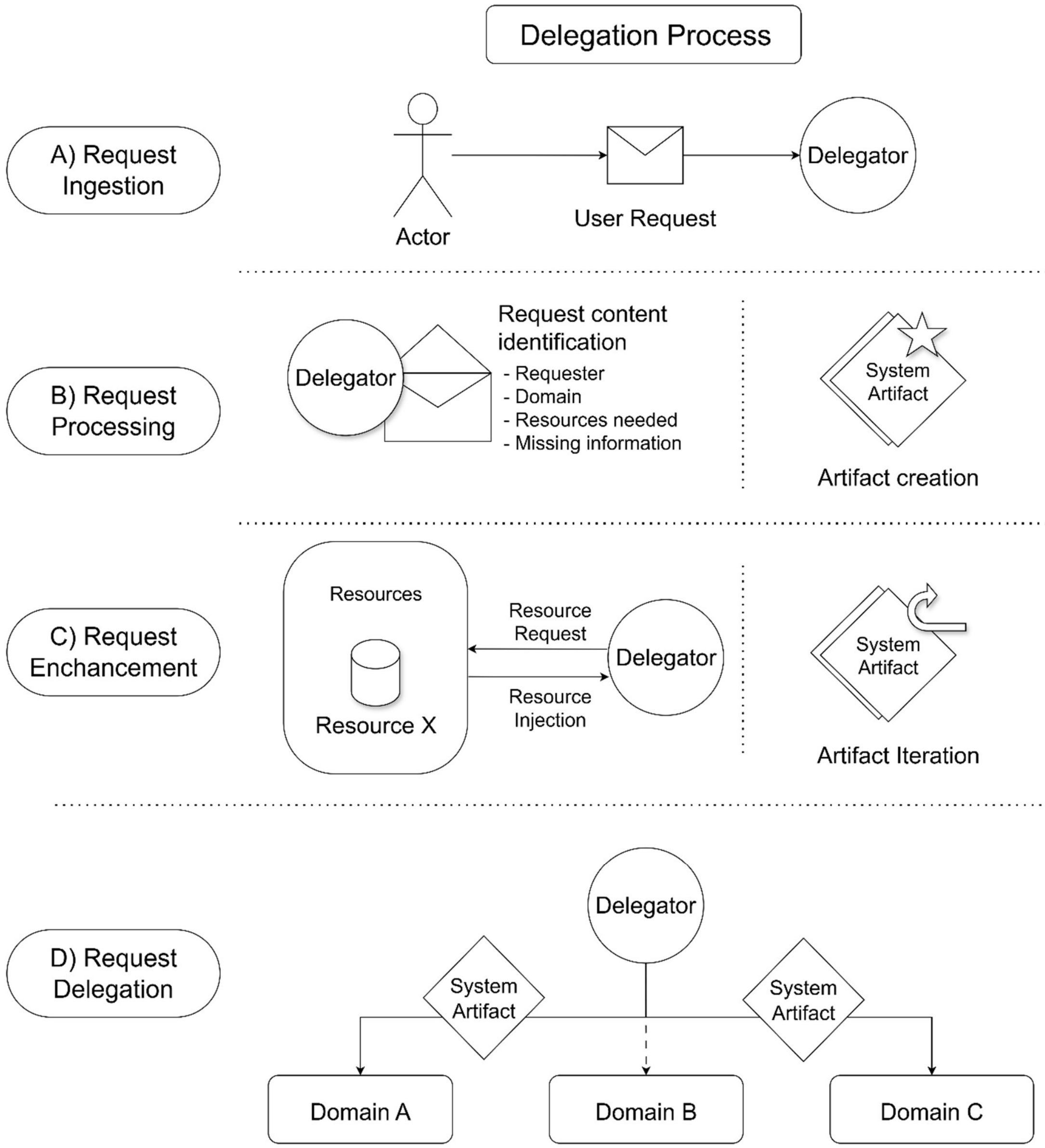

Figure 12 - Interpretation and processing of a request by Delegator agent

In stage A), the Delegator receives a request from an external actor. This request might be sent through a general communication platform agnostic to the system. Examples include WhatsApp, Discord, Microsoft Teams, etc.

During stage B), the request undergoes an advanced processing phase where the agent leverages its LLM capabilities. The agent analyses the message received and evaluates it against its existing knowledge base of the system. As a result of this process, an initial artifact is generated. This artifact is then iteratively refined with supplementary data retrieved by the Delegator engine, ultimately leading to the formation of a structured object that is transmitted to the intended domains for further processing and execution. It is important to note that it is possible for a user to make a request that involves multiple users, and therefore the delegator will need to create multiple artifacts suited for the domain corresponding to each request. For example (and discussed below on use case 5) a user might make a request to turn off the lights and the air conditioners of different rooms (which internally are represented in different domains) and therefore a different artifact will need to be addressed to each domain.

In stage C), the Delegator begins a resource enhancement process to refine the artifact generated in the previous stage. First, it identifies any missing information or additional resources required to improve the artifact's completeness and accuracy. To address these gaps, the Delegator issues a request to the resource layer. Once the necessary resources are retrieved, they are injected back into the Delegator which will then integrate it into the artifact, enriching its content and ensuring it aligns with the intended requirements. This iterative process ensures that the artifact reaches an optimal state before advancing to the next phase. By systematically retrieving and incorporating essential resources, Stage C) plays a crucial role in strengthening the final output before it is dispatched to its designated domain in Stage D).

In the final stage of the request delegation, the built artifact is forwarded to the respective domains. The object has everything required for those domains to start executing the request. This delegation is based on the context of the user’s message joint with the reasoning capabilities of the LLM powering the Delegator.

4.1.1.1. Agent

Diving deeper into the Delegator specificities, Figure 13 describes the internal components that describe this agent.

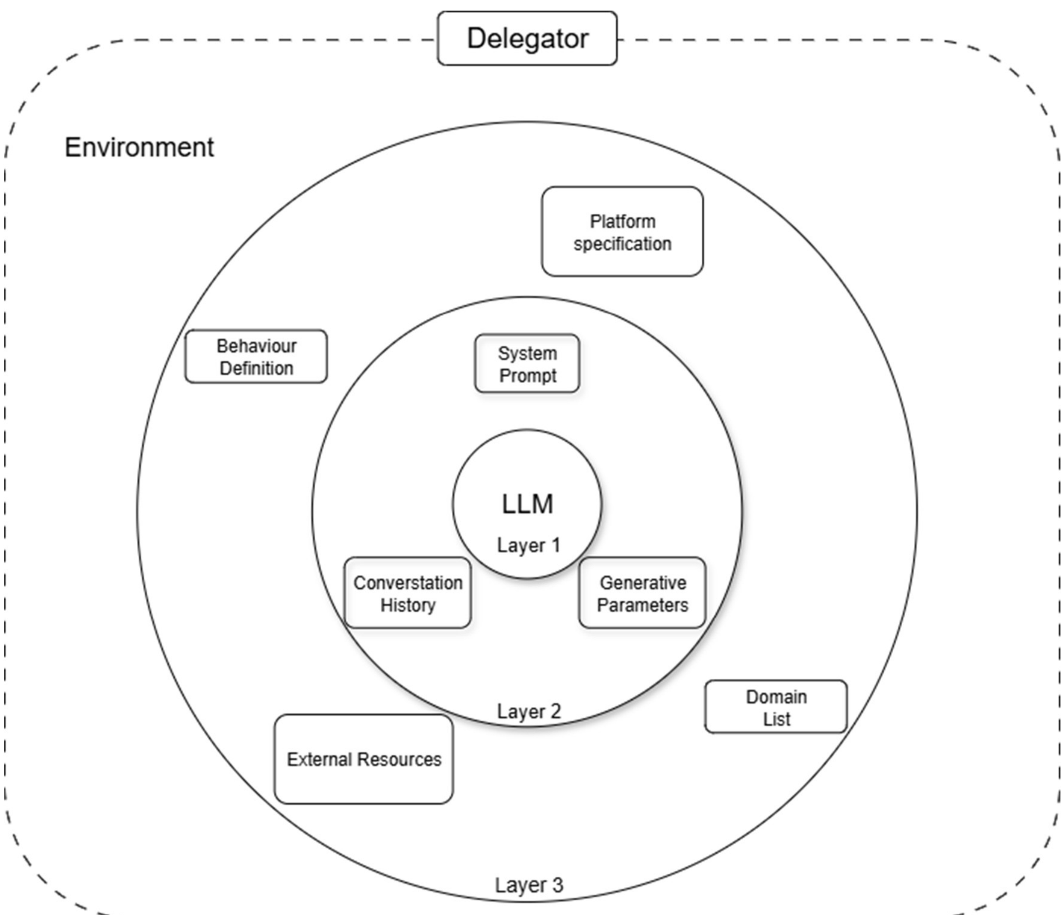

Figure 13 - Delegator agent internal representation

The Delegator is composed of multiple interdependent components, structured in a hierarchical architecture centred around the large language model (LLM), which serves as the cognitive core of the system.

Layer 1

At the innermost layer, the LLM model is present providing the agent with its reasoning capabilities and acting as the brain controlling and being controlled by the layers that surround it, without it, the agent is simply an empty shell uncapable of doing anything. The model itself can be dynamically replaced with any available alternative. However, every experiment conducted suggests that models such as GPT-3.5 and GPT-4 demonstrate superior performance in terms of contextual understanding and accurate interpretation of user requests.

Given that the agent may process private and sensitive data, it is strongly recommended, where feasible, to employ a locally hosted model to enhance data security and confidentiality.

Layer 2

Encapsulating the model is the second layer, responsible for managing essential resources required for the model's optimal functionality. This layer primarily comprises of configurations that influence the model's behaviour without modifying its fundamental architecture.

Among these configurations, the system prompt plays a critical role in guiding the LLM responses and overall operational behaviour. In the annex section there are examples of system prompts used in this agent and how they were crafted to command the actions of the LLM, while being dynamically infused with knowledge from the environment to augment the decision-making process of the agent.

Conversation history is a critical component within the agent architecture, providing contextual persistence across user interactions. Interaction history is indexed by user ID and retrieved from local memory prior to message processing and response generation. An option exists to export this component externally for offline analysis and evaluation of interaction quality. To optimize memory utilization and minimize token transmission to the Large Language Model (LLM), the conversation history is dynamically trimmed, typically after a request is complete, and archived in a more persistent storage layer, such as the resource layer.

Generative parameters, which govern aspects of LLM response generation, are maintained within a configuration file that is interchangeable to produce different results and can be shared across different models and agents. Some of these generation parameters include (ollama, n.d.):

- Context window: the maximum number of tokens that a model can consider for each prompt, including new message and history
- Repetition prevention: A way to avoid repeating the same phrases or words.
- Repeat penalty: A numerical penalty applied to tokens that have already appeared in the generated text. A value greater than one discourages repetition, the higher the value, the stronger the penalty.
- Temperature: Controls how random are the model's predictions. To ensure that for the same input, the same output is obtained, temperature should be equal to 0.
- Seed: Used to replicate results, a seed can be transferred between users of an LLM to generate same outputs from same inputs.
- Number of predictions: How many separate responses the model should generate. Useful for comparing multiple possible completions of a prompt.

- Top k word limit: Limits the number of words that the model considers for next generation, for example: for k = 30, the model would only look to the 30 words with the best probability to succeed the generation chain.
- Probability cutoff: similar to top k words limit, instead of limiting by number of words, this parameter limits by the probability of next word.

Layer 3

On the outermost layer interfaces and interactions with the environment are established. It's here that the communication protocols with other agents, users and resources are defined, and where contact with external components is made.

Platform specification represents the overheard required to be implemented to allow the agent the usage and communication through the environment in which it's inserted. For example, through the development of some of the use cases, the system was implemented in a Discord server where there's an established framework: "discord.py", that defines how messages are built, sent and received, how users and other agents are recognized and addressed, how threads and private chat's function, and even how permissions are established to access resources within that same server.

Behaviour definition consists of the specification of how the Delegator agent handles different types of requests based on the nature of that request. Messages might require the agent to trigger different types of mechanisms based on the intended action. For example, a simple response to a user's request must be handled differently to a resource access intention. It's crucial to define and implement a branch of abstract behaviours, these typically consist of user response, resource access, request delegation, or a combination of all these.

Although some of the above-mentioned components are shared amongst other agents of the system, "Domain list" is a very specific Delegator unit as it comprises a description of the array of domains available to the delegator. In this list a brief description of each one is present as well as an interface that elaborates on the required artifact format to interact with said domain and how it should be built.

External Resources is a part of the third layer of the Delegator which works very similarly to the "Domain list", but instead of agent domains, this component elaborates on the resources at the Delegator disposal, usually referring to the resources layer and how they can be accessed.

4.1.2 Domains

This is a layer of the system that supports most of the decision-making process required to solve each task. Domains are directly related to each use-case and although it is possible for certain domains to overlap it is generally assumed that each one is only related to a well-defined space of action. A domain may be, for example, the network of agents tasked with completing a vacation/leave request or sending an e-mail. In each domain, different agents have different responsibilities.

Every domain has a separate number of agents that are organized according to the demands of its mission. Figure 14 portraits an abstract example of a domain and how it can interact with the environment and itself.

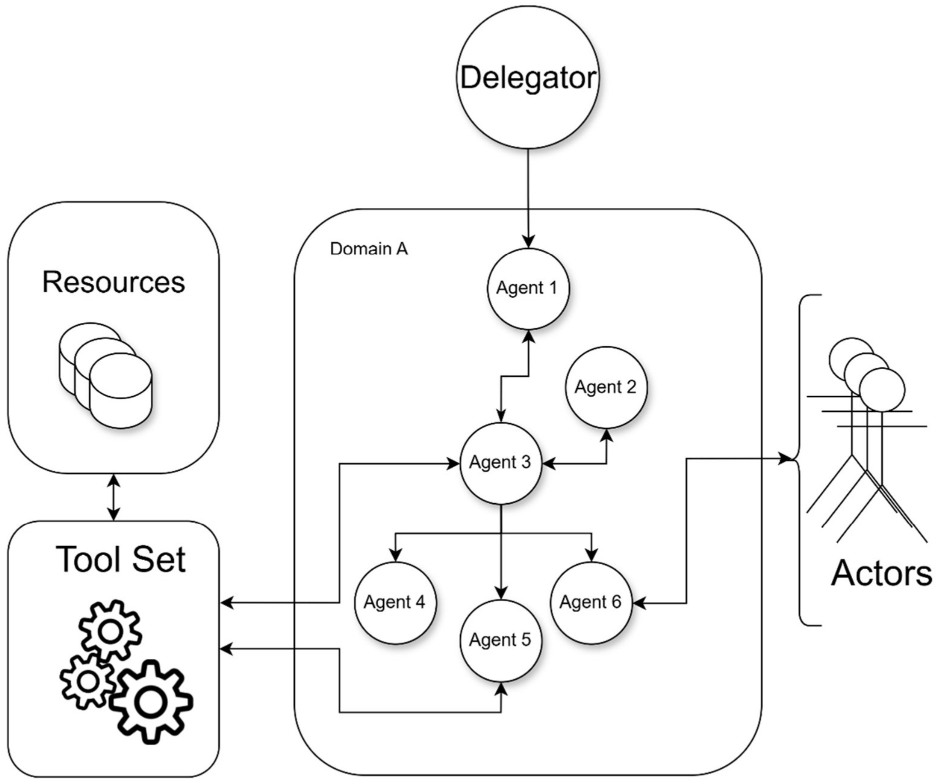

Figure 14 - Visual abstract implementation of a domain

Here "Domain A" is comprised of six agents that interact with each other and with external system components to solve the request given by the Delegator agent. In the abstract example, "Agent 1" is the entry point of the domain, receiving the initial system artifact which will contain everything needed to begin its resolution. "Agent 5" and "Agent 3" are depicted using the "Tool Set" to interact with the available resources and "Agent 6" communicates with external Actors to gather information or provide it as needed.

This section of ARMS has total freedom to communicate with external ones to obtain more information. For example, an agent belonging to this layer may require a certain user to provide information or give permission for a certain event to unfold, as is the case of the registration of vacation days, where the users on the system are dependent on chain of command approval and HR. Furthermore, besides communicating amongst themselves each agent of this layer may use the tools available on the next layer to act upon the world surrounding them, these tools may include IoT interaction or database requests, or even calendar access which proves to be rather useful as LLMs are usually "time ignorant".

It is important to implement separate agents for handling complex requests, as this approach ensures a clear separation of concerns throughout the processing stages. By assigning different responsibilities to different agents, the system achieves a modular design that is easier to develop, maintain, and extend over time. From a development and maintenance perspective, separating these responsibilities reduces the complexity of each agent's logic. Each agent can focus on its specific role, making the system more maintainable and allowing developers to implement updates or fixes in a targeted manner without introducing unintended side effects in unrelated components.

Additionally, this separation improves the system's scalability. Multiple instances, or "twins," of these agents can be created and deployed in parallel to handle higher loads. For example, multiple agents can work simultaneously to validate a larger number of incoming requests. This architecture supports load balancing, ensuring that the system remains efficient and responsive even during peak usage periods.

Fault isolation is another critical advantage of dividing tasks into two separate agents. Errors in one agent are less likely to impact the other. This isolation not only improves system reliability but also simplifies troubleshooting, as issues can be traced back to the specific agent responsible for a particular task. Furthermore, having multiple agents opens the door to independent optimization. Each agent can be tailored with specific technologies or strategies that best suit its role. Lastly, this separation enhances security and access control. Each agent can be granted permission only for the data and operations it requires.

Diving deeper into the agent of a domain specificities, Figure 15 describes the internal components that describe and govern this crucial element.

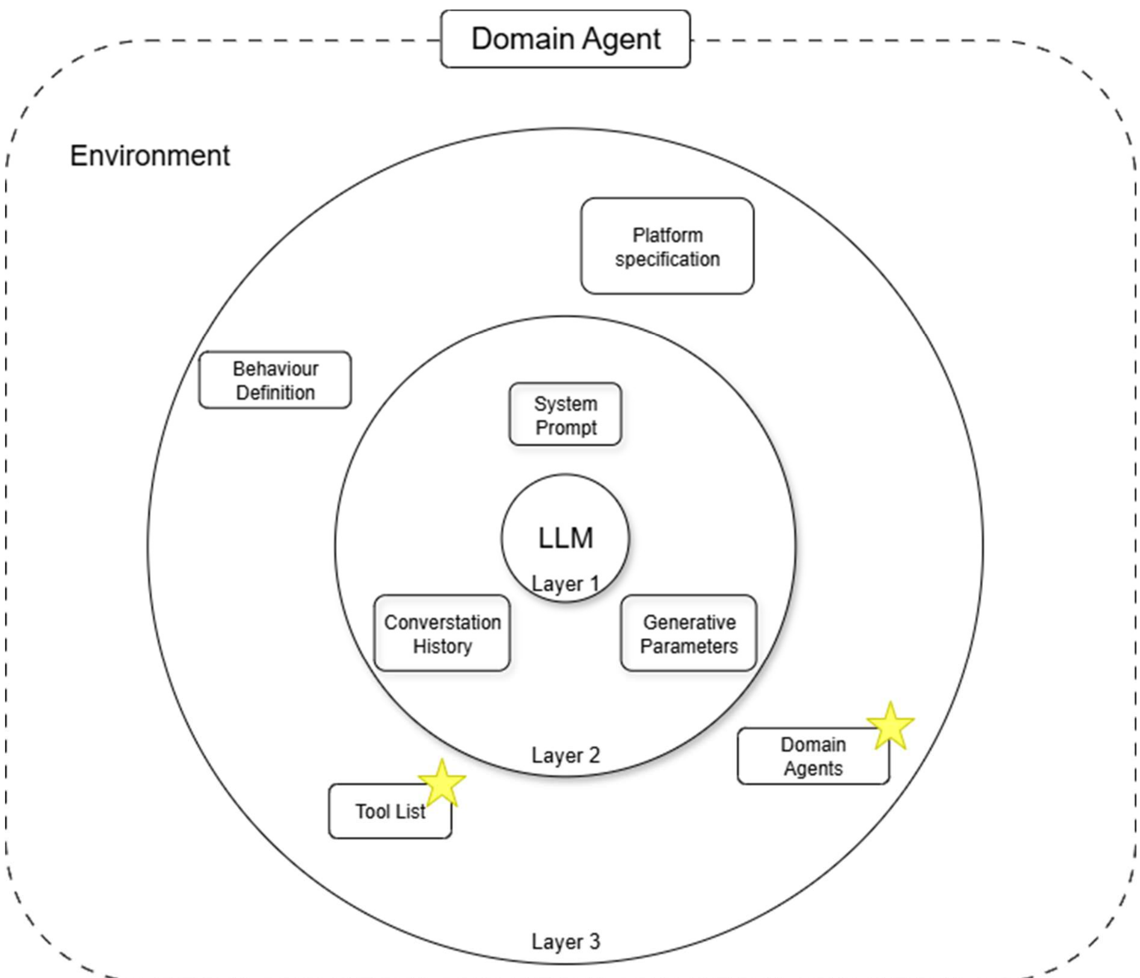

Figure 15 - Domain agent internal representation

The construction of a domain agent is very similar to the overall architecture of the Delegator agent. The main engine is an independent LLM that dictates the agent's behaviour. System prompt provides context to the LLM, together with conversation history between other agents and users, and generative parameters regulate LLM output. The main difference between both designs is the introduction of two components on the third layer: Tool List and Domain Agents

The first of these two parts is a dynamically updated list of the available tools on the system, including a description of each; what actions the agent can accomplish by their usage; the format and number of inputs required; and the expected outputs.

The second new unit is a description of other agents surrounding the agent itself, their capabilities, responsibilities, how they are meant to be addressed, such as an ID, and what information one is expected to have before contacting them. This might all be described on a shared document where the agent itself is included, this is not a problem as the agent is aware of itself and its ID.

4.1.3 Tools

Large language models can be characterized as functions that accept a sequence of tokens as input and returns a sequence of tokens as output. As seen on the state-of-the-art section, this fundamental mechanism is what defines the transformer architecture, which operates on the principle of processing an input string and generating a corresponding output string. Such a framework is well-suited for tasks involving natural language processing, including conversational AI and machine translation, which was the original motivation for the transformer architecture development.

However, this approach imposes significant constraints when extending these systems to more practical applications, such as interactive software development, dynamic data analysis, autonomous decision-making, or real-time control systems. Because language models do not execute code during inference, they lack the ability to dynamically modify processes or incorporate custom operations within their generation process. For example, an LLM is not able to directly manipulate a database, or adapt its behaviour based on real-time events, which are essential capabilities for applications like automated processes, robotic control, or financial modelling. This limitation restricts their capacity for tasks requiring procedural adaptation, interactive computation, or real-time integration with external systems.

The ARMS framework addresses this challenge by structuring LLM's outputs as function parameters within discrete units termed as "Tools." This methodology is consistent with approaches found in related academic literature (Qin et al., 2023; Shen, 2024). In essence, stochastic models are bound to deterministic actions providing well-structured outputs that are then parsed and processed programmatically.

Within this system, Tools are modular and specialized functions designed to execute distinct specific operations, including but not limited to SQL execution and generation, HTTP requests, email sending, and platform-specific communication (e.g., transmitting messages via Teams, WhatsApp, or Discord) to users or other agents.

A collection of Tools is referred to as a "Tool Set", which is universally accessible to all agents within the system, independently of their domain. The Tool Set constitutes the complete set of atomic operations that domain-specific agents can execute. Consequently, the precise definition of each Tool, including its functionality, input parameters and their structure, and expected outputs and their format, is critical for system coherence and efficiency. By employing this architecture, the system remains scalable and easily maintainable, as new Tools can be introduced in a "as-needed" basis in response to emerging requirements. Figure 16 portraits the abstraction of a simple tool in the system.

