NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

Henrique Duarte Mota dos Santos Chia
BSc in Electrical and Computer Engineering

Implementation of an Intelligent Virtual Assistant based on LLM for Irrigation Optimization

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING
NOVA University Lisbon
September, 2025

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENT OF
ELECTRICAL AND COMPUTER ENGINEERING

IMPLEMENTATION OF AN INTELLIGENT VIRTUAL ASSISTANT BASED ON LLM FOR IRRIGATION OPTIMIZATION

HENRIQUE DUARTE MOTA DOS SANTOS CHIA

BSc in Electrical and Computer Engineering

Adviser: Ana Inês Oliveira
Assistant Professor, NOVA University Lisbon, School of Science and Technology | FCT NOVA

Examination Committee:

Chair: Pedro Miguel Ribeiro Pereira,
Assistant Professor, *NOVA University Lisbon, School of Science and Technology | FCT NOVA*

Rapporteurs: Filipa Alexandra Moreira Ferrada,
Assistant Professor, *NOVA University Lisbon, School of Science and Technology | FCT NOVA*

Adviser: Ana Inês Oliveira
Assistant Professor, NOVA University Lisbon, School of Science and Technology | FCT NOVA

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2025

IMPLEMENTATION OF AN INTELLIGENT VIRTUAL ASSISTANT BASED ON LLM MODELS FOR IRRIGATION OPTIMIZATION

Copyright © Henrique Chia, NOVA School of Science and Technology, NOVA University Lisbon.

The NOVA School of Science and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

To my parents, friends and everyone who made this possible

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor, Professor Ana Inês Oliveira, for accepting this challenge with me, for the guidance provided, and for the patience shown to me throughout this thesis.

I would like to thank the NOVA School of Science and Technology. This institution, which provided an excellent environment for my growth, not only academically, but also as a person, through contact with the excellent professors and colleagues.

I am also grateful for the opportunity provided by Hidrosoph, which proposed an interesting challenge that gave me my first contact with a real-life environment of an organization. Specifically, I thank Eng. Pedro Azevedo, the person who guided the practical implementation of the thesis, who showed constant interest, willingness, and availability to help.

To my friends, both those who have accompanied me since before my academic journey and those I have had the pleasure of meeting along the way, thank you for sharing this journey with me, and for sharing the good and the bad moments. Your presence has been a constant source of motivation and joy throughout this process.

To my family, I want to thank you for the support given throughout my life, not only for the foundation laid out for my success, but also for all the small gestures shown and every moment of understanding that made this journey lighter.

I am very lucky and grateful to have you

Thank you

"Artificial intelligence is not a replacement for humans. It's about amplifying human potential."

(Amir Husain)

Abstract

New AI assistants are surging, and the technology is already being integrated into day-to-day life and even across organizations. The agriculture field is no exception, and this is a great opportunity to make knowledge available to a wider range of stakeholders across the sector. This technology can prove useful as it can help with information such as the ideal times of planting and the predicted time of plant growth, the protocol treatments for each plant to prevent diseases and optimize growth.

This thesis explores the implementation of an AI assistant with the limitations of the available computing resources and strict information security, focusing on the development of an AI assistant that can help the user in all the steps to cultivate a healthy and productive crop yield. The system will be integrated into the already existing platform, Irristrat, which is a monitoring system for the crops, where many factors are shown, such as soil humidity, temperature, … The main factor that led the work carried out was the vision that the assistant developed would use only controlled information and documents where the information is viable and trustworthy. The thesis was carried out in collaboration with Hidrosoph, the organization that developed the Irristrat platform, and installs the sensors used with it. Hidrosoph contributed to the supervision of the work and provided support with the implementation and validation of the system developed in the scope of the thesis.

With the limitations of resources available, the solutions discussed always had a factor in common, which is the use of an already existing Large Language Model. Some other options were discussed, like the fine-tuning method, but the RAG method is what ended up being explored.