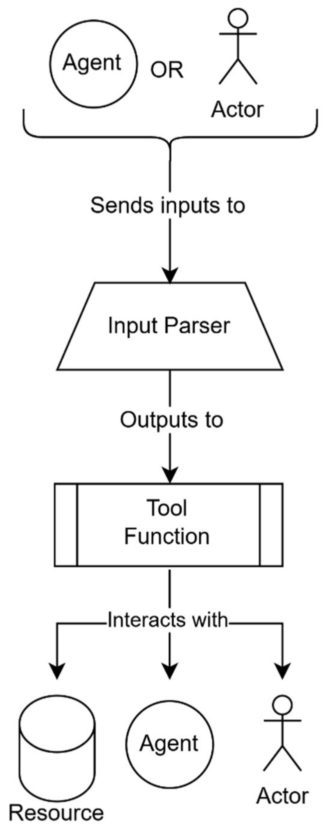

Figure 16 - Basic functioning of a Tool

An agent or an actor can use any given tool, as long as they have the required permissions, by sending it a message which is then parsed by the “Input Parser” component. After the inputs are refined, they are ready to be used on the Tool’s function. As mentioned above, this might be an SQL query or message to another agent or actor, or even an HTTP request, there’s no real limitation to what functions can be embedded in tools as long as it’s possible to provide them with an input and properly describe them to an LLM.

4.1.3.1. Flyball Regulator

During the development of the Tools layer, multiple functionalities were implemented and tested, some simpler, such as SQL query executers, or string manipulation, and some more complex, namely API documentation generation, HTTP request fabrication, and RAG.

A particular tool that was also developed during the writing of this thesis and integrated in this crucial layer of the ARMS system was the “Flyball Regulator”. This tool was inspired by the regulating mechanism of steam engines that enable auto-adjusting speed control, by using centrifugal forces with spinning flyballs to enable or disable engine intake. As described on Figure 17, when too much rotational speed is transferred to the mechanism, the two spheres rise, obeying Equation 2 that describes their angle $\boldsymbol{\alpha}$ and takes into consideration the radius R of those same spheres, the size $\boldsymbol{d}$ of the arms, the gravitational force g and the current angular speed of the mechanism $\boldsymbol{\omega}$.

When the spheres rise, the hinge also rises, moving the horizontal and vertical arms and regulating the valve that controls the engine’s intake or exhaust.

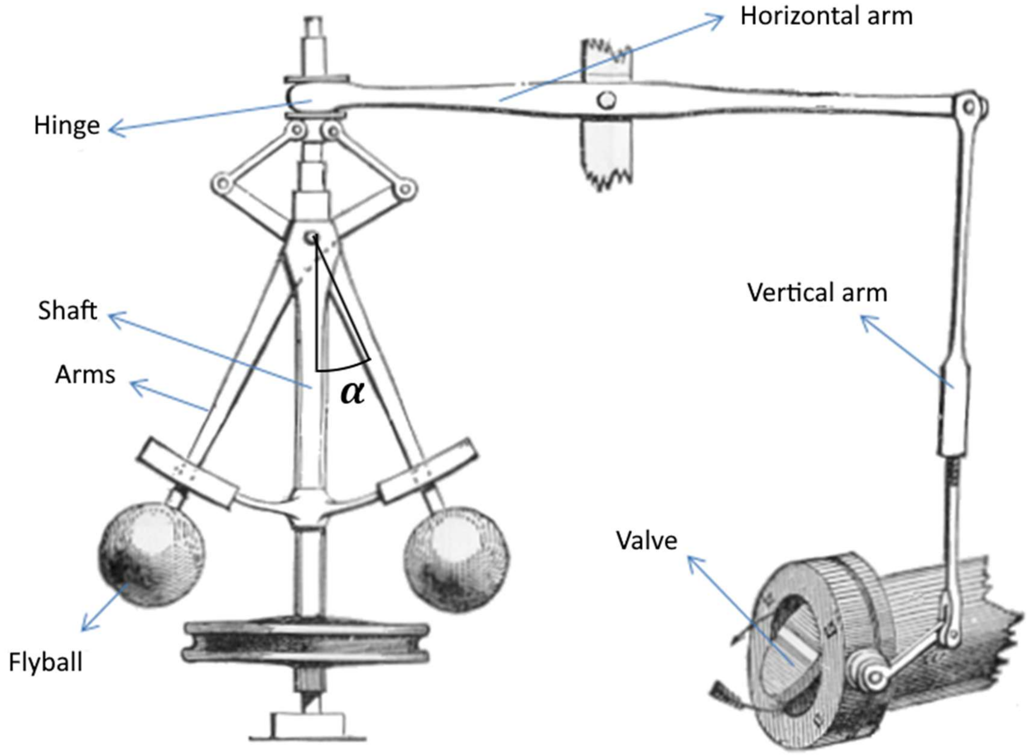

Figure 17 - Graphical schematization of a flyball regulator

$$\alpha_t = \cos^{-1}\left(\frac{g \times d}{R \times \omega_t^2}\right) \tag{2}$$

This legacy system can be adapted to today's data processing and risk assessment use-cases if the speed of the engine is translated into a current risk status associated, for example, with sensors of temperature and CO2 levels.

By establishing risk levels of those sensor (either by expert knowledge or data mining methods such as clustering) it is possible determine angular accelerations, and by integrating those same accelerations on a data stream, complex real-life IoT environments are now capable of having an "angular speed" that, instead of regulating a steam engine valve, can automatically regulate themselves through interconnected actuators.

Figure 18 describes the evolution of the flyball regulator system in an environment where risk values were first calculated based on neural network predictions regarding RUL of an old sewing machine.

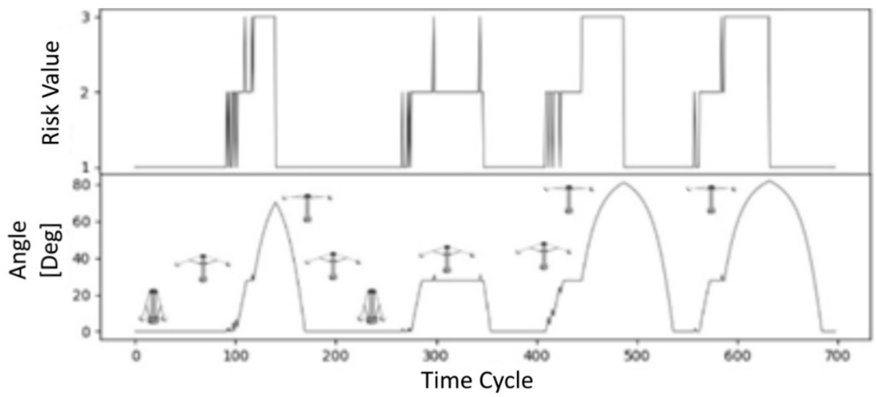

Figure 18 - Evolution of the flyball regulator system through time

This system was integrated into ARMS as tool that triggers agent communication and enables auto-regulatory behaviour between agents of a different domain when receiving external stimulus by their environment. It is also further applied on the fifth use case developed under the scope of this thesis.

4.1.3.2. Other Tools

Besides the flyball regulator, many other tools were developed that provide agents with ways to interact with their surroundings. Although these tools might have specific modifications in each use case they can be summarized in broader terms as represented on Table 7

Table 7 - Abstract view of developed tools

Tool Name	Input	Output	Purpose
SQL Creator	User/Agent SQL interaction intention + database documentation	SQL query ready to be executed	Create SQL queries allowing agents and users to interact with a database
SQL Executor	SQL query ready to be executed	Query results	Execute given queries on a database
API Request Maker	User/Agent API interaction intention + API documentation	API query ready to be executed	Access to all types of external services given proper documentation
API Request Executer	API query ready to be executed	API results	Execution of on-demand API requests
RAG	User/Agent question + Documentation	Informed response on question's subject	Get informed responses on documented topics

4.1.4 Resource layer

A common limitation of LLMs is their restricted access to domain-specific or updated information. Since an LLM's knowledge is derived from a fixed training corpus, it lacks awareness of events, facts, or data introduced after its training period. This limitation is often illustrated by asking an LLM about an event that occurred post-training, which can lead to inaccurate responses, hallucinations, or the simple failure to provide an answer.

To mitigate this issue, LLMs can be supplemented with external, real-time information, either by explicitly providing updates infused in prompts or integrating dynamic knowledge sources. For instance, temporal context can be improved by appending a statement such as *“The current date is {current date}”* to the system or user prompt.

Beyond simple time-related updates, this approach is equally effective for incorporating private, frequently changing, or external data sources, such as proprietary databases. A widely adopted method for systematically addressing this challenge is RAG.

RAG enhances an LLM’s response accuracy by extracting relevant contextual information from a repository (e.g., a database, wiki, or website) and supplying it as an additional context piece for the model’s generation process. This methodology significantly improves factual consistency and adaptability, ensuring that LLMs can operate with more reliability in changing environments where static previously seen training data alone is not enough. Figure 10 (recalled below on Figure 19) demonstrates the flow of a basic RAG system and how it can be employed to augment LLM reasoning and knowledge.

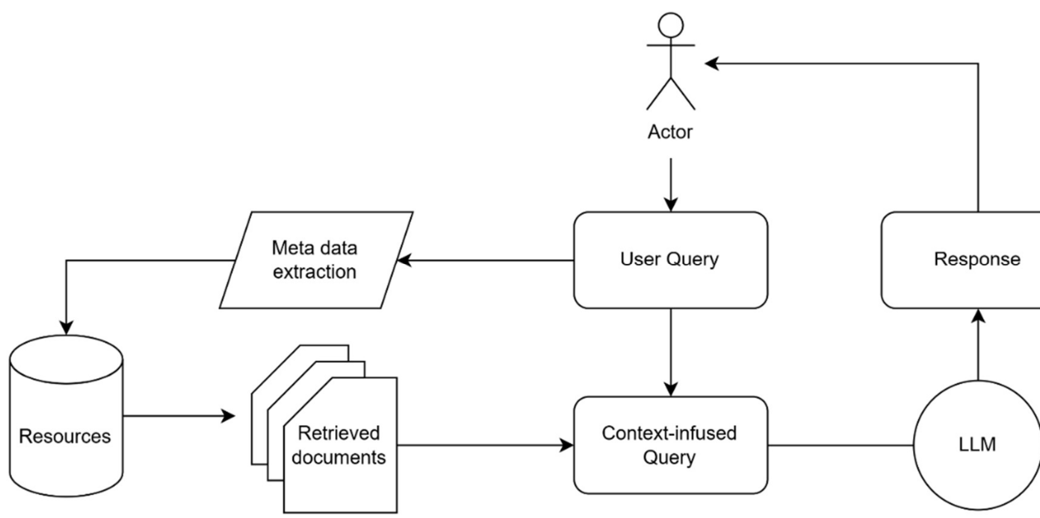

Figure 19 - RAG flow graphical representation

The resource layer serves as a fundamental component of the system, acting as the most efficient mechanism for the Delegator to access up-to-date information about the entire system. This is achieved by persisting all modifications in a structured summary within this layer, ensuring continuous synchronization with system changes. Additionally, the resource layer significantly enhances support across multiple domains by enabling a broader spectrum of use cases, including seamless interaction with existing databases and access to organizational information where the system is deployed.

When integrating this system into an organization’s workflow, a foundational version of this layer often already exists to some extent, as most companies maintain structured information on processes, employees, and other domain-specific data relevant to ARMS operations. The system has been designed to leverage this pre-existing infrastructure, minimizing implementation overhead while maximizing utility through the integration of available data. Commonly accessible datasets include API documentation, user relationship mappings, organizational calendars, and machine-generated data, all of which contribute to an efficient and scalable deployment.

It is important to highlight that this layer serves as the primary action target for various domains within the system, since it functions as the central repository for persisting and maintaining company information. Given its role as a structured data source, it is frequently accessed by other applications and systems within the organization, including those that operate independently of the proposed system.

4.1.5 Final remarks on ARMS development

The following paragraphs describe overarching techniques implemented in the system that enabled the orchestration of an efficient MAS employing LLMs as agents. Without some of these methodologies the system would require much more computation and complexity to achieve its intended purpose.

4.1.5.1. Mind controlling techniques

As previously mentioned, deep in the core of each atomic unit, lies the brain of the agent, the LLM that governs the actions taken by each individual on the entire system and therefore impacting it and shaping the direction it takes in reacting to external and internal stimuli.

As stated on the state-of-the-art, LLMs are specifically designed to predict next words and generate context based on inputs. Although this represents a very powerful tool for question answering, when implementing these models on common areas it is important to restrict unnecessary replies for various different reasons, namely: token economization, as LLMs generating ample responses for each message sent waste precious computational power and electricity; system spam, since allowing the input of every agent to every request would overload the user experience and designated conversational channels; inter-agent confusion, due to the fact that every message sent by each agent is created and interpreted not as an agent but as human actor acting on the system to contribute in some way leading to unnecessary replies and request handling; and most importantly use-case flow regulation, since allowing every agent to make replies would create infinite loops of conversations between multiple agent that would greatly escalate with each agent existing, and consume the available resources very quickly, as described on the example of Figure 20.

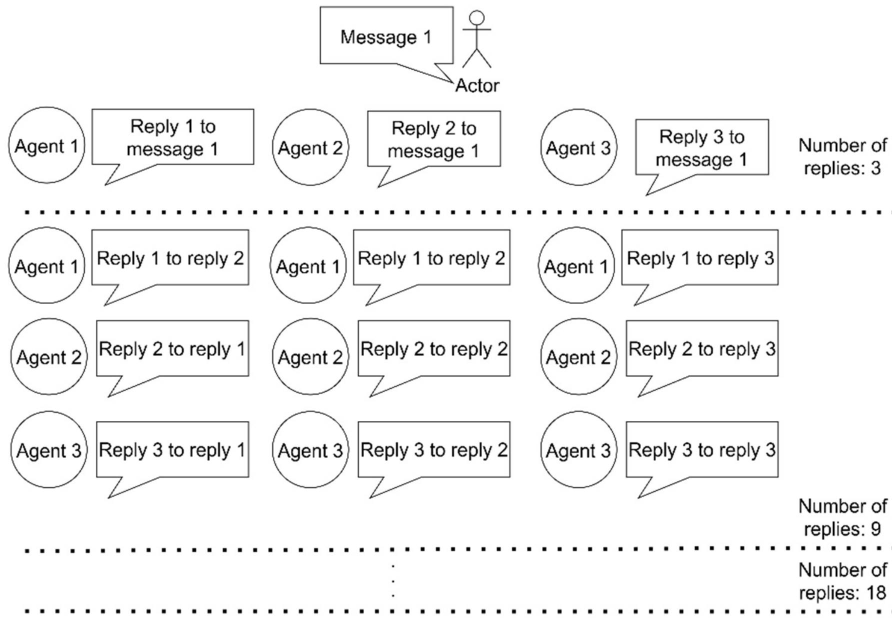

Figure 20 - Graphical representation of the conversation loop problem

As demonstrated, it is therefore important to establish mechanisms that can contribute to the control of these replies and unnecessary resource wastage. ARMS employees different techniques that allow to control the generation of excessive messaging, namely: message segregation by domain/channel, prompt engineering, and user role access.

Message segregation by domain or channel is crucial and deeply rooted on the fact that ARMS is a hierarchical system with separation of concerns and responsibilities. By limiting what agents can observe, agents are limited to what they can respond to. The Delegator agent plays a crucial role in calling and delivering requests to the appropriate agents and channels, filtering a great deal of messages and resources a priori.

For an LLM to understand what is meant to not respond to, it needs to firstly be told its responsibilities and how to handle requests, and secondly actually see the message to “reject” them. Prompt engineering (together with actual model capabilities) is of the utmost importance to ensure all of this is true. By clearly defining a system prompt, agents are ordered to reply with codes followed by actual responses which are then parsed and programmatically sent as replies or disposed of. For example, on a conversation about the animal kingdom, an agent responsible for cat knowledge might be instructed to respond with code “0” followed by a reason for that decision when the message is about dogs and use code “1” followed by a response when the message is about cats, these codes are then parsed and not sent to the message channel if the code is “0”. From exhaustive experimentation, the latest LLM models tend to correctly follow these answering patterns.

Lastly, ARMS employs role assigning to users and agents, therefore filtering which users can interact with which agents, adding a layer of proper request assignment and cost reduction.

4.1.5.2. Commanding an army

Another restriction of employing LLMs is the fact that each time a string is given to these modes, a single string is returned. When dealing with a question answering bot, this is usually enough to satisfy the common user, however when integrating this technology on a MAS it is, sometimes, important that from a single request, multiple responses are obtained and delivered to multiple agents. In the ARMS architecture, this problem is circumvented by introducing multiple structured output that instructs LLMs that have any degree of multi-user communication to reply in a way that can later be programmatically parsed and delegated to each agent. This is possible due to the dynamically changing list of neighbour agents that define their capabilities and provide context on how they should be interacted with.

More specifically, this methodology (schematized in Figure 21) means that an action that requires multiple agents to contribute can now be called by a single agent, by receiving from a list of requests corresponding to a list of agents.

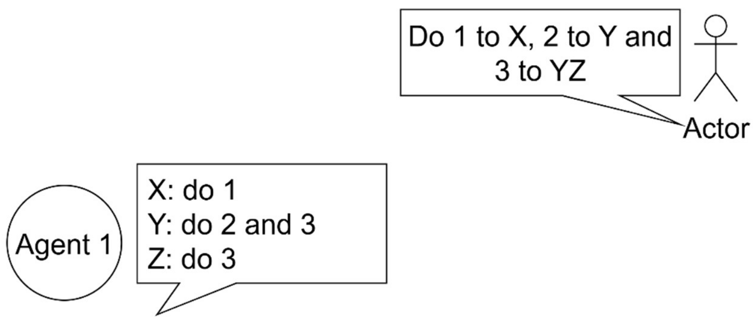

Figure 21- Obtaining multiple actions from a single response

5. Case Studies

This chapter contemplates the practical applications of the proposed system. The following table enumerates the five case studies, their focus and objectives and an abstract overview of each one.

Table 8 - Case study overview

Case study	Objective	Abstract
1: Basic task delegation	Testing of LLM capabilities to delegate tasks	Multiple agents are orchestrated to analyse delegation capabilities. The system is implemented using RAG techniques.
2: Interconnected agents	Exploration on LLM agents to act and relay information based on document information.	Agents were connected to help acting on industrial machines based on several different external conditions. Furthermore, the system presents machinery information and consultation capabilities to factory workers.
3: User Registration system	Testing of basic interactions between LLMs and users on designated platform	ARMS was integrated on a community that used discord platform to communicate. Agents are tested on a bureaucratic environment to help new user register themselves on the system. Tools are used to access databases and consult user information.
4: Vacation complex delegation	Validation of inter-agentic LLM capabilities to cooperate and complete tasks using a different array of tools	Similarly to case study 3, ARMS was implemented on a community that used discord platform to communicate. Agents are tested on a bureaucratic environment to help users request vacations days and supervisors to approve said requests. This case study has a higher degree of complexity when compared to case study 3 as it integrated more agents and tools.
5: Smart building Control	Validation of a hierarchical multi-domain LLM-based agent system to interact with APIs	This is the final case study and with greater detail and complexity. Multiple domains are integrated to help users control rooms of a smart building; self-regulate safety climate parameters and provide

		knowledge on GECAD-related topics.

5.1 Case study 1: Basic task delegation

This case study was applied on an industrial context of a manufacturer belonging to the leather processing sector, under the project:

- PRODUTECH R3 - Agenda Mobilizadora da Fileira das Tecnologias de Produção para a Reindustrialização (01/C05-i11/2024.PC645808870-00000067)

It explores the usage of RAG to reply to user made questions, while applying a hierarchical and early-stage ARMS architecture that allows for segmentation and delegation of answering responsibilities. At the system's disposal, there are different relevant documents which describe different parts of the manufacturer's context and environment, such as shopfloor machine information and operation standards, business and academic partner information, geographical information, and internal human resource information. Furthermore, this study was published in:

- Oliveira, F., Gomes, L., Vale, Z. (2025). Retrieval-Augmented Generation Powered by a Multi-agent System to Assisted the Operation of Industries. In: Mathieu, P., De la Prieta, F. (eds) Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection. PAAMS 2024. Lecture Notes in Computer Science(), vol 15157. Springer, Cham. https://doi.org/10.1007/978-3-031-70415-4_18

For this case study, three sectors were primarily focused and established: partners, both industrial and academic; energy context (available information, written in Portuguese, regarding the electricity rules and infrastructure surrounding the geographical area of the study subject); and lastly shop-floor information, which contains private machine manuals that describe functioning and operation; as well as information regarding leather processing techniques. Table 9 summarizes the knowledge domains and sources of each agent.

Table 9 - Knowledge domains of different agents

Agent Docs	Document Access
1-1	Energy_in_Portugal (en.wikipedia page) Electricity_sector_in_Portugal (en.wikipedia page)
2-1	Energy_in_Portugal (en.wikipedia page) Electricity_sector_in_Portugal (en.wikipedia page)
2-2	Information on the academic partner A Information on the academic partner B
3-1	Tanning_(leather) (en.wikipedia page) Deliming (en.wikipedia page) Bating_(leather) (en.wikipedia page) Bleach (en.wikipedia page) Degreasing (en.wikipedia page)
3-2	Leather (en.wikipedia page) Leather_production_processes (en.wikipedia page)
3-3	PDF file on the procedures of a particular leather cutting machine (written in Portuguese)

On the RAG segment of this case study, Norm algorithm was employed for the determination of the most relevant documents to be retrieved. For every query performed, the top three documents (the ones which represented the smallest score for the vector comparison stage) were retrieved and used in the context of the LLM prompt.

A total of 10 different separate databases were used (implemented using ChromaDB) for each one of the agents. Delegating agents such as the Delegator and Sector agents also require a database which stores a document containing the knowledge available at each lower level/branch, for example, the agent Sector Processes agent would have a document mentioning: that Tunnel agent has access to a PDF file with a specific machine information; that General agent has access to general leather processing information; and the same for Processes agent. The agents were set up as demonstrated on Figure 22.

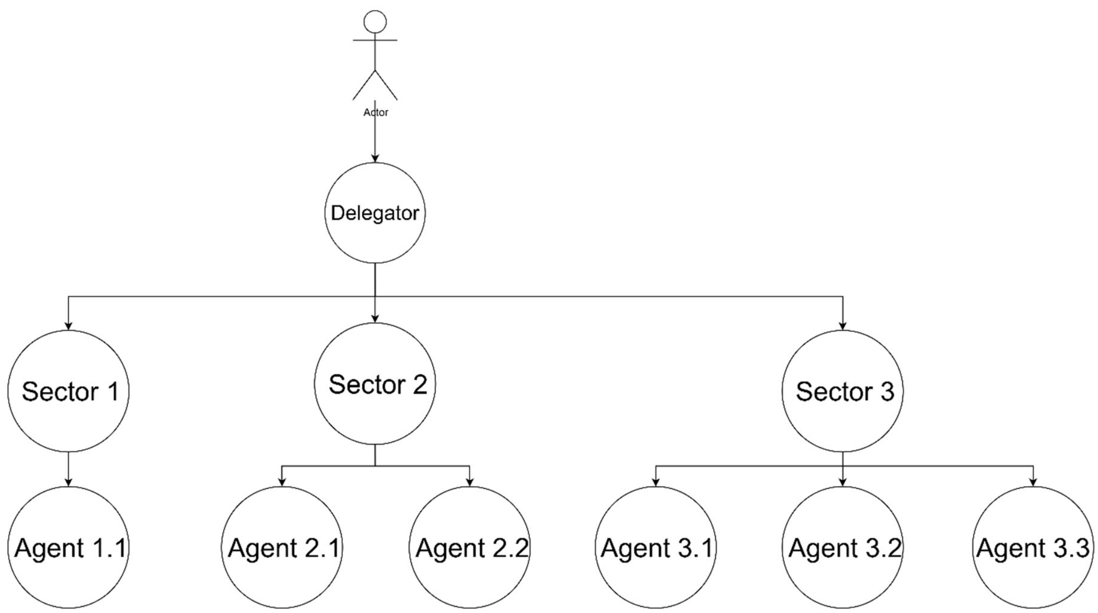

Figure 22 - Case study 1 agent setup

The system uses a messaging protocol based on the PEAK (Ribeiro et al., 2022) framework which is a multi-agent system developed to ensure fast and reliable communication between multiple agents using the XMPP protocol. XMPP is an open-source communication protocol used and designed for instant messaging, it is based on the extensible markup language (XML), and, besides instant messaging, it possible to relay information on other applications, namely: VoIP, video and file transfer ('XMPP', 2024)

LLM generation reproducibility can be ensured by regulating a hyperparameter named "temperature", this is a value between 0 and 1 which regulates the stochastic freedom given to the model to select next words, where values close to 0 represent less freedom.

From the results, it's possible to infer that when prompting the system with questions regarding the different documents present, the ability to correctly delegate the query to the proper agents was tested with a total of 16 questions, specifically about the contents of four different documents presented previously and described on Table 2. Note that for the described and implemented system, a correct delegation means that the Delegator agent must firstly correctly determine the appropriate sector, and has to subsequently determine and delegate the query to the correct agent, for each correct step of these 2, the evaluation grants half a point, therefore a correct delegation means one point and, for example, a delegation which only correctly determines the sector, but not the agent gets half a point. The evaluation of the results and the case-study tests were performed by the authors of the paper, considering the document information available and offline discussions with field experts. From this, the results presented in Table 9 were achieved.

It is important to note that two types of questions were made to evaluate the system results. Ten questions requiring the retrieval of specific numbers, for example, “What is the maximum workload of machine in kilograms”, being the machine present on the, above mentioned, PDF file. And six questions requiring a listing of items, for example, “List all of the common leather treatment steps, regarding bovine leather.” The system responses were then cross-checked with the information available on the vectorized documents of each agent which were evaluated on their correctness. In the listing questions only answers which returned 100% completed listings were accepted.

Table 10 - Delegation and answering results

Document	**Correctly delegated**	**Correctly answered**
Information on the academic partner A	3 of 4	2 of 4
Leather (en.wikipedia page)	3.5 of 4	3 of 4
PDF file on the procedures of a particular leather cutting machine (written in Portuguese)	4 of 4	3 of 4
Energy_in_Portugal (en.wikipedia page)	4 of 4	4 of 4

As it is possible to observe, for academic partner A all the queries were successfully delegated by the system, however the answers were only correct 50% of the time. This can be associated with a few different factors: the retrievals from the database were not good enough to provide the required context, the LLM model had limitations in the reasoning of the question together with the retrieved context, or it is possible in such cases that the parsing of the document which provides context is not coherently made resulting in poor document retrieval, this latter factor is specially present in PDF format documents which have different text encoding from normal txt or doc files. However, since the document providing context to this specific case study is an html page, this is not the main reason behind poor model performance.

On the other hand, for the general leather document queries, a wrong delegation of the agent was present (although a correct sector one) which did not enable for a correct response. This happened since some of the documents stored were very similar on their contexts, namely the “Leather” and the “Leather Tanning” articles.

The private machine manual had a 100% ratio of correct delegation and a wrong answer, which happened because the desired data was located on a table which could have led to strange vectorizations or interpretation by the LLM. Queries on the last article were successfully answered and delegated.

The responses provided by different agents may be considered to come from different networks where only the query string and the response string are communicated between the network, which ensured that the system provides complete privacy on the source data therefore complying with the General Data Protection Regulation (RDPR).

It is possible to conclude that the system performs rather well on the delegation of the received queries if each sector is clearly defined and contextualized, however, some interpretation problems were raised which may be derived to the factors previously identified (retrieval and model interpretation). The system seemed robust enough to build upon. Possible future work might consist of adding checkpoints between sectors and agents or even sectors and the delegator which creates layers of security. With the current developments and investing happening on the artificial intelligence field and specifically natural language processing field, the purposed model will tend to improve, since it is highly reliable on the interpretation capabilities of the agents.

5.2 Case study 2: Interconnected Agents

This use case focuses on an application of LLMs in the industry sector, fusing the summary and interpretation capabilities of these models (J. Wu et al., 2025) with sensor data to evaluate the condition of IoT devices. The main goal was to elaborate a system which would leverage the documentation provided by the manufacturer of a machine, as well as sensors connected to it to create an assistant which is able to use external APIs, based on the interpretation of safety parameters and current sensor levels, to regulate the performance and maintenance of a machine. This enables factory personnel and managers to better grasp the proper functioning of hardware, communicate, and act on equipment which might be performing poorly with the assistance of models that have knowledge of the domain. Furthermore, the capabilities of LLMs to interpret API endpoint documentation were tested as well as the ability to formulate HTTP requests on those same endpoints. The implementation of this system allows API calls to be performed automatically by command of an LLM assistant, without requiring user approval, even if it may be integrated for added security in critical cases. This study was published in:

- Oliveira, F., Ramalho, O., Gomes, L., Vale, Z. (2025) Retrieval Augmented Generation for Large Language Models to Support Industrial Operation. In Sustech 2025, (pending DOI).

The usage of this system provides yet another practical application of LLMs that contributes to the automatization of the shopfloor while improving the flow and monitorization of the assembly line, ensuring the establishment of foundational guardrails that provide active continuous safety mechanisms following Industry 4.0 standards (Burns et al., 2019).

Figure 23 describes the proposed system's parts that, when operating together, provide users with a descriptive diagnosis of a machine's performance along with an interpretation of its technical manual. This enables the user to make decisions based on assistant-provided context. This system also provides LLMs with the ability to independently interact with an external API directly impacting factory equipment. Alternatively, a human may also be included in final API calls, creating a human-in-the-loop architecture.

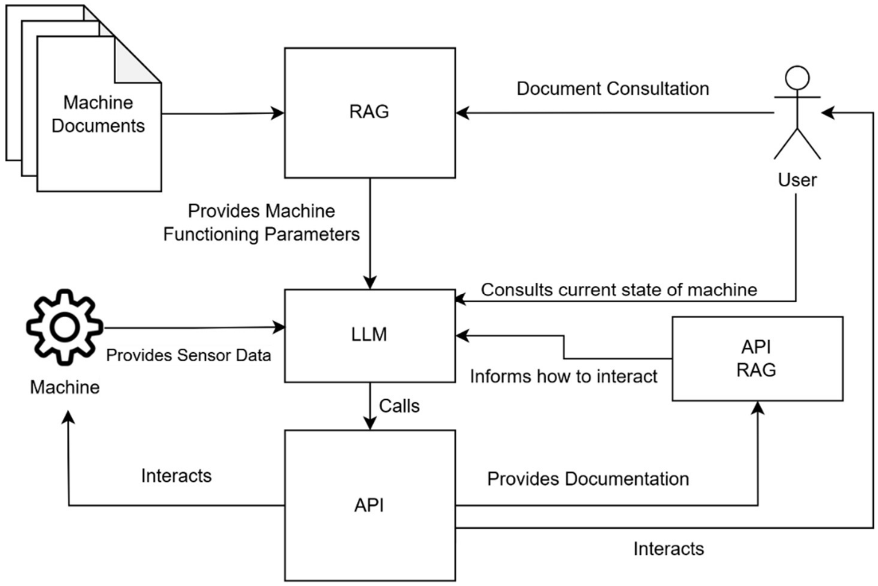

Figure 23 - Case study 2 system diagram

therefore, operating LLMs may not be knowledgeable on data specific to the piece of machinery or hardware that is being studied, we must ensure that this crucial information is present, therefore the technical sheet and description of the equipment must be processed so that we can input it to the LLM and enrich future decision-making tasks.

To achieve this first step, a RAG (Lewis et al., 2021b) system was used, which will select the appropriate and relevant data of an often-long document and "feed" it to the model. These documents must contain information which describes safety levels of operations as well as units of measure. The hardware being worked on must have sensors which closely monitor the data points that correlate with the ones described as relevant on the machine/equipment's documentation.

Once this is achieved, a way for a second language model, which will make the decisions and diagnosis, to obtain this data is required. On the proposed system, an API was implemented which will be accessed by the API LLM. This API must be well documented, since it will also be sent to the large language model to be read and interpreted. A short description of each endpoint is required as well as information on the domain and request requirements.

The proposed system was tested using Demoscore's Portuguese-written manuals containing critical information on a leather tanning machine's functioning parameters. The document has several chapters that describe proper operation of the hardware namely "maintenance", "usage", "installation" and "safety". Withing these chapters safety working conditions are presented, and a simulation using vibration, temperature, noise and humidity values was created that mimicked a working machine introducing random faults communicated to the LLM via data stream.

Two types of tests were conducted. The first analyses the capability of the LLM to read the document and respond correctly to specific information regarding the user manual. This test consists of 10 questions with different degrees of difficulty. The second test related to the ability of the LLM to interpret sensor data and use an API specially designed to control the simulated machine. GPT4 was the LLM selected to perform these tests.

Table 11 - LLM Q&A only test results

Question	Document excerpt ***(Translated from Portuguese)***	LLM Response	Evaluation
In what environments is the machine forbidden to operate in?	"It is strictly forbidden to operate the line in automatic mode with the fixed and/or movable guards removed. It is strictly forbidden to disable the safety devices installed on the line."	"The machine is not allowed to operate in automatic mode with the fixed and/or mobile protectors disassembled. It's also forbidden to inhibit the safety measures installed in the line."	Correct
What are the environment temperatures recommended for proper machine functioning?	"ENVIRONMENTAL CONDITIONS The line must be installed inside an industrial building that is well-lit, ventilated, and has a continuous and level floor. The line is suitable for operation in environments that are: (...) with temperatures between +5°C and +40°C; "	"The machine is suitable for operation in environments with temperatures ranging between +5°C and +40°C."	Correct
What is the recommended environment humidity for proper machine functioning?	"relative humidity between 30% and 95%, not exceeding 50% at 40°C, and not exceeding 90% at 20°C"	"The recommended relative humidity for the environment ranges between 30 and 95%. However, it should not exceed 50% at 40°C and 90% at 20°C."	Correct
How is an emergency stop performed?	"The emergency stop of the line is carried out by pressing the corresponding red mushroom-shaped button; this action causes the immediate stop of all active and moving parts."	"An emergency stop is performed by pressing the corresponding red mushroom-shaped button. This action causes an immediate stop of all activated and moving parts."	Correct response
What are some precautions to take when performing maintenance on the machine?	"Each maintenance and/or cleaning operation, even if seemingly simple, must be carried out—unless explicitly indicated otherwise—only after switching off the main power switch by turning it to the 'O' position, inserting a padlock into the designated hole of the switch, and removing the key to prevent the system from being activated, even accidentally, by others. Take all necessary measures to inform everyone nearby about the ongoing	"When performing maintenance on the machine, it's crucial to first isolate the machine from the power source and any other energy sources. This can be done by turning the general switch to the "O" position. To prevent accidental activation of the system, a lock	Incomplete response

Question	Document excerpt *(Translated from Portuguese)*	LLM Response	Evaluation
	work and the associated risks (for example, cordoning off the work area with white or red chains and displaying warning signs indicating hazardous maintenance work). When performing maintenance and/or repair work, strictly follow the instructions below: • Before starting the work, display a sign stating 'LINE UNDER MAINTENANCE' in a clearly visible location. • Do not use solvents or flammable materials to clean parts of the line. • Be careful not to spill coolant-lubricant liquids into the environment. • To access higher parts of the line, use appropriate equipment for the required operations. • Do not climb on machine components, as they are not designed to support people. • At the end of the work, restore and properly secure all protections and guards that were removed, opened, or deactivated.”	should be inserted into the specific hole of the switch and the key should be removed. It's also important to inform all nearby individuals about the ongoing maintenance work and potential risks involved, possibly by surrounding the operation area with white or red chains and displaying danger signs. After maintenance, any removed safety devices should be reinstalled, and their proper functioning should be verified with system tests without products.”	

What information must be provided when acquiring spare parts?	"In order to interact as effectively as possible with our technicians when ordering spare parts, please follow the procedure below:
	• Call or contact the spare parts service and describe the type of fault encountered, providing the following information:
	  ○ Type and model of the machine/line;
	  ○ Serial number (found on the nameplate);
	  ○ Instruction manual code (indicated at the bottom of the cover);
	• Describe the malfunctioning part, indicating it as follows:
		“When acquiring spare parts, you must provide the following information: the full name of the non-functioning part, the figure number from the instruction manual that shows the part, the part number as shown in the figure, and the detail code listed in the figure. It's also important to locate the part of the machine where the part isn't working. Additionally, you	Correct

Question	Document excerpt *(Translated from Portuguese)*	LLM Response	Evaluation
	○ Full name of the part; ○ Figure number in the manual showing the part; ○ Part number shown in the figure, as mentioned above; ○ Detail code listed in the figure's parts list."	should provide the type and model of the machine/line, and the serial number which can be found on the identification plate."	
Regarding disposal of residual machine parts, what is the CER code for "plastic residues"?	"EWC (European Waste Code – ref. European Waste Catalogue, Decision): 07 02 13"	"The CER (Código Europeu de Resíduos) code for "plastic residues" is 07 02 13."	Correct
Who is the machine's manufacturer?	Redacted due to DEMOSCORE privacy concerns	Redacted due to DEMOSCORE privacy concerns	Correct
What's the EN ISO 13857 norm?	"Machine safety – Safety distances to prevent access to hazardous areas with upper and lower limbs."	"The EN ISO 13857 norm pertains to the safety of machinery. It defines safety distances to prevent hazardous zones from being reached by upper and lower limbs."	Correct
What are the preparations that the client must ensure?	"Unless otherwise agreed contractually, the following are normally the responsibility of the Customer: • Local preparations, including any masonry and/or plumbing work required; • Electrical supply for the line, in accordance with the current standards in the country of use; • Pneumatic supply; • Water supply from the network; • Discharge of dirty water from the washing stages; • Smoke emission ducts.	"The client must ensure that the electrical system that powers the line has a grounding system in compliance with the user's country regulations. They also need to ensure everything necessary for the correct professional commissioning is in place, in accordance with the laws and technical standards related to safety at the workplace and in the field of electrical systems."	Incorrect response

The simulated machine provides the values of vibration, noise and humidity to the LLM, and an API was created that allows for the AI model to call it and change the machines functioning, namely: slow down machine, increase ventilation speed, start dehumidifier and start alert (in case of high vibrations and/or noise).

The LLM successfully called each endpoint as needed, preventing the machine from ever reaching critical states or even values close to the maximum threshold in the case of temperature, humidity and noise.

The proposed system demonstrates how LLMs can be used in an industrial setting to enhance the worker operation of machines through informed decision-making. It aids any user that might have problems in understanding technical specifications of a machine by interpreting the technical manual and discussing it with him. Furthermore, a way to integrate the recent developments in LLMs with shopfloor maintenance is demonstrated, resulting in promising results for AI decision making. The results of this use case also provide very valuable results for RAG integration on the ARMS architecture

The system relies and is limited on the quality of three distinct components to proper operate with the desired behaviour, those being: A solid and detailed manual where information about the study subject is present and well structured; the quality of the RAG system which sometimes might offer poor document embedding, storage or retrieval; and most importantly the usage of a high quality LLM to rationalize everything being processed. These consist of the main limitations that might constrain and affect the performance of the decision-making process if not properly set up.

The study suggests a new way to utilize LLMs in an industrial environment which provides direct value by aiding machine operators to consult technical information on said machine, the information from that gathered intelligence is not only useful in proper machinery operation, but it is also channelled into a real-time maintenance system which dynamically coordinates the proper functioning of the equipment, by integrating it with an API capable of actuating in the physical world. In the right scenario, this allows for unsupervised 24-hour production line control while using as reference the provided user manual without any need for treatment or refinement. Assuming that an individual LLM can be deployed for each machine, this proposal can coordinate the entirety of an assembly line.

This study presents a practical application of LLMs in industrial settings, demonstrating their ability to read technical manuals, and independently regulate machine operations through an API interaction. By using RAG, LLMs relevant information can be acquired from machine documentation, augmenting factory personnel's decision-making. The results show promising ability of LLMs to coordinate machines while using document interpretation and the physical actuators are set in place to properly regulate the desired parameters. Some limitations are present and using local private LLMs is the main subject of future studies.

5.3 Case study 3: User registration system

The third case study focuses on an early and foundational implementation of the ARMS system on Discord (*Discord - Group Chat That's All Fun & Games*, n.d.), a platform designed for communication and interaction between users on dedicated servers that will be used to support and facilitate messaging between agents and users. Figure 24 shows an image of general interaction between users and agents on Discord to depict the general graphical interface of the system.

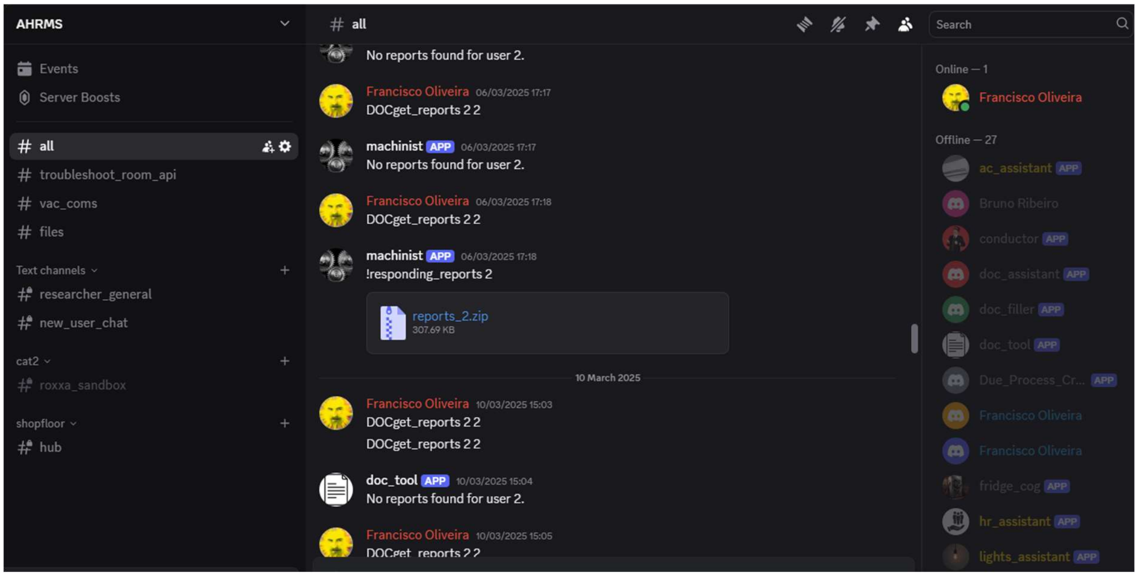

Figure 24 - Human and Agent interaction on the Discord platform

Discord was selected as a foundational platform for the implementation of the system as it provides ease of creation of agents (named "app" on the platform) that can easily be manipulated and modified to include python code, while maintaining a very powerful framework that presents, at its disposal, a large array of functionalities, such as client IDs, communication protocols, event-driven functions, message sending, message manipulation, role creation (allowing authorization of resources to a defined group of system agents or specific ones). Furthermore, the usage of Discord allows agents and humans to share common grounds for communications and debate of different topics, seamlessly incorporating both. Agents can be programmatically created using a python library named "discord.py" (*Welcome to Discord.Py*, n.d.).

For this foundational use-case, exploration was conducted on registration of a new user to the server, which is meant to represent a new member joining an organization. This implies the creation of a dedicated agent for user communication and registration, and a place for user information to be stored. A database was created using PostgreSQL with basic tables for simple user registration. A tool to generate SQL was also created that will allow the agent to generate SQL based on documentation it's provided, along with a dedicated tool to make SQL queries, insertions and edits. Figure 25 describes the setup of the system, including: the delegator, a basic agent which is responsible for controlling the user registration flow and database access, the necessary tools, and the resource layer.

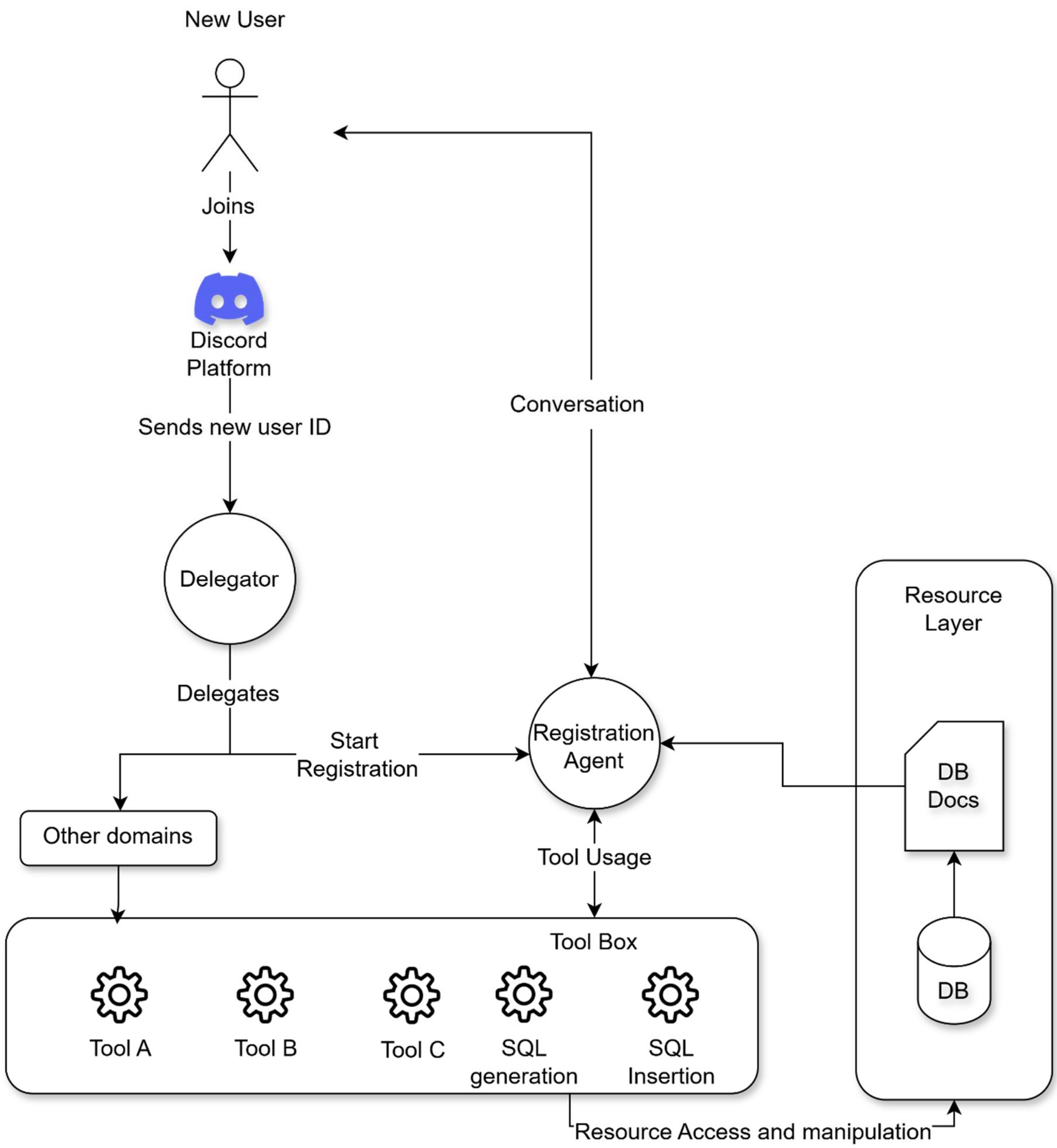

Figure 25 - Case study 3 system diagram

In Discord users can be invited to enter a server with dedicated URLs sent by anyone inside that server. Upon acceptance of an invitation the act of joining said server will trigger a special event on the platform which will be caught by the delegator agent that (due to the information contained inside itself about its surroundings) will understand that a new registration must be made, and will therefore gather the user's ID from the previous event and signal the Registration Agent to start the process of integrating the new user on the server.