The system proposed in this thesis involves the integration of a vector database, with the role of saving all the information fed to the system, with the main objective of being easy to access/gather. This allows for the system to be possible to be used in a real-time scenario, together with a locally run Large Language Model (LLM) in charge of understanding the user's

questions/conversation and using the gathered information to formulate a meaningful answer for the user

After the implementation, the system was evaluated in terms of response quality and performance. The results show that the proposed RAG-based assistant is capable of generating coherent and context-aware agronomic responses, demonstrating acceptable performance under constrained computational resources. A qualitive evaluation conducted by agronomy engineers confirmed the relevance and technical consistency of the generated answers, highlighting the feasibility of the proposed approach as a decision-support tool for irrigation optimization.

Keywords: Chat Bot, Virtual Assistant, RAG, LLM, Agriculture

RESUMO

Novos assistentes de inteligência artificial estão a surgir e a tecnologia já está a ser integrada na vida do dia a dia e até nos métodos de trabalho em organizações. O setor agrícola não é exceção, e esta é uma ótima oportunidade para tornar o conhecimento disponível a um leque maior de organizações e até indivíduos interessados no sector.

Esta tese explora a implementação de um assistente de inteligência artificial com as limitações dos recursos computacionais disponíveis e uma necessária segurança da informação, focando no desenvolvimento de um assistente IA que consegue ajudar os utilizadores em todos os passos para cultivar uma colheita saudável e produtiva. O sistema desenvolvido vai ser integrado na já existente plataforma, Irristrat, que é um sistema de monitoramento de plantações, que apresenta vários dados como a temperatura, humidade do solo, … O fator principal que guiou o projeto, foi a visão de que o assistente desenvolvido apenas utiliza informação controlada e documentos onde a informação é controlada e confiável. O projeto foi levado em colaboração com a Hidrosoph, a empresa que desenvolveu a plataforma Irristrat e que instala os sensores utilizados na recolha de dados. A Hidrosoph contribuiu para a supervisão do projeto e ajudou à implementação e validação do sistema.

Com as limitações de recursos disponíveis, as soluções ponderadas tiveram sempre um fator em comum, a utilização de Modelo de Linguagem de Grande Escala (Large Language Models - LLM) já existentes. Algumas opções foram analisadas, como o método de fine-tuning, mas o método de RAG foi o que acabou por ser explorado.

O sistema proposto envolve a integração de uma base de dados vetorial, cuja função é armazenar toda a informação fornecida ao sistema, com o principal objetivo de ser fácil de recolher a informação desejada em cenários de tempo real, em conjunto com um Modelo de Linguagem de Grande Escala (LLM) mantido num ambiente local, responsável por compreender as perguntas/conversas do utilizador e utilizar a informação recolhida para formular uma resposta correta e relevante.

Após a implementação, o sistema foi avaliado em termos da qualidade das respostas e do desempenho. Os resultados indicam que o assistente baseado em RAG proposto é capaz de gerar respostas agronómicas coerentes e contextualizadas, demonstrando um desempenho aceitável sob restrições computacionais. A avaliação qualitativa realizada por engenheiros de agronomia confirmou a relevância e a consistência técnica das respostas geradas, evidenciando a viabilidade da abordagem proposta como ferramenta de apoio à decisão para a otimização da rega.

Palavras-chave: Chat Bot, Assistente Virtual, RAG, LLM, Agricultura

CONTENTS

LIST OF FIGURES

List of Tables

ACRONYMS

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AWS	Amazon Web Services
BERT	Bidirectional-Encoder-Representations from Transformers
BLOOM	BigScience Large Open-science Open-access Multilingual Language Model
CNN	Convolutional Neural Network
DB	Data Base
DPR	Dense Passage Retrieval
ELECTRA	Efficiently Learning an Encoder that Classifies Token Replacements Accurately
FAISS	Facebook Artificial Intelligence Similarity Search
GAIA	General Artificial Intelligence Assistant benchmark
GLUE	General Language Understanding Evaluation
GPS	Global Positioning System
GPT	Generative Pre-Trained Transformer
HNSW	Hierarchical Navigable Small World
LAB	Large-scale Alignment for chatBots
LLM	Large Language Model
LSTM	Long Short-Term Memory Networks
MMLU	Massive Multitask Language Understanding
NDVI	Normalized Difference Vegetation Index
MT	Machine Translation
NLM	Natural Language Model