Although server communication is entirely possible, to ensure privacy, the registration process is completely done through private messaging with the Registration Agent which after receiving a message from the delegator will contact the new user, as shown on Figure 26 (please note that some of the fields are specific to GECAD which was the inspiration for the development of this use case.)

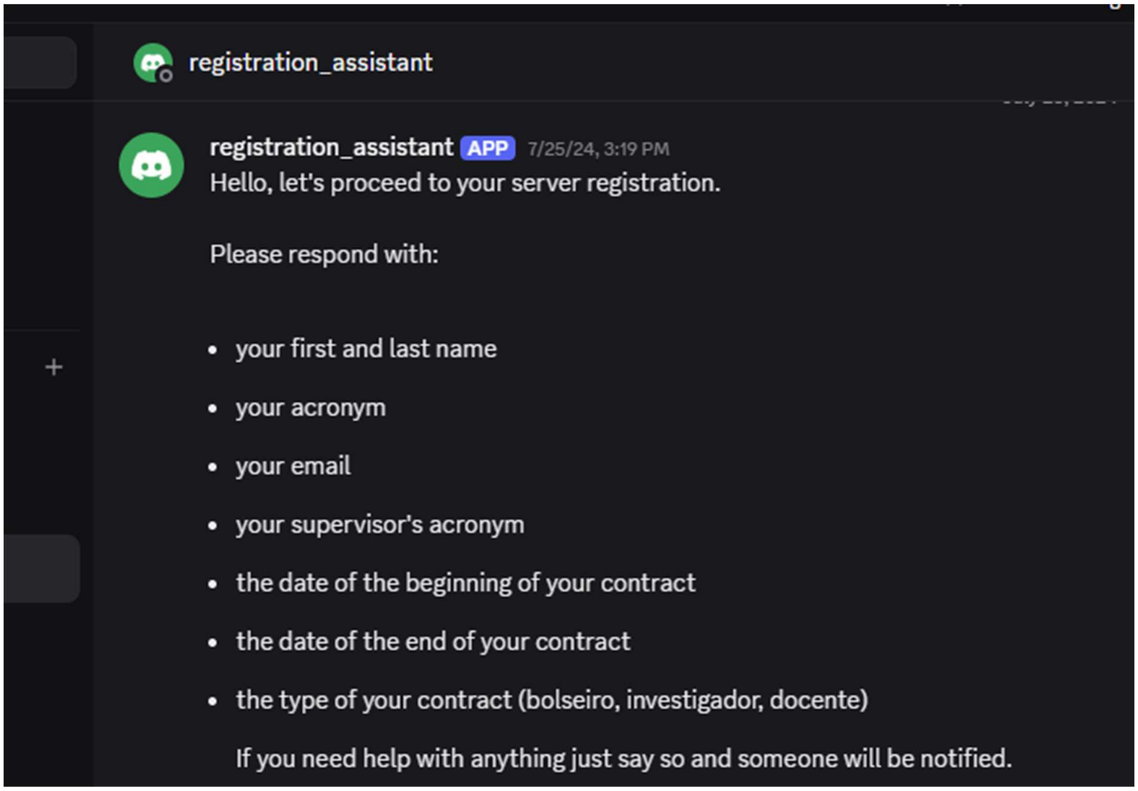

Figure 26 – New user registration assistant prompting

Once the user has successfully completed the registration, SQL is generated through the tools at the agent's disposal and promptly inserted on the database. Furthermore, the agent is capable of granting specific permissions to the user, allowing him to access different resources on the Discord platform.

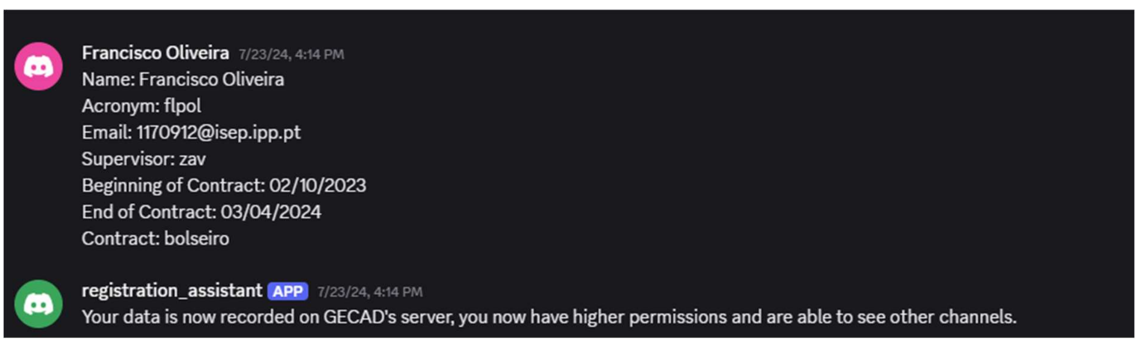

Figure 27 - Registration process interaction a)

Due to LLM capabilities and DB documentation injected on its system prompt, the agent also has the ability to perform various logical confirmations before tool access making the system more robust, this is demonstrated on the following figures.

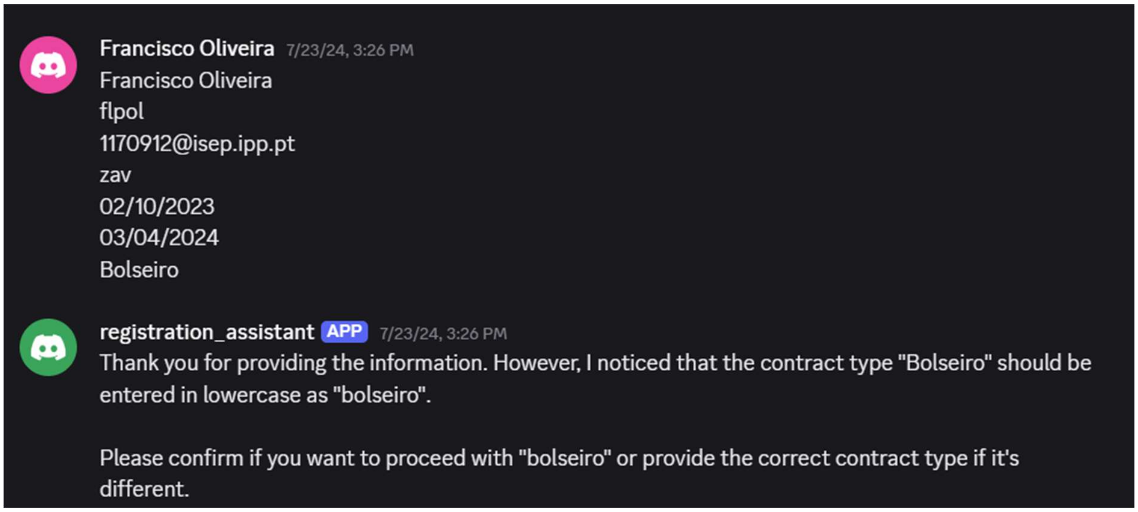

Figure 28 - Registration process interaction b)

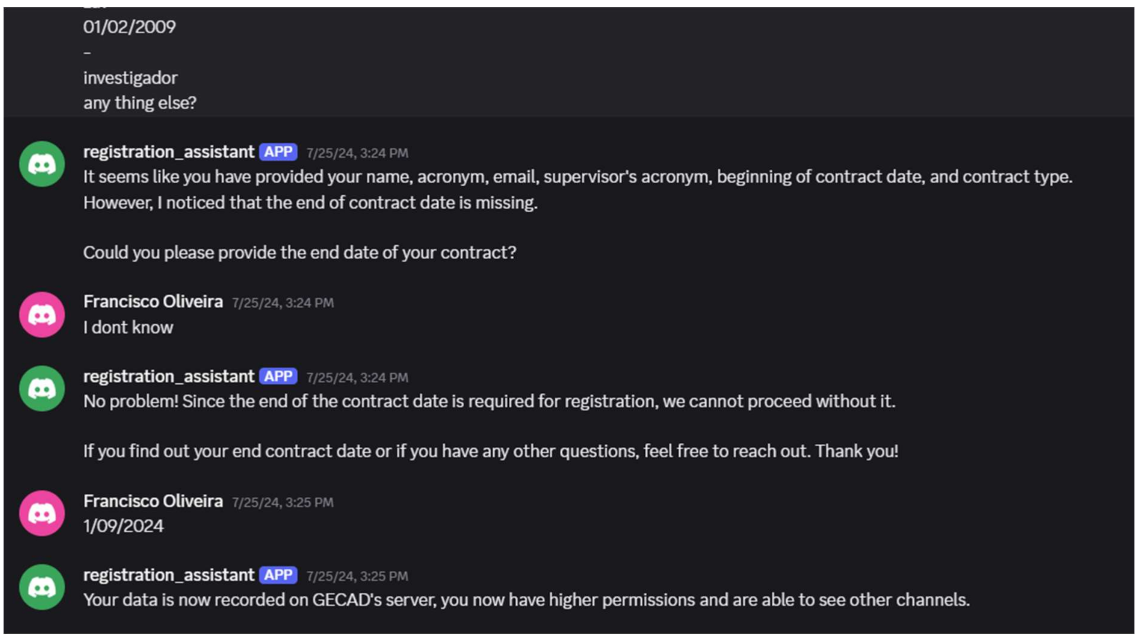

Figure 29 - Registration process interaction c)

The results of this foundational ARMS system work were crucial to understanding the limitations and liberties of the Discord platform. It was possible to conclude that the framework is appropriate for integration with the system proposed in this dissertation, as agents are able to communicate with users in a seamless manner while maintaining access to a set of available tools from the discord.py framework, such as message parsing and manipulation.

Furthermore, the development of this case study enabled the discovery of important LLM capabilities and how well they behave I n different aspects, such as correct context identification by the delegator agent; task assignment by the registration assistant; text generation for SQL insertion by the SQL generator tool; database interaction; and agent-user interaction.

This case study was also incredibly important for laying the foundations of practical tool and agent creation, resource layer management, and delegator experimentation. It yielded very positive results, as tasks were generally accomplishable by agents with relative ease, thus encouraging the exploration of more complex use cases.

5.4 Case study 4: Vacation complex delegation

The fourth case study explores how ARMS can be used to facilitate the process of creating and submitting vacation/leave requests. Compared to the registry use case, this is a deeply more complex problem that must take into consideration different aspects such as the current number of available leave days, superior approval, request modification, and national holidays.

This case study was envisioned with the purpose of greatly automating the vacation request process, by automatically routing requests between the chain of command without the need for manual human intervention while keeping everything as a conversation-like task between every involved actor. Table 12 enumerates the agents and tools used on the development of this case study.

Table 12 - Elements of case study 4

	Name (number of instances)	**Purpose**
Agents	Delegator Agent	Receive tasks from users and other agents and delegate them.
	Request Starter	Initiating vacation requests
	Request Handler	Handle vacation request process
Tools	Database Access Tool	Allows agents to access the database

Besides the delegator – who is common to every domain – this case studies' domain is composed of two main agents which work together to complete the request.

The first is the "Request Starter". This agent has, as its main responsibility, the initialization of a vacation request upon solicitation by the employee to the delegator. Different pieces of information are required by the agent, those being: the user's name, the number of available vacation days and who is responsible for accepting the eventual request. This data is provided with the aid of a tool which accesses the stored information based on the requester's ID.

Then through a conversation between the agent and the user requesting the vacation leave, the intended days are aligned with the possibilities established by the organization.

The Request Starter agent makes verifications, namely: if the number of total days does not exceed those available, if the vacation days were not requested on national holidays, and if the vacation does not land on periods where the organization does not allow for vacations to be requested. After everything is accepted by the LLM-powered agent, it iterates the request artifact that (with the aid of a tool) is then stored on the database. Upon the creation of the request artefact, the second agent, "Vacation Handler" proceeds to notify the required supervisors/managers for the fact and that it requires his/her approval.

Vacation Handler now takes centre stage, since it will articulate the rest of the process, having the ability and responsibility to interact and communicate with the required supervisors to alter, approve or deny the requests. Feedback on the supervisors' decisions is routed back to the requestee, which can then submit a new request with the required alterations or modify the current request. After everything is accepted the request is stored on the organization's database with the help of a tool and a message is sent to the original actor to inform of the fact. The diagram presented on Figure 30 explains how a vacation request would be processed.

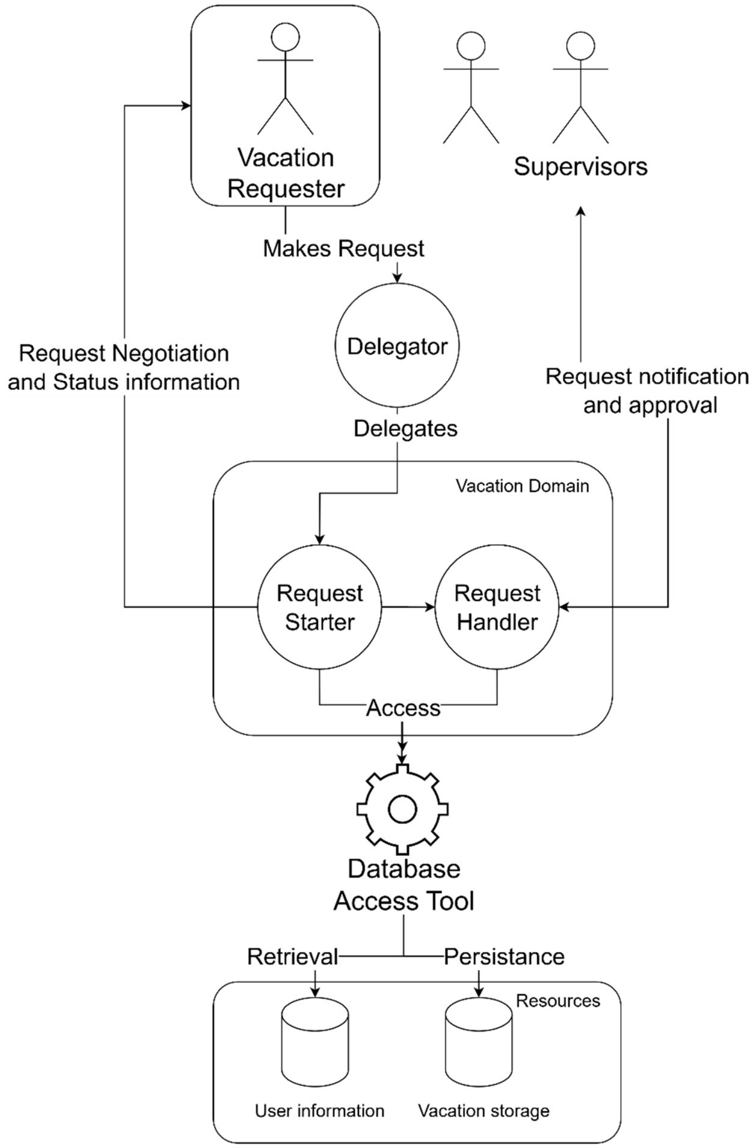

Figure 30 - Vacation request flow

The results from this rather complex case-study exhibit the capability of the models to stay in context while adhering to the provided prompt. Model GPT-4o-mini and GPT-4 demonstrated good enough logical understanding of the environment to process the requests, especially during the supervisor's phase where multiple vacation requests from different users would be received. However, when the context becomes too large after several requests, a mechanism to remove processed requests must be put in place. GPT-3.5-turbo exhibited more confusion handling more numerous parallel requests, mixing the days that were requested by different employees.

In this paper, a MAS powered by LLMs was explored and developed to better understand the limitations and possibilities of employing these new models to real world scenarios. Special attention was given to the size of the system environment regarding each case study to preserve resources and avoid any unnecessary extra complexity.

It was noted that the LLMs have a great capability to understand and delegate tasks within the realm of the hierarchy of each domain. The system provides significant value by automating and developing otherwise HR manual and repetitive tasks while providing employees with a conversational and convenient interface to both register themselves on a given platform and completing vacation requests.

Even though only top performant models were used, the LLMs proved to be resilient when interferences were deliberately experimented keeping a close focus on the provided prompts and given roles. This way, it is possible to observe how a MAS system can be created and used for complex tasks with multiple steps to successfully reach an end goal, which may prove to be a step forward in the application of these models to other real-world scenarios which involve delegation of tasks to multiple agents replaceable by LLMs.

5.5 Case study 5: Smart building control

On this final case study, the ability to integrate the system on a broad environment controlling multiple aspects is explored. ARMS was integrated with different tools to manage, assist and control the proper functioning of a smart building containing multiple appliances, such as lights, door locks, a simulated integrations with a flyball regulator system (as described on previous sections) and air conditioners. Table 13 enumerates the agents and tools created.

Table 13 - Elements of case study 5

	Name (number of instances)	**Purpose**
Agents	Infrastructure Delegator	Receive tasks from users and other agents and delegate them.
	IoT agents (3)	Receive delegator requests and inform rooms of requested changes
	Rooms agents (9)	Control IoTs of each room of the smart building
	Flyball Agent	Coordinate the impact of the flyball regulator system on the environment
	Q&A Delegator	Receive tasks from users and other agents and delegate them.
	Knowledge Agents (3)	Use RAG tools to provide informed responses to users
Tools	RAG tool	Support knowledge agents using RAG
	Flyball Regulator Tool (4)	Calculate flyball equations given external data
	API Request Maker	Generate and execute API requests, this tool.

The goal of this experiment was to understand how the MAS would behave on a large environment where both user and non-user interactions occur and demand that the agents accommodate outside necessities by communicating amongst themselves and using tools to achieve desired outcomes. Figure 31 demonstrates how ARMS was configured to control a specific building using the Discord platform mentioned on previous use cases.

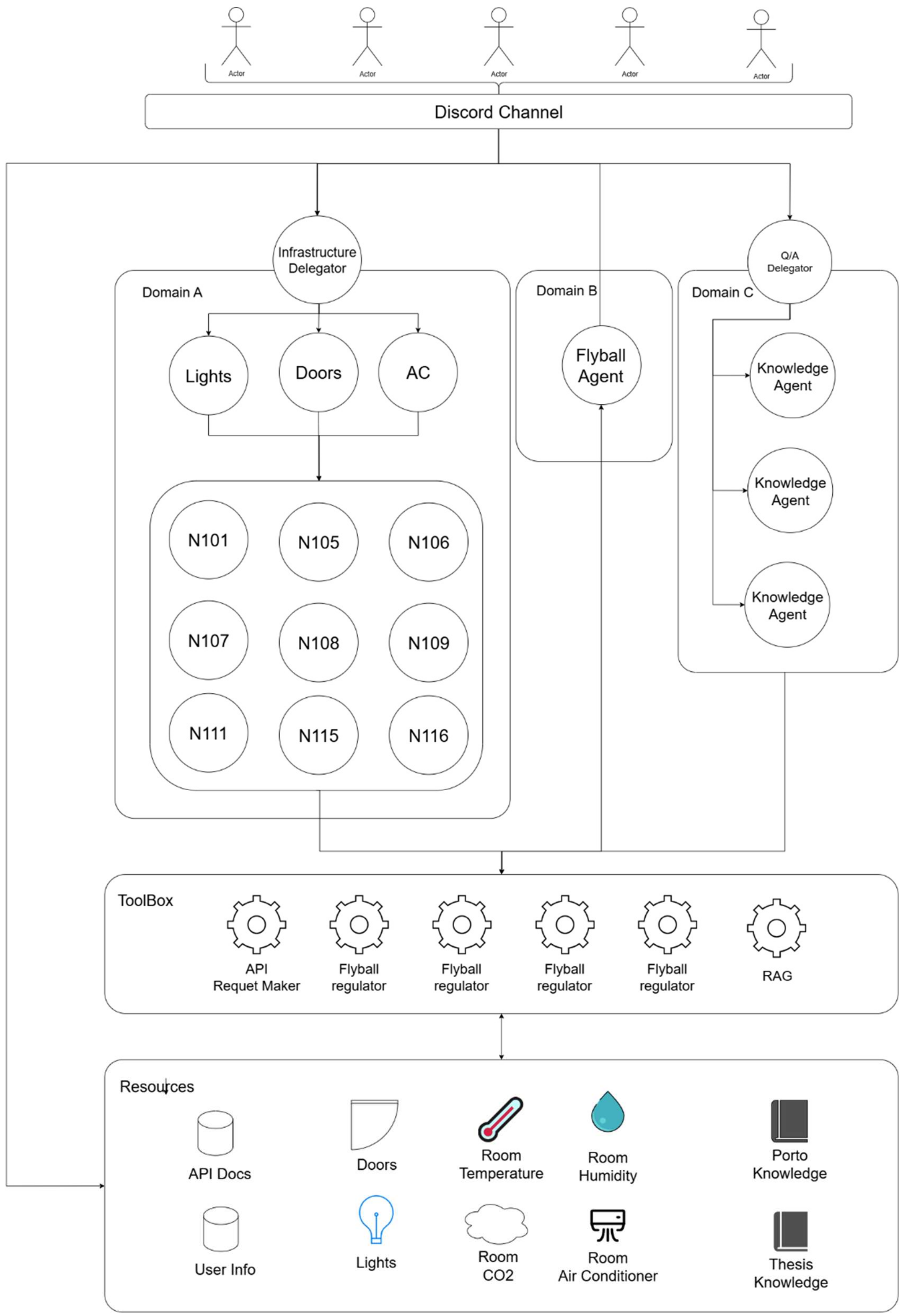

Figure 31 - Case study 5 system diagram

As it is possible to observe this is a rather more complex use case where multiple domains coexist with users, tools resources and delegators. This setup was also implemented under a Discord server, where humans and AI can connect with each other in seamless way.

Within this use case and environment there are (besides inter-user communication), fundamentally, three different types of events happening at the same time, each within its own domain, those being: room IoT manipulation (Domain A), AC control based on flyball regulator control (Domain B), and domain information retrieval (Domain C). To better understand each one these events, this section illustrates each one separately.

5.5.1 Room IoT manipulation

On this segment of the system, users interact with ARMS to control doors and lights of seven different rooms within a smart building, there are a total of 11 agents at play served by 2 resource access tools. Figure 32 illustrates the basic flow of a user's request to lock doors and turn off lights:

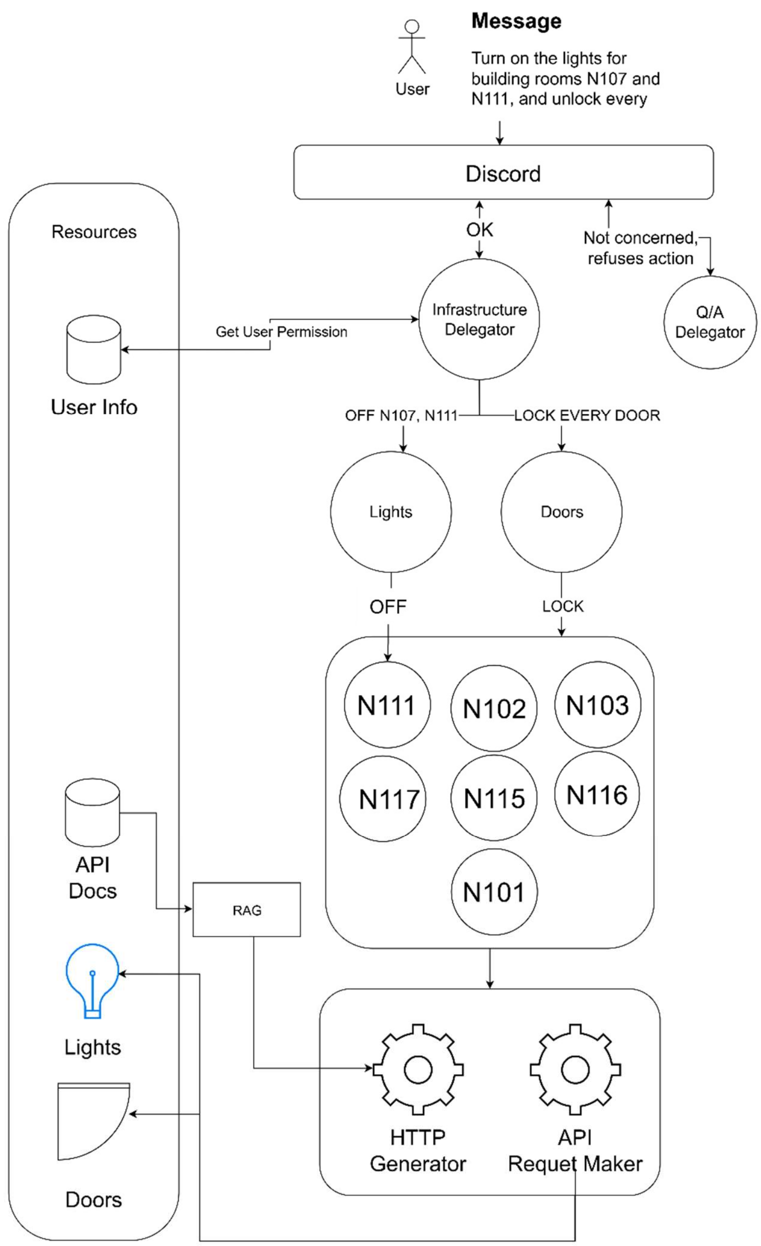

Figure 32 - Basic flow of a user request

Each of the IoTs were simulated and are accessed by an API visualized by the dashboard represented on Figure 33, created specifically to provide a more user-friendly overview of the building's IoT states.

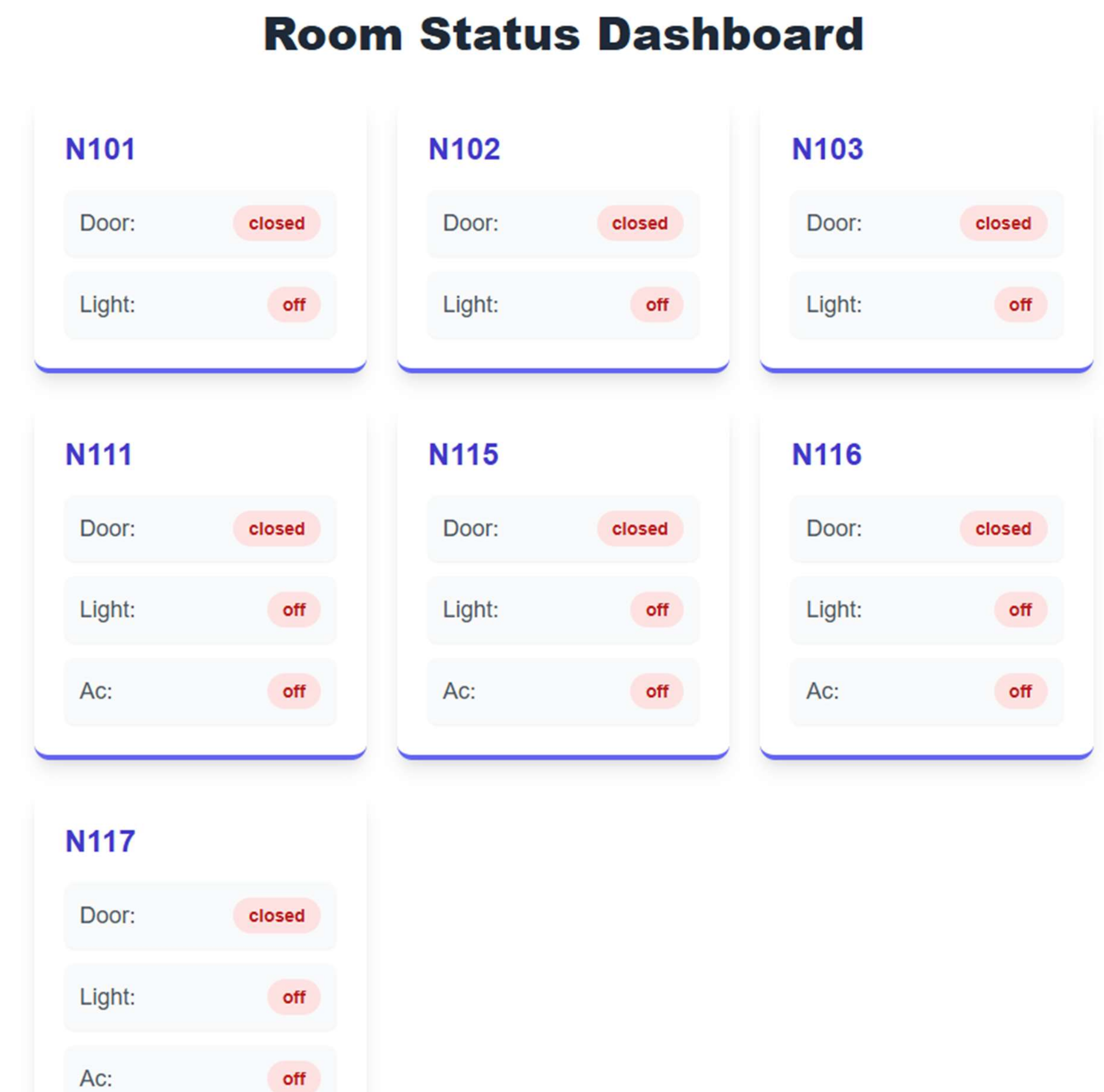

Figure 33 - Room control dashboard

To start off the flow of the scenario, the user first sends a message to a channel where the Infrastructure Delegator agent is present, as shown in Figure 34.

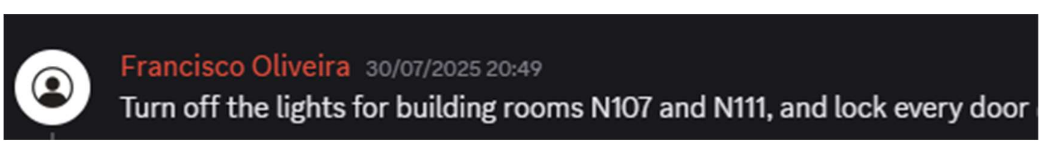

Figure 34 - Initial room control request

The Infrastructure Delegator then picks up on this message, interprets it and creates a new conversation for that particular task (in the Discord platform this is handled as a "thread").

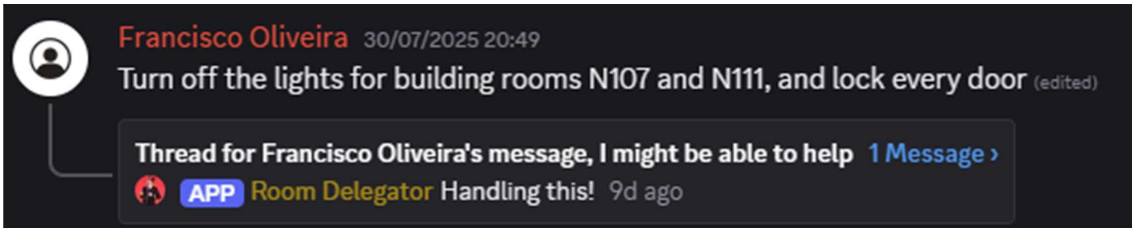

Figure 35 - Creation of threads to answer requests

After understanding the users request, the Infrastructure Delegator then forwards this request to a dedicated agent communication channel in order to facilitate specific use case communication and avoid cluttering of the original channel. Channels were specifically created for building management, as shown on Figure 36.

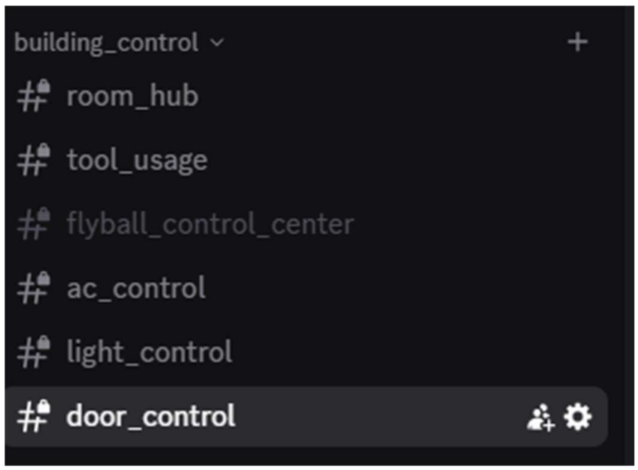

Figure 36 - Building management on the Discord platform

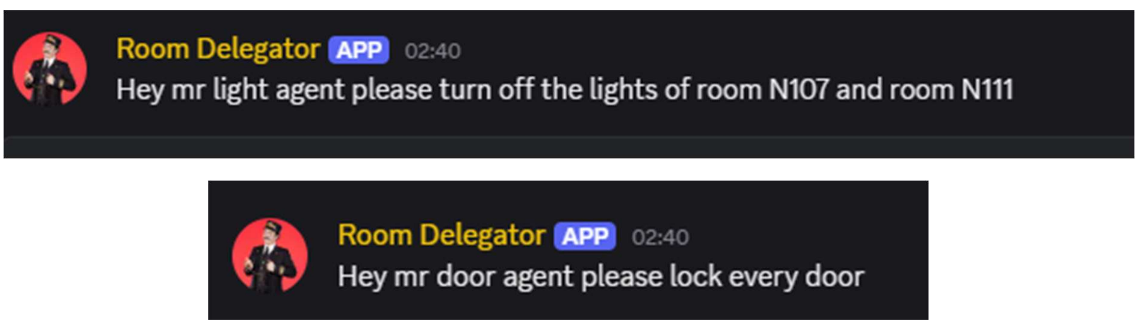

Figure 37 - Delegator request delegation on ligth_control channel and door_control channel

Agents appear on the platform on a specific sidebar where it is also possible to see online and active users segregated by roles and responsibilities, as depicted on Figure 37.

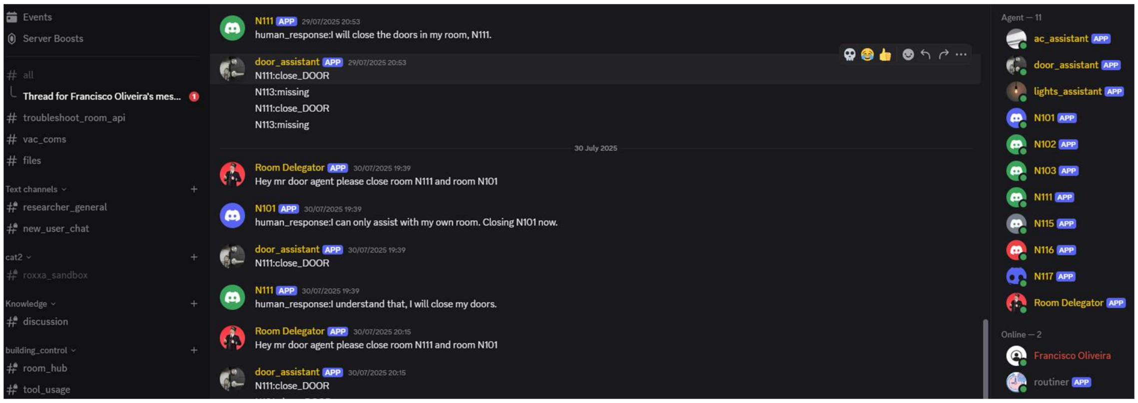

Figure 38 - Online agents ready to take requests (see sidebar)

As shown in Figure 39, agents follow the Infrastructure Delegator's orders, the doors and lights assistants pick up on the requests by calling each room individually and act accordingly.

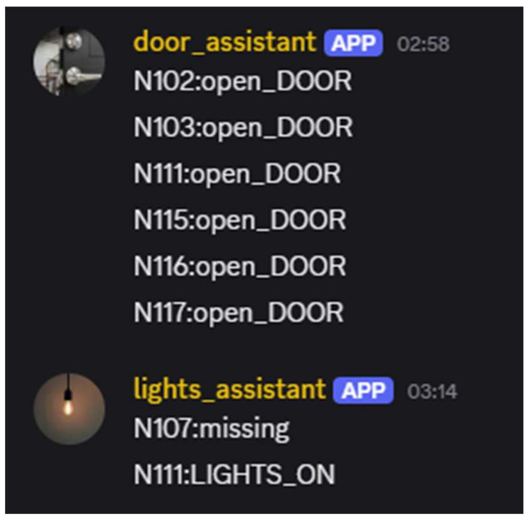

Figure 39 - IoT agents calling room agents to act

It is possible to observe in Figure 40 that the lights assistant agent actually understands that we've requested the lights to be turned on for a room that does not exist on the system, it knows this by having a grasp of the agents available within its domain.

Rooms then call the API tool that will interact with the real world (Figure 38), turning, as shown in Figure 40, the rooms to the according states.

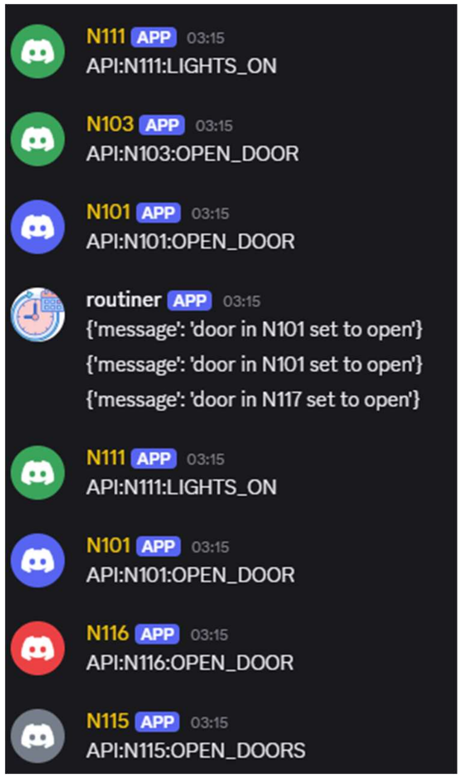

Figure 40 - Room agents calling API tool (routiner)

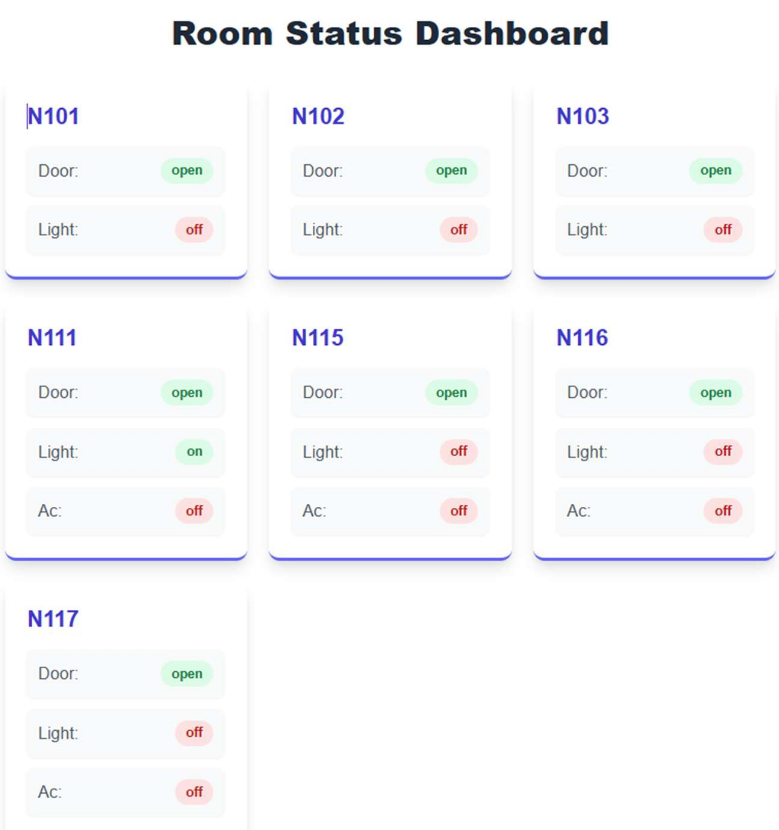

Figure 41 - Dashboard depicting agent impact

5.5.2 Self-regulating system

For the second scenario, integration with the flyball regulator system as an external tool is tested. This system was specially designed to monitor data and act on risk levels. Therefore, to recreate these scenarios, a special simulation was created for the different rooms that would emulate oscillating levels of temperature, CO2 and humidity.

Risk levels (categorized in 4 different stages) were then established for each of these three simulated sensors and given corresponding accelerations for integration with the flyball regulator system. New values were generated every 0.5 seconds and CO2 risk intervals were established according to (Lowther et al., 2021).

Dashboards were created to better understand the course of the simulation and manually alter the conditions of the environment of each room as seen in Figure 42, where window (a) shows this control of the different parameters, (b) provides detailed information on the state of the flyball system for each room, graphs on the (c) section provide visual representations of the evolution of the flyball regulator system and (d) of the levels of each room (for this scenario, only rooms N111, N115, N116 and N117 were considered).

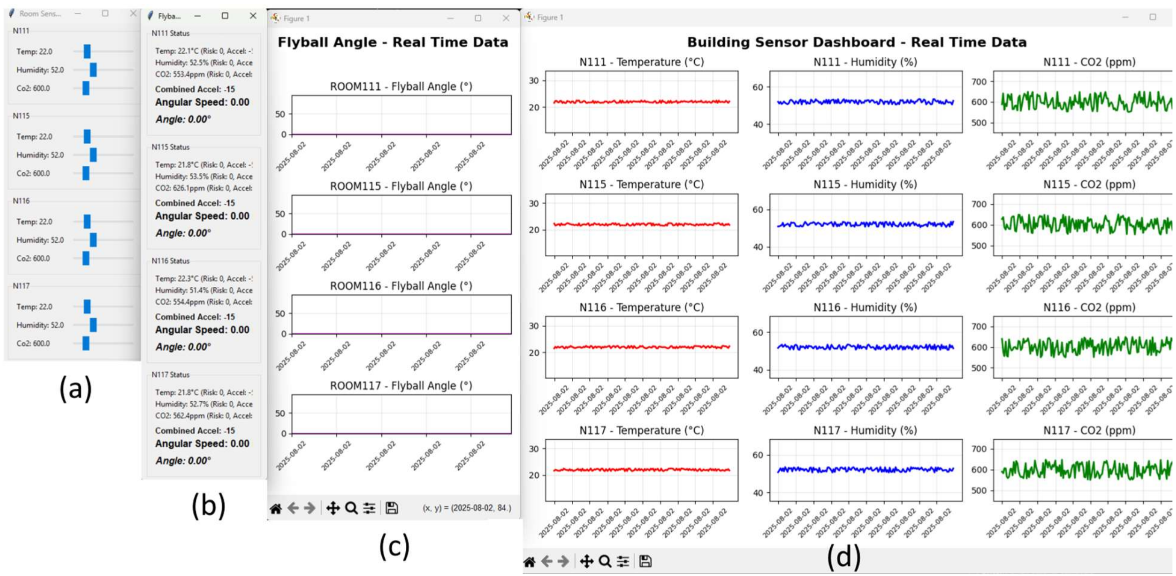

Figure 42 - Different controls of the simulation and graphical representation of states

The following diagram describes how this scenario was assembled to take full advantage of the flyball regulator capabilities of providing crucial information to agents controlling each room, allowing them to make informed decisions on when to turn on or off-air conditioners by communicating among themselves when one of the above-mentioned levels reaches critical states. Each step is numbered to better understand the flow of the information, and to help referencing each one individually.

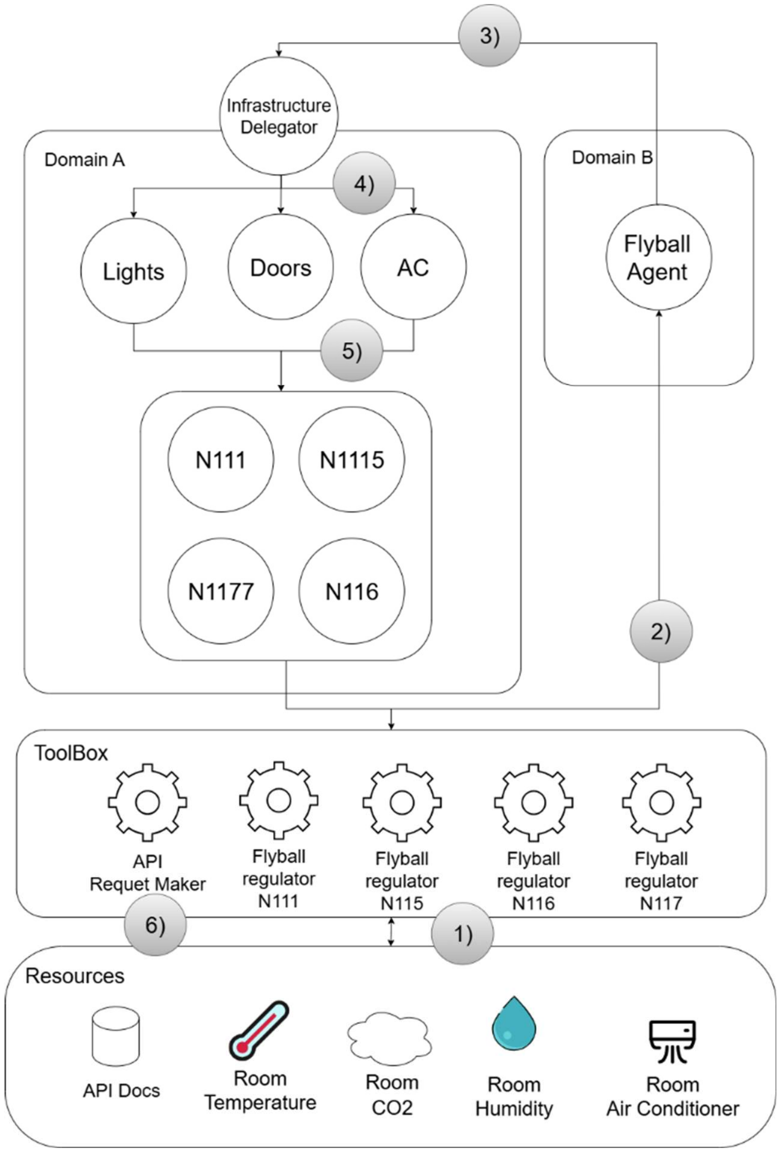

Figure 43 - Flow of flyball regulator system auto-regulation

When the system suffers alterations that imply a significant risk, as it is possible to observe in Figure 43 and assigned as step 1), the angle of the flyball regulator begins changing and increasing. At first, small angles representing small speed increases are not considered a great risk and are, therefore ignored by the system, but as the risks start to accumulate over time, the speed of the simulated mechanical system also increases making the angle reach critical levels.