NLP	Natural Language Processing
NLU	Natural Language Understanding
RAG	Retrieval Augmented Generation
RNN	Recurrent Neural Network
RLHF	Reinforcement Learning with Human Feedback
TAMS	Task-Adaptive Model Structures
TARP	Task-Adaptive Reparameterization
UUID	Universal Unique Identifier
XML	Extensible Markup Language
YOLO	You Only Look Once

1

INTRODUCTION

1.1 Historical Contextualization

Last century, a population boom created high concern about the capacity to feed the worldwide population. These conditions gave place to an environment where agriculture saw a big revolution. This revolution is known as the Green Revolution, and it took place somewhere between 1940 and 1970. At the time, new techniques were developed with the intent of producing more crops. With this Green Revolution, we started to produce crops more densely, we discovered new irrigation techniques, new synthetic fertilizers, and new pesticides were introduced. Norman Borlaug is now considered the father of the Green Revolution, credited with saving over a billion people from starvation. Norman Borlaug was responsible for creating genetically modified crops that showed better resistance to diseases. On top of that, these were new semi-dwarfed crops, these kinds of crops were more productive, losing less grains until they were harvested (Ameen & Raza, 2017).

With the food shortage, the agricultural policies were revised, and the agricultural machinery industry boomed, agriculture began a new transition, it is not a walking task anymore, but a driving work. These changes lead to a present where we have a machine for almost every task, making it easier and faster.

1.2 Change Necessity and New Strategies

In 1960 there were 3 billion humans on the planet, by 2050 we are expected to reach the number of 9 billion inhabitants, so in less than 100 years we are expected to triple the world population. With this fast-paced evolution, the quantity of food that we produce needs to get bigger, but there are more problems than just producing food for everyone, old ways are not sustainable, may increase risk to ecosystems and may not be the most ethical possible.

There is a relatively new trend called Precision Agriculture. This strategy tries to read every factor that can affect culture production and tries to optimize it at every level, from resource efficiency to time efficiency and space efficiency. With this strategy, we no longer look at the field as a whole, but we change the perspective to the square meters, so what can we do, with fewer resources, to produce more, in less space.

1.3 New Agriculture Revolution?

The last decades have been revolutionary, today you can't leave your home without a smartphone in your pocket, this same evolution is being explored, and it shows in every society sector.

Agriculture is no exception, at the beginning of the century the trend was to use man-handled machinery, and we are already seeing a change in this way (Lowenberg-Deboer & Erickson, 2019). There are new machines that don't even have a cab to support direct human input that will rely only on GPS location and previously mapped areas and defined routes/routines. New strategies are surging, drones are being used to run across the fields and pick ready crops one by one, at the same time they can look for diseased plants and monitor the crop's growth. Sensors are being implemented to keep track of the soil moisture and nutrients, these IoT systems are now accessible for every big exploration and with the new evolution, AI will get even more useful.

1.4 Problem Scope and Approach

The focus of this thesis is to design an intelligent assistant for agriculture management, with the aim of optimizing irrigation methods, water usage, and helping with general agriculture knowledge. The final vision fits within a chatbot, based on a Large Language Model (LLM) that understands human queries and can respond to agronomic questions.

This thesis consists of a main model in charge of understanding the human language, interacting with the user, and resuming the information collected by the database retriever that was classified as relevant to the query submitted by the user. The database is in charge of saving the information and keeping a similarity ranking that helps the retrieval process.

A secondary model was used to create testing data, processing the whole information available to the model, and creating questions to evaluate possible answers. This evaluation process consists of pushing and finding the limits of the model such as how hardware affects the limitation of the context size and a process of reiteration of the model prompt.

1.5 Dissertation Structure

This dissertation is organized as follows:

Chapter 1 introduces the dissertation context, motivation, and approach idealized for the problem presented.

Chapter 2 presents the available approaches and technologies that would be interesting to use to address the proposed problem. The current capabilities of LLMs are researched and collected, creating a starting point for the practical work side of the thesis.

Chapter 3 creates a general view of the proposed architecture, presenting the created pipeline and explaining in detail the steps that will be executed, starting with the gathering of information, to the final implementation of the chatbot

Chapter 4 explains the implementation of the utilized processes, providing an insightful view of the frameworks in use and the coding logic behind the operations.