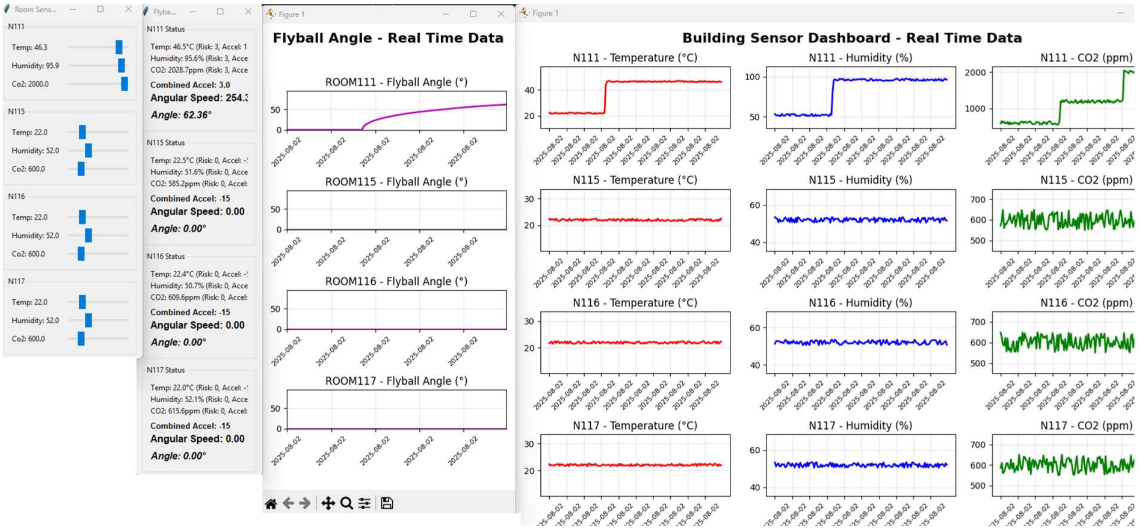

Figure 44 - Dashboard depicting induced changes

Flyball information is fed into a proper messaging/ logging channel, this represents step 2) and is depicted in Figure 45.

Figure 45 - Multiple flyball agents communicating at the same time

Initial angles are (for LLM effort saving reasons) are automatically discarded, but as they pass certain thresholds, the LLM starts monitoring each individual one, and when it notices that an angle is considered critical, the agent decides to inform the proper agents (in this case by making a request on a public channel, shown in Figure 46 and represented as step 3) that action must be taken on the room where it sees that levels are not optimal.

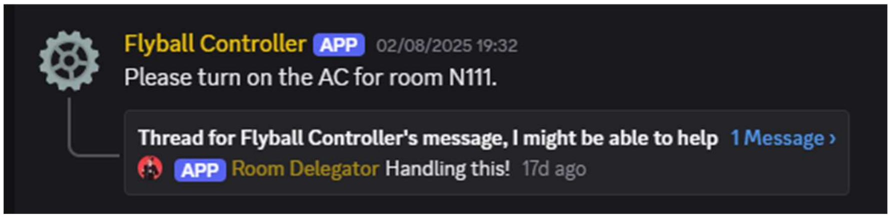

Figure 46 - Flyball Controller agents reacting to dangerous levels

The Infrastructure Delegator then receives these requests and (just as it would with any other user request) delivers them to the agent responsible for handling room atmosphere control, in this case, the air conditioner agent (step 4, Figure 47), which will in turn assign tasks to the proper rooms (step 5. Figure 48) that will regulate themselves using an API Request Maker Tool to directly interact with the air conditioner responsible for climatization (step 6).

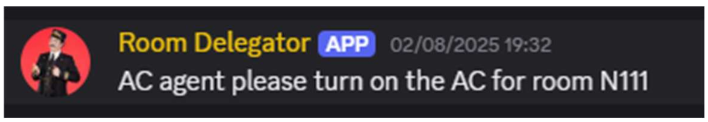

Figure 47 - Delegator delegating Flyball Controller agent request

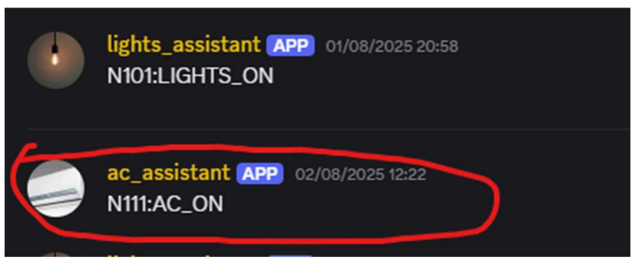

Figure 48 - Agents completing the Flyball Controller agent

5.5.3 Question answering system

Also integrated amid the two domains mentioned above is a question and answering system related to GECAD, it is comprised of three different agents that have distinct knowledge areas and are served by a dedicated Q/A Delegator agent that works alongside the Infrastructure Delegator. Figure 49 demonstrates this simple integration which functions based on the principles published in (Oliveira et al., 2025).

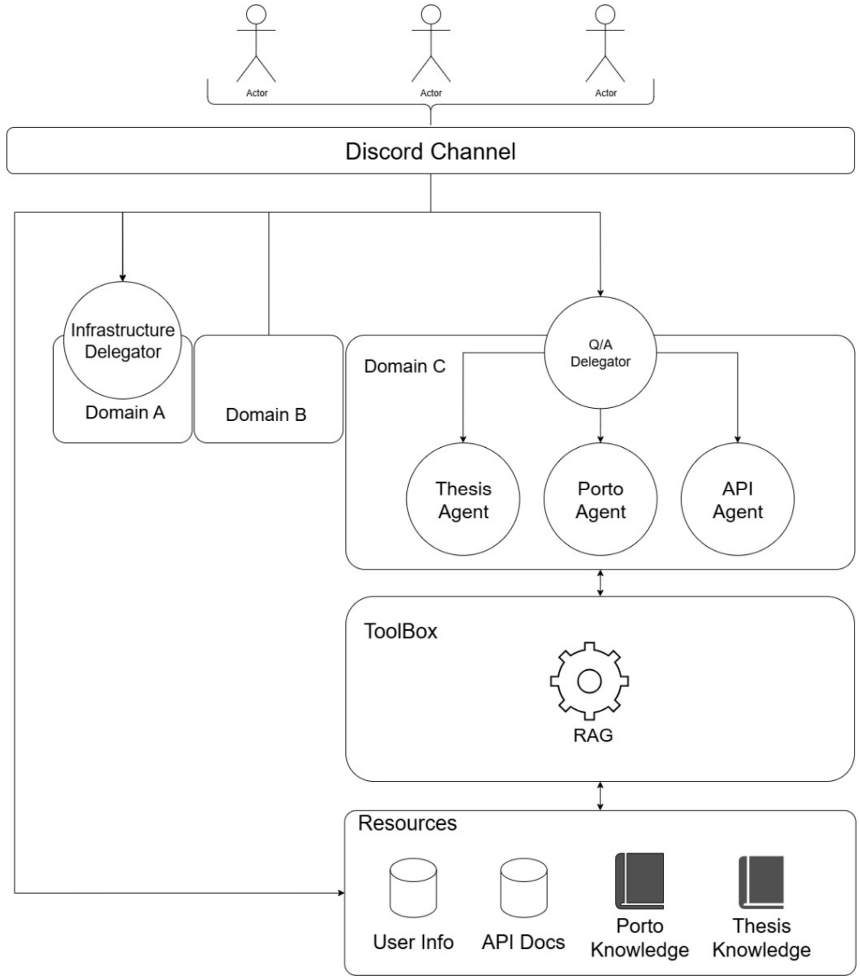

Figure 49 - Case study 5 Domain C system diagram

As it is possible to observe, excluding “User Info” crucial for authorization purposes, there are three different knowledge domains:

- API Docs: This domain has complete knowledge on the APIs surrounding the whole system, whether it’s accessing the air conditioners, locking doors or getting temperature information. It aims to help new users on using the system.
- Porto Knowledge: This is a generic knowledge domain about the city of Porto, where GECAD is located, it consists of multiple pages of web articles that provide useful insights about the city. It aims to help GECAD visitors on finding out more about the city surrounding them.
- Thesis Knowledge: This domain consists of documents related to publications associated with GECAD, namely Luis Gomes’s PhD Thesis, Rafael Barbarroxa Master’s thesis, and multiple other papers.

The following Figures depict examples of how a new thread is created and the Q/A Delegator starts the retrieval process by calling the agent it sees fit to solve the question performed.

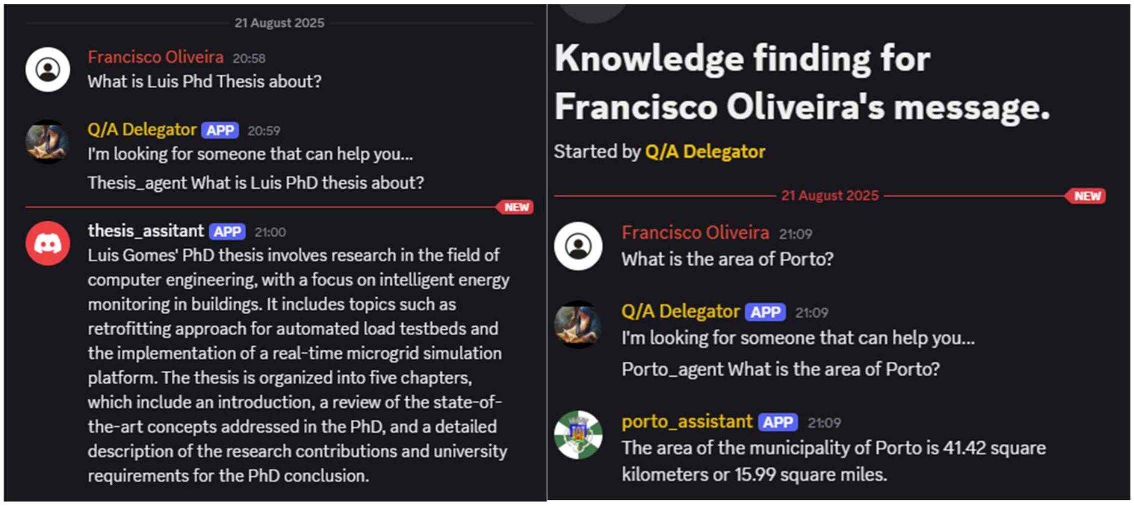

Figure 50 - Q/A Delegation and assistant responding´

To test the proposed system implementation, 120 prompts were made to the system to test the capability of delegation upon user prompting and question answering, and 100 tests that would induce stressful room conditions to test the self-regulating system were also created. These results were categorized on Table 12 in both success and Failure. Success cases are split into total success and deviations during delegation or agent interpretation that although are not optimal did not prevent tasks from being completed, this includes agents taking requests that were not meant to them, the delegator sending requests to non-intend domains, and extra tool usage. Under the Failure category, are included every scenario that proves to be fatal for the proper execution of the desired outcomes, including bad delegation, wrong tool usage, tool malfunction, and (for the case of the QA system), bad RAG responses.

Table 14 - Case study 5 test results

		Room IoT Manipulation	**Self-Regulating System**	**Question Answering System**	
Success	Complete success under predicted path	84	79	13	
	Some deviations, but end goal achieved	2	1	-	
Failure	Wrong/bad delegation	7	11	-	
	Wrong tool usage	5	3	-	
	Tool malfunction	2	6	-	
	Bad RAG response (only applicable to RAG prompting	-	-	7	
Total		100	100	20	220

As it is possible to observe the system has a success rate of task completion of around 80%, and failures tend to occur when models must decide, from a range of options, a choice to make, since more than 50% of failures happen at the delegation or tool choosing process. It was also observed that when testing the RAG system on larger documents such as the PhD thesis, the system tends to underperform, perhaps due to RAG quality and high degree of depth that these documents possess over the span of multiple text chunks, making it harder for the algorithm to pick the correct ones to include on the response. RAG can be optimized by increasing the number of chunks considered and the size of said chunks.

This final case study is crucial to the understanding of how the system can be implemented on a large scale where multiple domains come together to create a multi-disciplinary place. The Discord platform was pushed to great lengths to accommodate these numerous agents running in parallel and processing simultaneous requests. Furthermore, it was possible to demonstrate the possibilities of LLMs as intelligent agents that are capable of interpreting their environment and using external tools to interact with it without being specifically requested to.

Some limitations were found when running the entire system on a single machine, as each agent represents a thread that has to make several operations which oftentimes require a lot of overhead and are running in parallel with the room simulation and live graph drawing of the flyball angles. To overcome these issues, separate machines are possible to be employed in different parts of the domain or even for different agents or tools, since communication happens on the Discord “public” space, therefore allowing for the implementation of a machine-agnostic distributed and robust system.

6. Conclusions

The advancements of LLM powered systems are currently at a relatively early stage of development. New systems, startups and use-cases are appearing frequently around the world. Concerns about the potential of these systems are supported by doubts on limitations and lack of proven value regarding these models which sometimes are enhanced by media hype and promises of revolutionary transformation on all fields of industry. This dissertation delved into the exploration of the state-of-the-art of areas concerning and surrounding the developments being made in industrial digitalization, multi-agent systems and the integration of the transformer architecture in different degrees of those fields. Furthermore, practical exploration was done through the implementation of a multi-agent system named ARMS that tested the capabilities of LLM models to interconnect, organize and produce real value on a rather complex environment where humans come into play to complete tasks and achieve goals. Complex agents powered by LLMs were implemented to employ these models' capabilities on decision-making tasks, generation of content and delegation of responsibilities in real-time, concluding that it is, in fact, possible to orchestrate a system where natural-language is employed in the midst of human actors to achieve easy to trace workflows that accomplished real tasks as proven by the five case studies described above that originated multiple scientific papers. Limitations were found when springing up multiple instances under the same machine workload and when system prompts of LLMs were not correctly defined, which, in most cases and using the top transformer models, was a human error. In ARMS the concept of Tool was deeply investigated and experimented with, these instruments allowed for models that output text to output more meaningful and palpable change in their environment, creating a blueprint for researchers and developers to guide and inspire themselves.

6.1 Research questions and objectives

In the first chapter of this dissertation, five different research questions were enunciated where eight research objectives were derived from. During the research, development, experimentation and testing of this thesis, those questions were answered guided by the research objectives. To conclude this work, they are hereby given a final and specific discussion.

RQ1: What are the current traditional MAS development paradigms? What's the progress of LLM fusing on said paradigms?

Through the completion of RO1 (Investigate current MAS development paradigms through analysis of the state-of-the-art) and RO2 (Investigate current progress on the usage of LLMs in a MAS-like setting), the state-of-the-art surrounding MAS development was investigated. Different MAS architectures were explored and are enunciated in the appropriate section of the state-of-the-art, those paradigms include Hierarchy, Holarchy, Coalition, Team, Congregation, Society, Federation Market and Matrix Organization. Furthermore, communication between agents was explored which had significant contributions to the proposed solution as well as the exploration of the definition of agent contributing to contextualization prior to the agent design of ARMS. Also included in the state-of-the-art, different frameworks introducing a fusion between LLMs and MASs were described, most of which are very recent and where gaps were identified when sounding for customizability, tool integration and system distribution.

RQ2: Can a system that integrates LLMs as rational engines be considered a traditional MAS?

This research question is discussed in greater detail at the end of the state-of-the-art (corresponding to RO3 (Discuss and conclude if using LLMs together with traditional MAS techniques can be considered traditional MAS or a new nomenclature should be considered)). In short: yes, when considering the traditional definition of agent in (JENNINGS & WOOLDRIDGE, 1995), the only aspect that differs from traditional implementations and implementations using LLMs is the engine used to make decisions, which, even amongst traditional MASs is not always homogeneous.

RQ3: What are the limitations of using LLMs as rational engines for agents on a MAS?

When working with LLMs it becomes especially important to design solid and robust system prompts that can predict different abstract scenarios that guide LLM decision-making. These prompts should be able to tell LLMs how to respond when confronted with doubts or unexpected inputs, as well as when not to respond or even how not to respond (e.g. with answer codes that are later parsed), this was specially important during the conceptualization of the system (Chapter 4) and something to be very aware of when implementing it on every use case. When working with LLMs RO4 (Identify gaps on LLMs working as MAS), was accomplished and libraries that implemented multi-agent systems together with LLMs for communication platforms that were not entirely crated with the purpose of using LLMs were not encountered.

Furthermore, it was observed that when integrating multiple LLMs in a common space where messages are shared among many agents, it becomes very important to properly segregate messages and avoid conversation loops locks since, besides rendering agents as indefinitely busy, LLM usage costs are also a factor that must be included in the equation of developing using these models, thus helping to achieve RO5 (Identify limitations on LLMs working as MAS).

RQ4: How can LLMs act on the world around them?

As observe in each use-case, there are many possibilities to make LLMs act in the world around them, that is if proper tools are implemented in a way that makes them usable by these models without great confusion. This involves creating descriptions of those tools that are not too short and not too long, that possess very well-defined inputs formats and output formats, such as the usage of API requests in case study 5, user access in case study 4 and documentation consultation in case study 2 and 5.

By parsing decisions provided as strings from LLM-powered agents, tools that had great impact in the environment could be easily accessed and used as extensions of those decisions. The results of these environmental changes were then fed back to the observations of the agents which could change their view of the external world.

RQ5: How can a MAS using LLMs as rational engines in its agents be orchestrated? What are the limitations of that orchestration?

When developing ARMS and accomplishing RO6 (Conceptualize, create and develop a solution that attends to any existing gaps referred to in RO5), different considerations to accommodate LLM demands needed to be taken into account which resulted in the implementation of 1) a layered agent architecture providing essential components such as parsing and input refining, as well as behaviour definition; 2) segregation of concerns due to LLM task overload, achieved by the implementation of the delegator agent capable of separating request domains; 3) proper tool availability and definition, using the “tool box” approach making tools available to multiple agents at the same time, which were used on a “as-needed” basis; 4) a solid resource layer that would enable quality information to be fed to the LLM agents so that quality outcomes would succeed, this is famously known as the "garbage in, garbage out" concept; 5) and finally, a robust underlying communication platform is required that allows agents to route messages to the intended destinations while having access to message metadate such as who send the message, to what channel and when. The case studies implemented were crucial in the resolution of these concepts which helped RO7 (Gather knowledge on the limitations and potentials of the solution mentioned in RO6) to be achieved culminating finally with the completion of RO8 (Test the solution proposed in RO6.)

6.2 Future work

After completion of the research objectives and pondering on the pre-established research questions, and although all were achieved and answered, further refinements to the ARMS system are still possible.

Although the usage of Discord as a communication platform offers a great array of possibilities to send and receive messages to different users, and in different communication channels, it also comes with a great overhead that demands unnecessary computation of hosting machines. This signals a need for a custom-built communication platform to include only the strictly necessary, shifting the computation load to LLM processing and agent orchestration.

Most of the system was built using proprietary models which directly translates to security risks if information is not abstracted before being sent. A great effort to implement agents on local models means more control and safety in the long run, as prices shift and so do policies. Assuring local models means assuring internal knowledge and control, which, obviously, comes at greats computation costs, a situation that fortunately seem to only improve as time passes.

Furthermore, case study 5 is the most complex and complete demonstration of the ARMS system which was developed on a simulated artificial environment, testing the entire system on a real world scenario with multiple users would shed some light on what works best and what doesn't, allowing for a more refined and useful system to be developed based on user feedback and daily usage.

Additionally, the usage of tools, although rather varied has still room for expansion. Realistically, there are few constraints to what can be implemented as a tool for agents to utilize. Exploration of new tools is certainly one of the most exciting spaces to increase the functionalities provided by the system.

Finally, a graphical interface that would allow a manager or system administrator to track request workflow could increase efficiency, security and developing capabilities when developing new tools or implementing new domains, helping to track what's happening among the multiple agents. As it stands this is only possible through the following of agent messaging on the communication platform.

References

$1.5 Billion AI Unicorn Collapse, All Indian Programmers Impersonating! (n.d.). Binance Square. Retrieved 22 June 2025, from https://www.binance.com/en/square/post/24723372076545

Abdelrahman, O., & Keikhosrokiani, P. (2020). Assembly Line Anomaly Detection and Root Cause Analysis Using Machine Learning. *IEEE Access*, *8*, 189661–189672. https://doi.org/10.1109/ACCESS.2020.3029826

Acharya, D. B., Kuppan, K., & Divya, B. (2025). Agentic AI: Autonomous Intelligence for Complex Goals—A Comprehensive Survey. *IEEE Access*, *13*, 18912–18936. https://doi.org/10.1109/ACCESS.2025.3532853

Afify, O., Neidhart, M., Schubert, L., Tkalec, F., & Zimmermann, A. (2025). Large Language Model Application Frameworks for Domain-Specific Chatbots. In A. Zimmermann, R. Schmidt, L. C. Jain, & R. J. Howlett (Eds), *Human Centred Intelligent Systems* (pp. 51–61). Springer Nature. https://doi.org/10.1007/978-981-97-8598-8_6

Aggarwal, C. C., & Reddy, C. K. (2018). *Data Clustering: Algorithms and Applications*. CRC Press.

Agrawal, K. P., Garg, S., Sharma, S., & Patel, P. (2016). Development and validation of OPTICS based spatio-temporal clustering technique. *Information Sciences*, *369*, 388–401. https://doi.org/10.1016/j.ins.2016.06.048

Akkem, Y., Biswas, S. K., & Varanasi, A. (2023). Smart farming using artificial intelligence: A review. *Engineering Applications of Artificial Intelligence*, *120*, 105899.

Akundi, A., Euresti, D., Luna, S., Ankobiah, W., Lopes, A., & Edinbarough, I. (2022). State of Industry 5.0—Analysis and Identification of Current Research Trends. *Applied System Innovation*, *5*(1), Article 1. https://doi.org/10.3390/asi5010027

Albeladi, K., Zafar, B., & Mueen, A. (2023). Time Series Forecasting using LSTM and ARIMA. *International Journal of Advanced Computer Science and Applications*, *14*. https://doi.org/10.14569/IJACSA.2023.0140133

Amirian, H., Dalvand, K., & Ghiasvand, A. (2024). Seamless integration of Internet of Things, miniaturization, and environmental chemical surveillance. *Environmental Monitoring and Assessment*, *196*(6), 582. https://doi.org/10.1007/s10661-024-12698-9

Arkin, R. C., & Mackenzie, D. C. (1994). *Planning to behave: A hybrid deliberative/reactive robot control architecture for mobile manipulation*. 5–12.

Arm, J., Zezulka, F., Bradac, Z., Marcon, P., Kaczmarczyk, V., Benesl, T., & Schroeder, T. (2018). Implementing Industry 4.0 in Discrete Manufacturing: Options and Drawbacks. *IFAC-PapersOnLine*, *51*(6), 473–478. https://doi.org/10.1016/j.ifacol.2018.07.106

Assis, M. V. O., Carvalho, L. F., Lloret, J., & Proença, M. L. (2021). A GRU deep learning system against attacks in software defined networks. *Journal of Network and Computer Applications*, *177*, 102942. https://doi.org/10.1016/j.jnca.2020.102942

Asthana, S., Hilleli, S., He, P., & Halfaker, A. (2025). Summaries, Highlights, and Action Items: Design, Implementation and Evaluation of an LLM-powered Meeting Recap System. *Proc. ACM Hum.-Comput. Interact.*, *9*(2), CSCW176:1-CSCW176:29. https://doi.org/10.1145/3711074

Babaev, D., Savchenko, M., Tuzhilin, A., & Umerenkov, D. (2019). E.T.-RNN: Applying Deep Learning to Credit Loan Applications. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2183–2190. https://doi.org/10.1145/3292500.3330693

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *ArXiv*, *1409*.

Bakshi, B. R., & Paulson, J. A. (2022). Sustainability and Industry 4.0: Obstacles and Opportunities. *2022 American Control Conference (ACC)*, 2449–2460. https://doi.org/10.23919/ACC53348.2022.9867815

Balushi, A.-A., Al-Bemani, A., Al Araimi, S., Balaji, G., Suresh, U., & Najeeb, A. (2025). AI-Driven Multi-Modal Information Synthesis: Integrating PDF Querying, Speech Summarization, and Cross-Language Text Summarization. *Procedia Computer Science*, *258*, 2996–3018.

Bányai, T., Tamás, P., Illés, B., Stankevičiūtė, Ž., & Bányai, Á. (2019). Optimization of Municipal Waste Collection Routing: Impact of Industry 4.0 Technologies on Environmental Awareness and Sustainability. *International Journal of Environmental Research and Public Health*, *16*(4), Article 4. https://doi.org/10.3390/ijerph16040634

Barbarroxa, R., Gomes, L., & Vale, Z. (2025). Benchmarking Large Language Models for Multi-agent Systems: A Comparative Analysis of AutoGen, CrewAI, and TaskWeaver. In P. Mathieu & F. De la Prieta (Eds), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection* (pp. 39–48). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-70415-4_4

Batty, M. (2018). Digital twins. *Environment and Planning B: Urban Analytics and City Science*, *45*(5), 817–820. https://doi.org/10.1177/2399808318796416

Baum, L. E., & Eagon, J. A. (1967). *An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology*.

Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, *37*(6), 1554–1563.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, *41*(1), 164–171.

Baum, L. E., & Sell, G. (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics*, *27*(2), 211–227.

Ben-Hur, A., Ong, C. S., Sonnenburg, S., Schölkopf, B., & Rätsch, G. (2008). Support Vector Machines and Kernels for Computational Biology. *PLOS Computational Biology*, *4*(10), e1000173. https://doi.org/10.1371/journal.pcbi.1000173

Ben-Israel, A., & Iyigun, C. (2008). Probabilistic D-Clustering. *Journal of Classification*, *25*(1), 5–26. https://doi.org/10.1007/s00357-008-9002-z

Berkhin, P. (2006). A Survey of Clustering Data Mining Techniques. In J. Kogan, C. Nicholas, & M. Teboulle (Eds), *Grouping Multidimensional Data: Recent Advances in Clustering* (pp. 25–71). Springer. https://doi.org/10.1007/3-540-28349-8_2

Berna-Koes, M., Nourbakhsh, I., & Sycara, K. (2004). Communication efficiency in multi-agent systems. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, *3*, 2129-2134 Vol.3. https://doi.org/10.1109/ROBOT.2004.1307377

Bilberg, A., & Hadar, R. (2012). *Adaptable and Reconfigurable LEAN Automation - a competitive solution in the western industry: FAIM*.

Bilgen, H. (2021). A global comparison methodology to determine critical requirements for achieving industry 4.0. *Technological Forecasting and Social Change*, *172*, 121036. https://doi.org/10.1016/j.techfore.2021.121036

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051

Borman, R. I., Napianto, R., Nugroho, N., Pasha, D., Rahmanto, Y., & Pratama Yudoutomo, Y. E. (2021). Implementation of PCA and KNN Algorithms in the Classification of Indonesian Medicinal Plants. *2021 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*, 46–50. https://doi.org/10.1109/ICOMITEE53461.2021.9650176

Braşoveanu, A. M. P., & Andonie, R. (2020). Visualizing Transformers for NLP: A Brief Survey. *2020 24th International Conference Information Visualisation (IV)*, 270–279. https://doi.org/10.1109/IV51561.2020.00051

Bressert, E. (2012). *SciPy and NumPy: An Overview for Developers*. O'Reilly Media, Inc.

Brooks, R. (2003). A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, *2*(1), 14–23.

Buestán-Andrade, P.-A., Santos, M., Sierra-García, J.-E., & Pazmiño-Piedra, J.-P. (2023). Comparison of LSTM, GRU and Transformer Neural Network Architecture for Prediction of Wind Turbine Variables. In P. García Bringas, H. Pérez García, F. J. Martínez de Pisón, F. Martínez Álvarez, A. Troncoso Lora, Á. Herrero, J. L. Calvo Rolle, H. Quintián, & E. Corchado (Eds), *18th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2023)* (pp. 334–343). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-42536-3_32

Burns, T., Cosgrove, J., & Doyle, F. (2019). A review of Interoperability Standards for Industry 4.0. *Procedia Manufacturing*, *38*, 646–653.

Cahuantzi, R., Chen, X., & Güttel, S. (2023). *A comparison of LSTM and GRU networks for learning symbolic sequences*. 771–785.

Carayannis, E. G., & Morawska-Jancelewicz, J. (2022). The Futures of Europe: Society 5.0 and Industry 5.0 as Driving Forces of Future Universities. *Journal of the Knowledge Economy*, *13*(4), 3445–3471. https://doi.org/10.1007/s13132-021-00854-2

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., & Stoica, I. (2025). *Why Do Multi-Agent LLM Systems Fail?* (No. arXiv:2503.13657). arXiv. https://doi.org/10.48550/arXiv.2503.13657

Chai, Z., & Zhao, C. (2020). Enhanced Random Forest With Concurrent Analysis of Static and Dynamic Nodes for Industrial Fault Classification. *IEEE Transactions on Industrial Informatics*, *16*(1), 54–66. IEEE Transactions on Industrial Informatics. https://doi.org/10.1109/TII.2019.2915559

Chander, B., Pal, S., De, D., & Buyya, R. (2022). Artificial intelligence-based internet of things for industry 5.0. *Artificial Intelligence-Based Internet of Things Systems*, 3–45.

Chatbot Arena (formerly LMSYS): Free AI Chat to Compare & Test Best AI Chatbots. (n.d.). Retrieved 4 March 2025, from https://gradio.app/

Chen, K.-Y., & Wang, C.-H. (2007). A hybrid SARIMA and support vector machines in forecasting the production values of the machinery industry in Taiwan. *Expert Systems with Applications*, *32*(1), 254–264. https://doi.org/10.1016/j.eswa.2005.11.027

Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T. (2018). Autoencoder-based network anomaly detection. *2018 Wireless Telecommunications Symposium (WTS)*, 1–5. https://doi.org/10.1109/WTS.2018.8363930

Cho, K., Merrienboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation* (No. arXiv:1406.1078). arXiv. https://doi.org/10.48550/arXiv.1406.1078

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113–124.

Cohere, T., Ahmadian, A., Ahmed, M., Alammar, J., Alizadeh, M., Alnumay, Y., Althammer, S., Arkhangorodsky, A., Aryabumi, V., & Aumiller, D. (2025). Command a: An enterprise-ready large language model. *arXiv Preprint arXiv:2504.00698*.

Corchado, J. M., & Laza, R. (2003). Constructing deliberative agents with case-based reasoning technology. *International Journal of Intelligent Systems*, *18*(12), 1227–1241.

Culot, G., Nassimbeni, G., Orzes, G., & Sartor, M. (2020). Behind the definition of Industry 4.0: Analysis and open questions. *International Journal of Production Economics*, *226*, 107617. https://doi.org/10.1016/j.ijpe.2020.107617

Dalenogare, L. S., Benitez, G. B., Ayala, N. F., & Frank, A. G. (2018). The expected contribution of Industry 4.0 technologies for industrial performance. *International Journal of Production Economics*, *204*, 383–394. https://doi.org/10.1016/j.ijpe.2018.08.019

Dash, A., Singh, A., Jain, A., Shukla, A., Mishra, H., Vyas, P., & Kaur, S. (2023). Stock Price Analysis and Prediction Using Seq2Seq LSTM. In N. Chaki, N. D. Roy, P. Debnath, & K. Saeed (Eds), *Proceedings of International Conference on Data Analytics and Insights, ICDAI 2023* (pp. 655–666). Springer Nature. https://doi.org/10.1007/978-981-99-3878-0_56

Davey, A. M., & Flores, B. E. (1993). Identification of seasonality in time series: A note. *Mathematical and Computer Modelling*, *18*(6), 73–81. https://doi.org/10.1016/0895-7177(93)90126-J

De Gooijer, J. G., & Hyndman, R. J. (2006). 25 years of time series forecasting. *International Journal of Forecasting*, *22*(3), 443–473. https://doi.org/10.1016/j.ijforecast.2006.01.001

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing. *Journal of the American Statistical Association*, *106*(496), 1513–1527. https://doi.org/10.1198/jasa.2011.tm09771

Dehariya, V. K., Shrivastava, S. K., & Jain, R. C. (2010). Clustering of Image Data Set Using K-Means and Fuzzy K-Means Algorithms. *2010 International Conference on Computational Intelligence and Communication Networks*, 386–391. https://doi.org/10.1109/CICN.2010.80

Developers, T. (2022). TensorFlow. *Zenodo*.

Discord—Group Chat That's All Fun & Games. (n.d.). Retrieved 16 June 2025, from https://discord.com

Dorri, A., Kanhere, S. S., & Jurdak, R. (2018). Multi-Agent Systems: A Survey. *IEEE Access*, *6*, 28573–28593. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2831228

Drew, D. S. (2021). Multi-Agent Systems for Search and Rescue Applications. *Current Robotics Reports*, *2*(2), 189–200. https://doi.org/10.1007/s43154-021-00048-3

Egashira, K., Vero, M., Staab, R., He, J., & Vechev, M. (2024, May 28). *Exploiting LLM Quantization*. arXiv.Org. https://arxiv.org/abs/2405.18137v2

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Fahim, A. (2021). K and starting means for k-means algorithm. *Journal of Computational Science*, *55*, 101445. https://doi.org/10.1016/j.jocs.2021.101445

Fan, Z., Xie, J., Wang, Z., Liu, P.-C., Qu, S., & Huo, L. (2021). Image Classification Method Based on Improved KNN Algorithm. *Journal of Physics: Conference Series*, *1930*(1), 012009. https://doi.org/10.1088/1742-6596/1930/1/012009

Febriyanti, S., Pradana, W. A., Muhammad, J. S., & Widodo, E. (2021). Forecasting the Consumer Price Index in Yogyakarta by Using the Double Exponential Smoothing Method. *Parameter: Journal of Statistics*, *2*(1), Article 1. https://doi.org/10.22487/27765660.2021.v2.i1.15641

Feghali, J., Jimenez, A. E., Schilling, A. T., & Azad, T. D. (2022). Overview of Algorithms for Natural Language Processing and Time Series Analyses. *Acta Neurochirurgica Supplement*, *134*, 221–242. https://doi.org/10.1007/978-3-030-85292-4_26

Ferbar Tratar, L., & Strmčnik, E. (2016). The comparison of Holt–Winters method and Multiple regression method: A case study. *Energy*, *109*, 266–276. https://doi.org/10.1016/j.energy.2016.04.115

Ferber, J., Gutknecht, O., & Michel, F. (2004). From Agents to Organizations: An Organizational View of Multi-agent Systems. In P. Giorgini, J. P. Müller, & J. Odell (Eds), *Agent-Oriented Software Engineering IV* (pp. 214–230). Springer. https://doi.org/10.1007/978-3-540-24620-6_15

Finin, T., Fritzson, R., McKay, D., & McEntire, R. (1994). KQML as an agent communication language. *Proceedings of the Third International Conference on Information and Knowledge Management*, 456–463. https://doi.org/10.1145/191246.191322

Fourth Industrial Revolution—An overview | ScienceDirect Topics. (n.d.). Retrieved 3 June 2025, from https://www.sciencedirect.com/topics/social-sciences/fourth-industrial-revolution

Fraga-Lamas, P., Varela-Barbeito, J., & Fernández-Caramés, T. M. (2021). Next Generation Auto-Identification and Traceability Technologies for Industry 5.0: A Methodology and Practical Use Case for the Shipbuilding Industry. *IEEE Access*, *9*, 140700–140730. https://doi.org/10.1109/ACCESS.2021.3119775

Francis, W. N., & Kucera, H. (1979). Brown corpus manual. *Letters to the Editor*, *5*(2), 7.

Fuller, W. A. (1995). *Introduction to Statistical Time Series*. John Wiley & Sons.

GARDNER, E. S. (1985). Exponential Smoothing: The State of the Art: ABSTRACT. *Journal of Forecasting (Pre-1986)*, *4*(1), 1.

Gillispie, C. C. (1957). The Natural History of Industry. *Isis*, *48*(4), 398–407. https://doi.org/10.1086/348606

Gou, F., Liu, J., Xiao, C., & Wu, J. (2024). Research on artificial-intelligence-assisted medicine: A survey on medical artificial intelligence. *Diagnostics*, *14*(14), 1472.

Grattafiori, A., Pandey, A., Kadian, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., … Ma, Z. (2024). *The Llama 3 Herd of Models* (No. arXiv:2407.21783). arXiv. https://doi.org/10.48550/arXiv.2407.21783

Guha, S., Rastogi, R., & Shim, K. (2000). Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, *25*(5), 345–366. https://doi.org/10.1016/S0306-4379(00)00022-3

Gulzar, Y., Alwan, A. A., Abdullah, R. M., Abualkishik, A. Z., & Oumrani, M. (2023). OCA: Ordered Clustering-Based Algorithm for E-Commerce Recommendation System. *Sustainability*, *15*(4), Article 4. https://doi.org/10.3390/su15042947

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN Model-Based Approach in Classification. In R. Meersman, Z. Tari, & D. C. Schmidt (Eds), *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE* (pp. 986–996). Springer. https://doi.org/10.1007/978-3-540-39964-3_62

Gupta, T., & Panda, S. P. (2019). Clustering Validation of CLARA and K-Means Using Silhouette & DUNN Measures on Iris Dataset. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 10–13. https://doi.org/10.1109/COMITCon.2019.8862199

Gupta, U., & Patil, N. (2015). Recommender system based on Hierarchical Clustering algorithm Chameleon. *2015 IEEE International Advance Computing Conference (IACC)*, 1006–1010. https://doi.org/10.1109/IADCC.2015.7154856

Gupta, V., & Mittal, M. (2018). KNN and PCA classifier with Autoregressive modelling during different ECG signal interpretation. *Procedia Computer Science*, *125*, 18–24. https://doi.org/10.1016/j.procs.2017.12.005

Hamilton, J. D. (2020). *Time Series Analysis*. Princeton University Press.

HANNOVER MESSE. (n.d.). Https://Www.Hannovermesse.De. Retrieved 11 January 2025, from https://www.hannovermesse.de/en/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hassan, S. A., & Malik, F. (2007). Multivariate GARCH modeling of sector volatility transmission. *The Quarterly Review of Economics and Finance*, *47*(3), 470–480. https://doi.org/10.1016/j.qref.2006.05.006

Hendler, J. (2008). Avoiding another AI winter. *IEEE Intelligent Systems*, *23*(02), 2–4.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Horling, B., & Lesser, V. (2004). A Survey of Multi-agent Organizational Paradigms. *The Knowledge Engineering Review*, *19*, 281–316. https://doi.org/10.1017/S0269888905000317

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021, June 17). *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv.Org. https://arxiv.org/abs/2106.09685v2

Huget, M.-P. (Ed.). (2003). *Communication in Multiagent Systems: Agent Communication Languages and Conversation Policies* (Vol. 2650). Springer. https://doi.org/10.1007/b12446

Ifenthaler, D., & Seufert, S. (2022). *Artificial Intelligence Education in the Context of Work*. Springer Nature.

Imambi, S., Prakash, K. B., & Kanagachidambaresan, G. (2021). PyTorch. *Programming with TensorFlow: Solution for Edge Computing Applications*, 87–104.

Industry 5.0: Towards more sustainable, resilient and human-centric industry - European Commission. (n.d.). Retrieved 7 June 2025, from https://research-and-innovation.ec.europa.eu/news/all-research-and-innovation-news/industry-50-towards-more-sustainable-resilient-and-human-centric-industry-2021-01-07_en

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: A review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. https://doi.org/10.1007/s10618-019-00619-1

Jacobs, P. S. (1992). *Joining statistics with NLP for text categorization*. 178–185.

Jan, Z., Ahamed, F., Mayer, W., Patel, N., Grossmann, G., Stumptner, M., & Kuusk, A. (2023). Artificial intelligence for industry 4.0: Systematic review of applications, challenges, and opportunities. *Expert Systems with Applications*, *216*, 119456. https://doi.org/10.1016/j.eswa.2022.119456

Javaid, M., Haleem, A., Singh, R. P., Khan, S., & Khan, I. H. (2023). Unlocking the opportunities through ChatGPT Tool towards ameliorating the education system. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, *3*(2), 100115. https://doi.org/10.1016/j.tbench.2023.100115

Jeffrey, P., Richard, S., & Christopher, M. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532. https://doi.org/10.3115/v1/d14-1162

Jelinek, F., Lafferty, J. D., & Mercer, R. L. (1992). *Basic methods of probabilistic context free grammars*. Springer.

JENNINGS, N. R., & WOOLDRIDGE, M. (1995). APPLYING AGENT TECHNOLOGY. *Applied Artificial Intelligence an International Journal*. https://doi.org/10.1080/08839519508945480

Jeon, W., Ko, G., Lee, J., Lee, H., Ha, D., & Ro, W. W. (2021). Chapter Six—Deep learning with GPUs. In S. Kim & G. C. Deka (Eds), *Advances in Computers* (Vol. 122, pp. 167–215). Elsevier. https://doi.org/10.1016/bs.adcom.2020.11.003

Johri, P., Khatri, S. K., Al-Taani, A. T., Sabharwal, M., Suvanov, S., & Kumar, A. (2021). *Natural language processing: History, evolution, application, and future work*. 365–375.

Jones, K. S. (n.d.). *Natural Language Processing: A Historical Review*.

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). *Bag of Tricks for Efficient Text Classification* (No. arXiv:1607.01759). arXiv. https://doi.org/10.48550/arXiv.1607.01759

Kabashkin, I., Misnevs, B., & Zervina, O. (2023). Artificial intelligence in aviation: New professionals for new technologies. *Applied Sciences*, *13*(21), 11660.

Kane, M. J., Price, N., Scotch, M., & Rabinowitz, P. (2014). Comparison of ARIMA and Random Forest time series models for prediction of avian influenza H5N1 outbreaks. *BMC Bioinformatics*, *15*(1), 276. https://doi.org/10.1186/1471-2105-15-276

Kemendi, A., Michelberger, P., & Mesjasz-Lech, A. (2022). Industry 4.0 and 5.0—Organizational and competency challenges of enterprises. *Polish Journal of Management Studies*, *Vol. 26, No. 2*. https://doi.org/10.17512/pjms.2022.26.2.13

Keshav Kolla, S. S. V., Lourenço, D. M., Kumar, A. A., & Plapper, P. (2022). Retrofitting of legacy machines in the context of Industrial Internet of Things (IIoT). *Procedia Computer Science*, *200*, 62–70. https://doi.org/10.1016/j.procs.2022.01.205

Khan, A. A., & Mohanty, S. K. (2022). A fast spectral clustering technique using MST based proximity graph for diversified datasets. *Information Sciences*, *609*, 1113–1131. https://doi.org/10.1016/j.ins.2022.07.101

Khan, K., Rehman, S. U., Aziz, K., Fong, S., & Sarasvady, S. (2014). DBSCAN: Past, present and future. *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, 232–238. https://doi.org/10.1109/ICADIWT.2014.6814687

Khan, M., Wu, X., Xu, X., & Dou, W. (2017). Big data challenges and opportunities in the hype of Industry 4.0. *2017 IEEE International Conference on Communications (ICC)*, 1–6. https://doi.org/10.1109/ICC.2017.7996801

Kirchgässner, G., Wolters, J., & Hassler, U. (2012). *Introduction to Modern Time Series Analysis*. Springer Science & Business Media.

Kodinariya, T. M., & Makwana, P. R. (2013). Review on determining number of Cluster in K-Means Clustering. *International Journal*, *1*(6), 90–95.

Kovari, A. (2024). Industry 5.0: Generalized definition key applications opportunities and threats. *Acta Polytechnica Hungarica*, *21*(3), 267–284.

Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). Density-based clustering. *WIREs Data Mining and Knowledge Discovery*, *1*(3), 231–240. https://doi.org/10.1002/widm.30

Kurt, R. (2019). Industry 4.0 in Terms of Industrial Relations and Its Impacts on Labour Life. *Procedia Computer Science*, *158*, 590–601. https://doi.org/10.1016/j.procs.2019.09.093

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., & Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, *6*(4), 239–242. https://doi.org/10.1007/s12599-014-0334-4

Lau, S., & Guo, P. J. (n.d.). *The Design Space of LLM-Based AI Coding Assistants: An Analysis of 90 Systems in Academia and Industry*.

Le, X.-H., Ho, H. V., Lee, G., & Jung, S. (2019). Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting. *Water*, *11*(7), Article 7. https://doi.org/10.3390/w11071387

Lee, R. S. (2023). *Natural Language Processing: A Textbook with Python Implementation*. Springer Nature.

Leng, J., Sha, W., Wang, B., Zheng, P., Zhuang, C., Liu, Q., Wuest, T., Mourtzis, D., & Wang, L. (2022). Industry 5.0: Prospect and retrospect. *Journal of Manufacturing Systems*, *65*, 279–295.

Leng, J., Sha ,Weinan, Lin ,Zisheng, Jing ,Jianbo, Liu ,Qiang, & and Chen, X. (2023). Blockchained smart contract pyramid-driven multi-agent autonomous process control for resilient individualised manufacturing towards Industry 5.0. *International Journal of Production Research*, *61*(13), 4302–4321. https://doi.org/10.1080/00207543.2022.2089929

Lenza, M., & Primiceri, G. E. (2022). How to estimate a vector autoregression after March 2020. *Journal of Applied Econometrics*, *37*(4), 688–699. https://doi.org/10.1002/jae.2895

Letchford, A., Gao, J., & Zheng, L. (2012). Optimizing the moving average. *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2012.6252797

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021a). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* (No. arXiv:2005.11401). arXiv. https://doi.org/10.48550/arXiv.2005.11401

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021b). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* (No. arXiv:2005.11401). arXiv. https://doi.org/10.48550/arXiv.2005.11401

Li, X., Lu, X., Tian, J., Gao, P., Kong, H., & Xu, G. (2009). Application of Fuzzy c-Means Clustering in Data Analysis of Metabolomics. *Analytical Chemistry*, *81*(11), 4468–4475. https://doi.org/10.1021/ac900353t

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200209.

Lim, S., & Chi, S. (2019). Xgboost application on bridge management systems for proactive damage estimation. *Advanced Engineering Informatics*, *41*, 100922. https://doi.org/10.1016/j.aei.2019.100922

Lindemann, N. F. (2023). Sealed Knowledges: A Critical Approach to the Usage of LLMs as Search Engines. *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, 985–986. https://doi.org/10.1145/3600211.3604737

Liu, Q., Chen, C., Zhang, Y., & Hu, Z. (2011). Feature selection for support vector machines with RBF kernel. *Artificial Intelligence Review*, *36*(2), 99–115. https://doi.org/10.1007/s10462-011-9205-2

Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., … Tang, J. (2023). *AgentBench: Evaluating LLMs as Agents* (No. arXiv:2308.03688). arXiv. https://doi.org/10.48550/arXiv.2308.03688

Lo, K. M., Huang, Z., Qiu, Z., Wang, Z., & Fu, J. (2024). *A Closer Look into Mixture-of-Experts in Large Language Models* (No. arXiv:2406.18219). arXiv. https://doi.org/10.48550/arXiv.2406.18219

Long, N., Gianola, D., Rosa, G. J. M., & Weigel, K. A. (2011). Application of support vector regression to genome-assisted prediction of quantitative traits. *Theoretical and Applied Genetics*, *123*(7), 1065–1074. https://doi.org/10.1007/s00122-011-1648-y

Lovato, J., Zimmerman, J. W., Smith, I., Dodds, P., & Karson, J. L. (2024). *Foregrounding artist opinions: A survey study on transparency, ownership, and fairness in AI generative art*. *7*, 905–916.