Chapter 5 is dedicated to testing and validation processes to check the final product's capabilities, limitations, and reliability. This chapter also explores a variation of the main architecture utilized to create validation data.

Chapter 6 summarizes the work developed in this thesis, highlighting points where improvements can be implemented, outlining future work for which this thesis will serve as a foundation.

2

Background and Related Work

With the new AI evolution, it is expected that it will be seen all over society. Every year we see new uses and new adaptations of AI in day-to-day life. Every field of investigation now uses some kind of AI, be it just simple prediction models or more complex uses like the new chatbots that are emerging.

This new market is being explored, and companies are taking action. The first big example is OpenAI, which started with the image generator DALL-E, and now their main product is ChatGPT, which took over the world in a year. Another good example is Nvidia, they changed their main product from graphics cards to AI software, their market is no longer only physical products, but the ways they can make their products better and more efficient with software.

Every field of research is trying to use AI in some way. Medical researchers are using AI to predict diseases in patients, car companies are developing autopilot features that rely on computer vision and pathfinding, social media is adapting the algorithms to give better recommendations and retaining more users and showing better-targeted advertising.

In this section, the main AI capabilities are explored, analyzing how good the current models are and what the state of the assistant bots and LLMs is. Aside from AI, the current state of agriculture will be explored.

2.1 The Evolution of the Industrial Revolutions and their Impact on Agriculture

The agricultural sector stands at the forefront of the digital transformation, driven by the principles of Industry 4.0 and evolving toward the human-centric vision of Industry 5.0. Advanced technologies like artificial intelligence, big data analytics, and IoT enable precision agriculture, allowing tailored interventions and optimized resource use (Fasciolo et al., 2024). However, beyond automation, the future demands systems that empower human decision-making and prioritize sustainability.

One of the most promising applications of this technological evolution is the use of Artificial Intelligence to support precision agriculture. In particular, conversational agents powered by Large Language Models (LLMs) represent a powerful means of making agronomic expertise more accessible (Zhang et al., 2025). By translating complex technical knowledge into understandable, context-aware responses, these tools help democratize information, enabling farmers and agricultural professionals to make informed decisions regardless of their background or resources. This shift reflects the core values of Industry 5.0, not just automating tasks, but empowering people through inclusive, intelligent systems.

Understanding the emergence of Agriculture 5.0 requires a historical perspective that traces the parallel development of the industrial and agricultural sectors. To capture this progression, Figure 1 illustrates the parallel roadmaps of the agricultural and industrial revolutions.

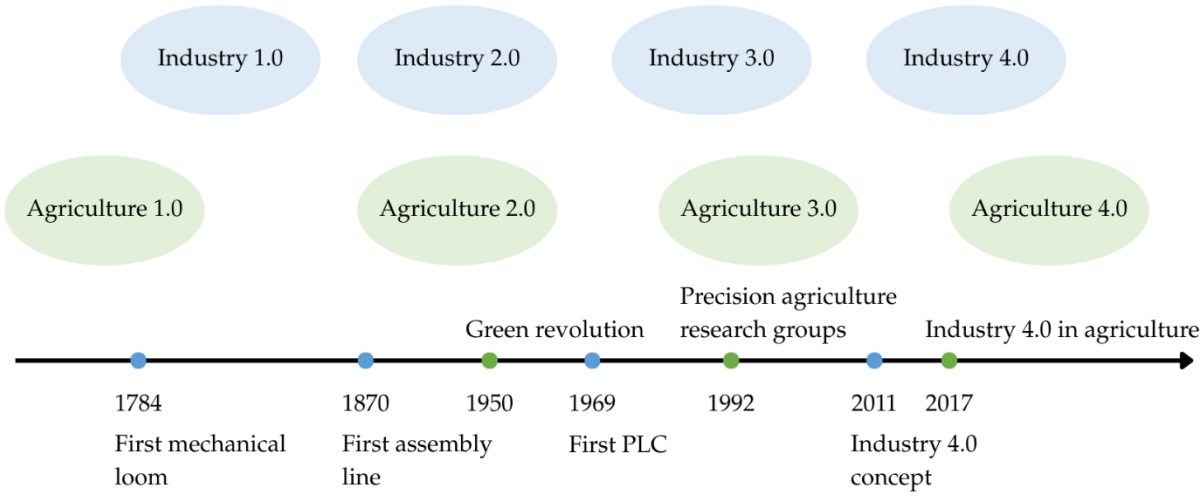

Figure 1 Timeline of Industrial and Agricultural Revolutions.

Agriculture 1.0 refers to the era of traditional farming practices that lasted until the late 19th century. It was characterized by heavy reliance on manual labor and animal power, with farmers using basic tools.

The first industrial revolution (from the end of 18th to the start of 20th century), represented by the invention of the mechanical loom in 1784, laid the groundwork for Agriculture 2.0, which emerged in the early 20th century. This era was marked by the introduction of agricultural machinery, leading to increased food production and a substantial reduction in manual labor. The second industrial revolution (from the end of the 19th century to the early 1970s) was marked by the first assembly line in 1870, brought electric power, mass production, and breakthroughs such as the internal combustion engine. The increase in available products and the new transportation methods contributed to the development of the supply chain, which also affected the agri-food supply chain, allowing agricultural goods to reach distant markets. The rise of assembly-line mass production boosted efficiency and was later adapted to livestock farming, replacing traditional home-based methods with large-scale intensive operations. The third industrial revolution (1970s-2010s) marked the invention of the programmable logic controller. Advancements in embedded systems, software engineering, and communication technologies laid the foundation for Agriculture 3.0. These technologies enabled the development of precision agriculture, including yield monitoring, variable rate applications, and GPS-guided farming systems (Aceto et al., 2019; Holzinger et al., 2024; Liu et al., 2021; Pilevari & Yavari, 2020).

The fourth industrial revolution (2011-present) is characterized by new technologies such as the Internet of Things (IoT), robotics, big data, machine learning, and blockchain technology.

These technologies enable real-time monitoring of soil, crops, and livestock, optimizing resource utilization. The integration of cloud computing with machine learning helps not only to predict trends and improve efficiency but also drives predictive analytics for yield forecasting, pest detection, and climate adaptation. There's also experimenting with autonomous machinery to automate farm operations, paving the way for fully smart farms. Additionally, blockchain enhances transparency and traceability across the agri-food supply chain (Aceto et al., 2019; Razak et al., 2024).

2.1.1 Agriculture 5.0 and Big Data

Agriculture 5.0, although still largely a theoretical concept framework, is already taking shape as the technologies that enable its transition emerge. It represents a transformative phase in the evolution of farming, driven by the urgent need to address global challenges, including rapid population growth, resource scarcity, climate change, and labor shortages. With finite resources like land, water, and energy under increasing pressure, there is a growing demand for agricultural systems that are more sustainable, efficient, and resilient (Razak et al., 2024).

Big Data lies at the core of the transition from Agriculture 4.0 to Agriculture 5.0. Modern farming systems generate unprecedented amounts of data from different tools, such as sensors and imaging. The challenge in this transition now lies in analyzing and transforming it into meaningful insights that can help in decision-making (Fountas et al., 2024; Saiz-Rubio & Rovira-Más, 2020). Agriculture 5.0 also envisions humans taking on high-level guidance roles, while autonomous machines handle repetitive tasks such as weeding, irrigating, and even complex data analysis.

When combined with knowledge graphs and enriched by real-time data from IoT systems, LLMs have the potential to create these insights that can meet each farmer's needs. This is where conversational AI systems and large language models come into play by translating complex datasets and technical knowledge into clear, context-aware guidance (Fountas et al., 2024). This human-centered integration of Big Data and artificial intelligence reflects the essence of Agriculture 5.0: not simply automating agriculture, but equipping farmers with intelligent, inclusive, and sustainable decision-support tools.

2.2 Artificial Intelligence

As mentioned before, AI has a wide range of applications, but it appears in diverse forms, and the relevant technologies for the work being developed are explored in this subsection.