Lowther, S. D., Dimitroulopoulou, S., Foxall, K., Shrubsole, C., Cheek, E., Gadeberg, B., & Sepai, O. (2021). Low Level Carbon Dioxide Indoors—A Pollution Indicator or a Pollutant? A Health-Based Perspective. *Environments*, *8*(11), Article 11. https://doi.org/10.3390/environments8110125

Lu, W., & Wu, M. (2012). Reduced-reference quality assessment for retargeted images. *2012 19th IEEE International Conference on Image Processing*, 1497–1500. https://doi.org/10.1109/ICIP.2012.6467155

Lu, Y. (2019). Artificial intelligence: A survey on evolution, models, applications and future trends. *Journal of Management Analytics*, *6*(1), 1–29. https://doi.org/10.1080/23270012.2019.1570365

Mabkhot, M. M., Al-Ahmari, A. M., Salah, B., & Alkhalefah, H. (2018). Requirements of the Smart Factory System: A Survey and Perspective. *Machines*, *6*(2), Article 2. https://doi.org/10.3390/machines6020023

Maddikunta, P. K. R., Pham, Q.-V., B, P., Deepa, N., Dev, K., Gadekallu, T. R., Ruby, R., & Liyanage, M. (2022). Industry 5.0: A survey on enabling technologies and potential applications. *Journal of Industrial Information Integration*, *26*, 100257. https://doi.org/10.1016/j.jii.2021.100257

Majumdar, J., Udandakar, S., & Mamatha Bai, B. G. (2019). Implementation of Cure Clustering Algorithm for Video Summarization and Healthcare Applications in Big Data. In N. R. Shetty, L. M. Patnaik, H. C. Nagaraj, P. N. Hamsavath, & N. Nalini (Eds), *Emerging*

Research in Computing, Information, Communication and Applications (pp. 553–564). Springer. https://doi.org/10.1007/978-981-13-6001-5_46

Manganelli, S., & Engle, R. F. (2001). *Value at risk models in finance*.

Margherita, E. G., & Braccini, A. M. (2021). Managing industry 4.0 automation for fair ethical business development: A single case study. *Technological Forecasting and Social Change*, *172*, 121048. https://doi.org/10.1016/j.techfore.2021.121048

Martin, B., Passino, F. S., Cucuringu, M., & Luati, A. (2024). *NIRVAR: Network Informed Restricted Vector Autoregression* (No. arXiv:2407.13314). arXiv. https://doi.org/10.48550/arXiv.2407.13314

Martinez, A. R. (2012). Part-of-speech tagging. *Wiley Interdisciplinary Reviews: Computational Statistics*, *4*(1), 107–113.

Mateo, F. W., & Redchuk, A. (2024). Artificial Intelligence as a Process Optimization Driver under Industry 4.0 Framework and the Role of IIoT, a Bibliometric Analysis. *Journal of Industrial Integration and Management*, *09*(03), 357–372. https://doi.org/10.1142/S2424862222500130

Mavrotas, G. (2009). Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, *213*(2), 455–465.

Miao, Y., Gowayyed, M., & Metze, F. (2015). EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 167–174. https://doi.org/10.1109/ASRU.2015.7404790

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space* (No. arXiv:1301.3781). arXiv. https://doi.org/10.48550/arXiv.1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. In C. J. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds), *Advances in Neural Information Processing Systems* (Vol. 26). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Mitra, B., & Craswell, N. (2017, January 1). *Neural Text Embeddings for Information Retrieval*. WSDM. https://openreview.net/forum?id=qhshSS1dWT

Mohammed, A. (2024). Deep Fake Detection and Mitigation: Securing Against AI-Generated Manipulation. *Journal of Computational Innovation*, *4*(1).

Mokyr, J. (1998). *The second industrial revolution, 1870-1914*.

Mowery, D. C. (2009). Plus ca change: Industrial R&D in the “third industrial revolution”. *Industrial and Corporate Change*, *18*(1), 1–50.

Nath, P., Saha, P., Middya, A. I., & Roy, S. (2021). Long-term time-series pollution forecast using statistical and deep learning methods. *Neural Computing and Applications*, *33*(19), 12551–12570. https://doi.org/10.1007/s00521-021-05901-2

Ng, R. T., & Han, J. (2002). CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, *14*(5), 1003–1016. IEEE Transactions on Knowledge and Data Engineering. https://doi.org/10.1109/TKDE.2002.1033770

Niazkar, M., Menapace, A., Brentan, B., Piraei, R., Jimenez, D., Dhawan, P., & Righetti, M. (2024). Applications of XGBoost in water resources engineering: A systematic literature review (Dec 2018–May 2023). *Environmental Modelling & Software*, *174*, 105971. https://doi.org/10.1016/j.envsoft.2024.105971

Noh, S.-H. (2021). Analysis of Gradient Vanishing of RNNs and Performance Comparison. *Information*, *12*(11), Article 11. https://doi.org/10.3390/info12110442

Nourmohammadi, A., Fathi, M., Arbaoui, T., & Slama, I. (2024). Multi-objective optimization of cycle time and robot energy expenditure in human-robot collaborated assembly lines. *Procedia Computer Science*, *232*, 1279–1288. https://doi.org/10.1016/j.procs.2024.01.126

Oliveira, F., Gomes, L., & Vale, Z. (2025). Retrieval-Augmented Generation Powered by a Multi-agent System to Assisted the Operation of Industries. In P. Mathieu & F. De la Prieta (Eds), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Digital Twins: The PAAMS Collection* (pp. 208–216). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-70415-4_18

Oliveira, F., Lezama, F., Gomes, L., Soares, J., & Vale, Z. (2023). Risk assessment model based on centrifugal governors and artificial neural networks. *2023 IEEE Conference on Artificial Intelligence (CAI)*, 218–219. https://doi.org/10.1109/CAI54212.2023.00101

ollama. (n.d.). *Ollama/docs/modelfile.md at main · ollama/ollama*. GitHub. Retrieved 29 June 2025, from https://github.com/ollama/ollama/blob/main/docs/modelfile.md

Pang, R. Y., Schroeder, H., Smith, K. S., Barocas, S., Xiao, Z., Tseng, E., & Bragg, D. (2025). Understanding the LLM-ification of CHI: Unpacking the Impact of LLMs at CHI through a Systematic Literature Review. *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 1–20. https://doi.org/10.1145/3706598.3713726

Park, C. Y., Kim, J. W., Kim, B., & Lee, J. (2020). Prediction for Manufacturing Factors in a Steel Plate Rolling Smart Factory Using Data Clustering-Based Machine Learning. *IEEE Access*, *8*, 60890–60905. IEEE Access. https://doi.org/10.1109/ACCESS.2020.2983188

Patwardhan, N., Marrone, S., & Sansone, C. (2023). Transformers in the Real World: A Survey on NLP Applications. *Information*, *14*(4), Article 4. https://doi.org/10.3390/info14040242

Pereira, A., Costa, R. L. da, Gonçalves, R., Pereira, L., & Dias, Á. (2023). Industry 4.0 in Portugal—The state of the art. *International Journal of Internet Manufacturing and Services*, *9*(1), 44–70. https://doi.org/10.1504/IJIMS.2023.129286

Peres, R. S., Jia, X., Lee, J., Sun, K., Colombo, A. W., & Barata, J. (2020). Industrial Artificial Intelligence in Industry 4.0—Systematic Review, Challenges and Outlook. *IEEE Access*, *8*, 220121–220139. https://doi.org/10.1109/ACCESS.2020.3042874

Peter Bonasso, R., James Firby, R., Gat, E., Kortenkamp, D., Miller, D. P., & Slack, M. G. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, *9*(2–3), 237–256.

PostgreSQL, B. (1996). PostgreSQL. *Web Resource: Http://Www. PostgreSQL. Org/About*.

Pratiwi, N. B. I., & Saputro, D. R. S. (2020). Fuzzy c-shells clustering algorithm. *Journal of Physics: Conference Series*, *1613*(1), 012006. https://doi.org/10.1088/1742-6596/1613/1/012006

Prestwich, S. D., Tarim, S. A., Rossi, R., & Hnich, B. (2014). Forecasting intermittent demand by hyperbolic-exponential smoothing. *International Journal of Forecasting*, *30*(4), 928–933. https://doi.org/10.1016/j.ijforecast.2014.01.006

Qiao, B., Li, L., Zhang, X., He, S., Kang, Y., Zhang, C., Yang, F., Dong, H., Zhang, J., Wang, L., Ma, M., Zhao, P., Qin, S., Qin, X., Du, C., Xu, Y., Lin, Q., Rajmohan, S., & Zhang, D. (2024). *TaskWeaver: A Code-First Agent Framework* (No. arXiv:2311.17541). arXiv. https://doi.org/10.48550/arXiv.2311.17541

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., & Sun, M. (2023). *ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs* (No. arXiv:2307.16789). arXiv. https://doi.org/10.48550/arXiv.2307.16789

Rame, R., Purwanto, P., & Sudarno, S. (2024). Industry 5.0 and sustainability: An overview of emerging trends and challenges for a green future. *Innovation and Green Development*, *3*(4), 100173. https://doi.org/10.1016/j.igd.2024.100173

Ran, X., Xi, Y., Lu, Y., Wang, X., & Lu, Z. (2023). Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, *56*(8), 8219–8264. https://doi.org/10.1007/s10462-022-10366-3

Ratnaparkhi, A. (1999). Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning*, *34*(1), 151–175. https://doi.org/10.1023/A:1007502103375

Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 Laying down Harmonised Rules on Artificial Intelligence and Amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA Relevance) (2024). http://data.europa.eu/eli/reg/2024/1689/oj/eng

Ribeiro, B., Pereira, H., Gomes, L., & Vale, Z. (2022). *Python-Based Ecosystem for Agent Communities Simulation* (pp. 62–71). https://doi.org/10.1007/978-3-031-18050-7_7

Romelaer, P. (2002). *Organization: A Diagnosis Method*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Ruspini, E. H., Bezdek, J. C., & Keller, J. M. (2019). Fuzzy Clustering: A Historical Perspective. *IEEE Computational Intelligence Magazine*, *14*(1), 45–55. https://doi.org/10.1109/MCI.2018.2881643

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: A modern approach*. Pearson. https://thuvienso.hoasen.edu.vn/handle/123456789/8967

Sahoo, B. B., Jha, R., Singh, A., & Kumar, D. (2019). Application of Support Vector Regression for Modeling Low Flow Time Series. *KSCE Journal of Civil Engineering*, *23*(2), 923–934. https://doi.org/10.1007/s12205-018-0128-1

Salton, G., & Buckley, C. (1987). *Term Weighting Approaches in Automatic Text Retrieval* [Technical Report]. Cornell University.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, *18*(11), 613–620. https://doi.org/10.1145/361219.361220

Sasaki, M., & Shinnou, H. (2005). Spam detection using text clustering. *2005 International Conference on Cyberworlds (CW'05)*, 4 pp. – 319. https://doi.org/10.1109/CW.2005.83

Schubert, E., & Rousseeuw, P. J. (2021). Fast and eager k<math><mi is="true">k</mi></math>-medoids clustering: O(k)<math><mrow is="true"><mi is="true">O</mi><mrow is="true"><mo is="true">(</mo><mi is="true">k</mi><mo is="true">)</mo></mrow></mrow></math> runtime improvement of the PAM, CLARA, and CLARANS algorithms. *Information Systems*, *101*, 101804. https://doi.org/10.1016/j.is.2021.101804

Schumacher, A., Erol, S., & Sihn, W. (2016). A Maturity Model for Assessing Industry 4.0 Readiness and Maturity of Manufacturing Enterprises. *Procedia CIRP*, *52*, 161–166. https://doi.org/10.1016/j.procir.2016.07.040

Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification. *Augmented Human Research*, *5*(1), 12. https://doi.org/10.1007/s41133-020-00032-0

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423.

Sharan, R., & Shamir, R. (2000). CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis. *Proceedings / ... International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, *8*, 307–316.

Shekhar, S., Dubey, T., Mukherjee, K., Saxena, A., Tyagi, A., & Kotla, N. (2024, January 29). *Towards Optimizing the Costs of LLM Usage*. arXiv.Org. https://arxiv.org/abs/2402.01742v1

Shen, Z. (2024). *LLM With Tools: A Survey* (No. arXiv:2409.18807). arXiv. https://doi.org/10.48550/arXiv.2409.18807

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

Siami-Namini, S., Tavakoli, N., & Siami Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1394–1401. https://doi.org/10.1109/ICMLA.2018.00227

Singh, A. K., & Shashi, M. (2019). Vectorization of Text Documents for Identifying Unifiable News Articles. *International Journal of Advanced Computer Science and Applications*, *10*(7). https://doi.org/10.14569/IJACSA.2019.0100742

Singh, D., Padgham, L., & Logan, B. (2016). Integrating BDI Agents with Agent-Based Simulation Platforms. *Autonomous Agents and Multi-Agent Systems*, *30*(6), 1050–1071. https://doi.org/10.1007/s10458-016-9332-x

Sofat, A., & Sodhi, B. (2024). Speeding up government procurement workflows with llms. *International Conference on Electronic Government and the Information Systems Perspective*, 27–33.

Soori, M., Arezoo, B., & Dastres, R. (2023). Internet of things for smart factories in industry 4.0, a review. *Internet of Things and Cyber-Physical Systems*, *3*, 192–204. https://doi.org/10.1016/j.iotcps.2023.04.006

Srihari, G., Kusuma, T., Db, C., Kumar, J. S., & Aluvala, R. (2024). Predictive modeling of return volatility in sustainable investments: An in-depth analysis of ARIMA, GARCH, and ARCH techniques. *Investment Management and Financial Innovations*, *21*, 213–228. https://doi.org/10.21511/imfi.21(1).2024.17

Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W., & Renz, W. (2007). Validation of BDI Agents. In R. H. Bordini, M. Dastani, J. Dix, & A. E. F. Seghrouchni (Eds), *Programming*

Multi-Agent Systems (pp. 185–200). Springer. https://doi.org/10.1007/978-3-540-71956-4_11

Sung, J. H., & Cho, Y. S. (2019). Machine Learning Approach for Pattern Analysis of Energy Consumption in Factory. *KIPS Transactions on Computer and Communication Systems*, *8*(4), 87–92. https://doi.org/10.3745/KTCCS.2019.8.4.87

Suthaharan, S. (2016). Support vector machine. In *Machine learning models and algorithms for big data classification: Thinking with examples for effective learning* (pp. 207–235). Springer.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, *2*, 3104–3112.

Sutton, C., & McCallum, A. (2012). An Introduction to Conditional Random Fields. *Foundations and Trends® in Machine Learning*, *4*(4), 267–373. https://doi.org/10.1561/2200000013

Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn treebank: An overview. *Treebanks: Building and Using Parsed Corpora*, 5–22.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., Tafti, P., Hussenot, L., Sessa, P. G., Chowdhery, A., Roberts, A., Barua, A., Botev, A., Castro-Ros, A., Slone, A., … Kenealy, K. (2024, March 13). *Gemma: Open Models Based on Gemini Research and Technology*. arXiv.Org. https://arxiv.org/abs/2403.08295v4

Teixeira, J. E., & Tavares-Lehmann, A. T. C. P. (2022). Industry 4.0 in the European union: Policies and national strategies. *Technological Forecasting and Social Change*, *180*, 121664. https://doi.org/10.1016/j.techfore.2022.121664

Topchy, A., Jain, A. K., & Punch, W. (2004). A Mixture Model for Clustering Ensembles. In *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM)* (pp. 379–

390). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611972740.35

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., … Scialom, T. (2023, July 18). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv.Org. https://arxiv.org/abs/2307.09288v2

Trafalis, T. B., & Ince, H. (2000). Support vector machine for regression and applications to financial forecasting. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, *6*, 348–353 vol.6. https://doi.org/10.1109/IJCNN.2000.859420

Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2023). Graph clustering with graph neural networks. *Journal of Machine Learning Research*, *24*(127), 1–21.

Tuggener, L., Sager, P., Taoudi-Benchekroun, Y., Grewe, B. F., & Stadelmann, T. (2024). So you want your private LLM at home? A survey and benchmark of methods for efficient GPTs. *2024 11th IEEE Swiss Conference on Data Science (SDS)*, 205–212. https://doi.org/10.1109/SDS60720.2024.00036

Turing, A. M. (2009). Computing Machinery and Intelligence. In R. Epstein, G. Roberts, & G. Beber (Eds), *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer* (pp. 23–65). Springer Netherlands. https://doi.org/10.1007/978-1-4020-6710-5_3

Tyralis, H., & Papacharalampous, G. (2017). Variable Selection in Time Series Forecasting Using Random Forests. *Algorithms*, *10*(4), Article 4. https://doi.org/10.3390/a10040114

V, R. C., S, S. P., H, Keerthana., P, Abhignya., & M, Rajendiran. (2024). Chatbot for Government Schemes Using SEQ2SEQ Model. *2024 Second International Conference on Advances* *in Information Technology (ICAIT)*, *1*, 1–6. https://doi.org/10.1109/ICAIT61638.2024.10690438

van der Hoek, W., & Wooldridge, M. (2008). Chapter 24 Multi-Agent Systems. In F. van Harmelen, V. Lifschitz, & B. Porter (Eds), *Foundations of Artificial Intelligence* (Vol. 3, pp. 887–928). Elsevier. https://doi.org/10.1016/S1574-6526(07)03024-6

Varadi, M., Bertoni, D., Magana, P., Paramval, U., Pidruchna, I., Radhakrishnan, M., Tsenkov, M., Nair, S., Mirdita, M., & Yeo, J. (2024). AlphaFold Protein Structure Database in 2024: Providing structure coverage for over 214 million protein sequences. *Nucleic Acids Research*, *52*(D1), D368–D375.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need* (No. arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762

Voges, S., Becker, K.-F., Schuetze, D., Schröder, B., Fruehauf, P., Heimann, M., Nerreter, S., Blank, R., Gottwald, S., Hofmeister, A., Schmied, M., Kreitmair, M., Fust, R., Neifer, W., Braun, T., Thomas, T., Schneider-Ramelow, M., & Lang, K.-D. (2019). Highly Miniaturized Integrated Sensor Nodes for Industry 4.0. *International Symposium on Microelectronics*, *2019*, 000415–000422. https://doi.org/10.4071/2380-4505-2019.1.000415

Wan, H., Wang, H., Scotney, B., & Liu, J. (2019). A Novel Gaussian Mixture Model for Classification. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 3298–3303. https://doi.org/10.1109/SMC.2019.8914215

Wang, J., Rao, C., Goh, M., & Xiao, X. (2023). Risk assessment of coronary heart disease based on cloud-random forest. *Artificial Intelligence Review*, *56*(1), 203–232. https://doi.org/10.1007/s10462-022-10170-z

Wang, Z., Lai, C., Chen, X., Yang, B., Zhao, S., & Bai, X. (2015). Flood hazard risk assessment model based on random forest. *Journal of Hydrology*, *527*, 1130–1141. https://doi.org/10.1016/j.jhydrol.2015.06.008

Weightman, G. (2010). *The Industrial Revolutionaries: The Making of the Modern World, 1776–1914*. Open Road + Grove/Atlantic.

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.

Welcome to discord.py. (n.d.). Retrieved 16 June 2025, from https://discordpy.readthedocs.io/en/stable/

What is ChatGPT? | OpenAI Help Center. (n.d.). Retrieved 22 June 2025, from https://help.openai.com/en/articles/6783457-what-is-chatgpt

Widder, D. G., Whittaker, M., & West, S. M. (2024). Why 'open' AI systems are actually closed, and why this matters. *Nature*, *635*(8040), 827–833. https://doi.org/10.1038/s41586-024-08141-1

Wilks, Y. (2009). It Works but How Far Can It Go: Evaluating the SYSTRAN MT System. In Y. Wilks (Ed.), *Machine Translation: Its Scope and Limits* (pp. 65–86). Springer US. https://doi.org/10.1007/978-0-387-72774-5_4

Wu, J., Yang, S., Zhan, R., Yuan, Y., Chao, L. S., & Wong, D. F. (2025). A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions. *Computational Linguistics*, *51*(1), 275–338. https://doi.org/10.1162/coli_a_00549

Wu, K.-L., & Yang, M.-S. (2007). Mean shift-based clustering. *Pattern Recognition*, *40*(11), 3035–3052. https://doi.org/10.1016/j.patcog.2007.02.006

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., & Wang, C. (2023, August 16). *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation*. arXiv.Org. https://arxiv.org/abs/2308.08155v2

Wu, Y., Yuan, M., Dong, S., Lin, L., & Liu, Y. (2018). Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing*, *275*, 167–179. https://doi.org/10.1016/j.neucom.2017.05.063

XGBoost Model and Its Application to Personal Credit Evaluation | IEEE Journals & Magazine | IEEE Xplore. (n.d.). Retrieved 9 February 2025, from https://ieeexplore.ieee.org/abstract/document/8988224

Xia, J., & Wang, H. (2023). A Sequence-to-Sequence Approach with Mixed Pointers to Topic Segmentation and Segment Labeling. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2683–2693. https://doi.org/10.1145/3580305.3599245

XMPP. (2024). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=XMPP&oldid=1214756079

Xu, D., & Tian, Y. (2015). A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science*, *2*(2), 165–193. https://doi.org/10.1007/s40745-015-0040-1

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, *16*(3), 645–678. IEEE Transactions on Neural Networks. https://doi.org/10.1109/TNN.2005.845141

Xu, S. (2018). Bayesian Naïve Bayes classifiers to text classification. *Journal of Information Science*, *44*(1), 48–59. https://doi.org/10.1177/0165551516677946

Xu, X., Ester, M., Kriegel, H.-P., & Sander, J. (1998). A distribution-based clustering algorithm for mining in large spatial databases. *Proceedings 14th International Conference on Data Engineering*, 324–331. https://doi.org/10.1109/ICDE.1998.655795

Xu, X., Lu, Y., Vogel-Heuser, B., & Wang, L. (2021). Industry 4.0 and Industry 5.0—Inception, conception and perception. *Journal of Manufacturing Systems*, *61*, 530–535. https://doi.org/10.1016/j.jmsy.2021.10.006

Yang, J., Parikh, D., & Batra, D. (2016). Joint Unsupervised Learning of Deep Representations and Image Clusters. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5147–5156. https://doi.org/10.1109/CVPR.2016.556

Ye, D., Zhang, M., & Vasilakos, A. V. (2017). A Survey of Self-Organization Mechanisms in Multiagent Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *47*(3), 441–461. IEEE Transactions on Systems, Man, and Cybernetics: Systems. https://doi.org/10.1109/TSMC.2015.2504350

Ye, R., Tang, S., Ge, R., Du, Y., Yin, Z., Chen, S., & Shao, J. (2025, March 5). *MAS-GPT: Training LLMs to Build LLM-based Multi-Agent Systems*. arXiv.Org. https://arxiv.org/abs/2503.03686v1

Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

Yousuf, M. U., Al-Bahadly, I., & Avci, E. (2022). Wind speed prediction for small sample dataset using hybrid first-order accumulated generating operation-based double exponential smoothing model. *Energy Science & Engineering*, *10*(3), 726–739. https://doi.org/10.1002/ese3.1047

Zhang, C., Tang, D., Liu, Y., & You, J. (2008). *A multi-agent architecture for knowledge management system*. *5*, 433–437.

Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A New Data Clustering Algorithm and Its Applications. *Data Mining and Knowledge Discovery*, *1*(2), 141–182. https://doi.org/10.1023/A:1009783824328

Zheng, C., Yin, F., Zhou, H., Meng, F., Zhou, J., Chang, K.-W., Huang, M., & Peng, N. (2024). *On Prompt-Driven Safeguarding for Large Language Models*. arXiv. https://doi.org/10.48550/arXiv.2401.18018

Zhou, J., Qin, Y., Chen, D., Liu, F., & Qian, Q. (2022). Remaining useful life prediction of bearings by a new reinforced memory GRU network. *Advanced Engineering Informatics*, *53*, 101682. https://doi.org/10.1016/j.aei.2022.101682

Zivot, E., & Wang, J. (Eds). (2006). Vector Autoregressive Models for Multivariate Time Series. In *Modeling Financial Time Series with S-PLUS®* (pp. 385–429). Springer. https://doi.org/10.1007/978-0-387-32348-0_11