2.2.1 Artificial Neural Networks

Neural networks can be applied to a wide range of problems. The general architecture of a neural network (Figure 2) uses an input layer that receives pre-processed data, often in an already simplified form. The data is then passed to the hidden layer, which may consist of multiple layers. The connections within the layers are not limited to a sequential structure, and there are other connections, like cascade or recurrent connections (Teixeira & Fernandes, 2012).

In this subsection, some of the possible architectures for neural networks will be explored, along with their main uses.

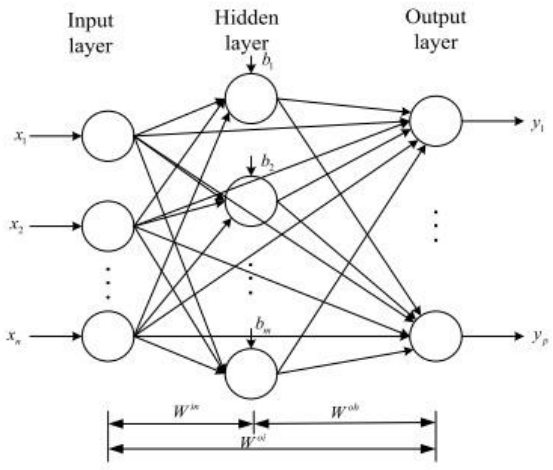

Figure 2 General ANN architecture (Zhao et al., 2020).

Convolutional Neural Network

The convolutional neural network (CNN) incorporates some additional layers to the general neural network architecture. The input is filtered by a convolutional layer, and it outputs a feature map where some patterns are detected. This feature map is used as an input for the pooling layer. The feature map input is generally very complex and shows a very high number of dimensions, so the pooling layer aggregates the features. The main objective in this layer is

to simplify the feature map and prepare it to be used in a next layer (Cui et al., 2017; Gu et al., 2018).

This type of ANN is well known because of the results obtained in the branch of computer vision and its many uses like face recognition, object detection, medical imaging. Models like YOLO (You Only Look Once) are emerging and evolving by the day. YOLO is an object detection model that is already on version 8, having varied sizes, like nano, small, medium, large and extra-large. While the nano version may not be as accurate as the large version, it shows great advantages on computational capacity and recognition time.

Long Short-Term Memory Network

The Long Short-Term Memory Networks (LSTM) are considered to be the go-to when facing problems that involve sequential and temporal data. This architecture was based on a Recurrent Neural Network (RNN), known for having a backward connection in same layers, using the layer output as an input for the same layer (Teixeira & Fernandes, 2012).

This kind of network is already in use and new applications are being explored every year, ranging from weather predictions to oil prices and sunspot occurrences.

LSTM with forget gate. The architecture of a basic LSTM with a forget gate creates an additional layer to the basic RNN where it is added a forget gate to the recurrent connection, this gate is then connected to a memory block that will be reset when the data is out of date (Gers et al., 1999). The modern representation (Figure 3) of the architecture consists of three gates, the forget gate that controls the importance of previous data, the input that controls the importance of the new information and the output gate that controls the importance of the information that will be use from the output.

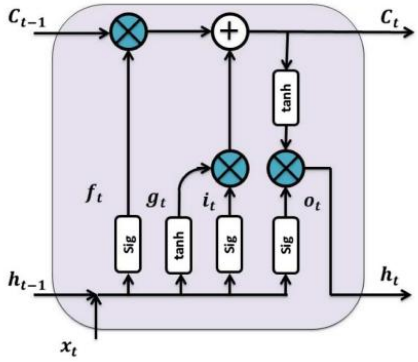

Figure 3 Modern LSTM with forget gate representation (Smagulova & James, 2019).

Bidirectional LSTM. Bidirectional LSTM (BiLSTM) are often used in speech recognition and text recognition, analyzing the sentiment of it, there are some hybrid uses with CNN to create a larger level text classification.

The BiLSTM architecture uses two LSTM Networks, one that reads the input in a forward direction and the other that reads the input backwards. Both LSTMs share the recurrent connection for the respective parts of the input, and the final output will be a combination of both LSTMs. This concept is based on a bidirectional RNN studied by (Schuster & Paliwal, 1997)

2.2.2 Large Language Models

With the new breakthrough in Large Language Models (LLM) with ChatGPT and other similar, the concept went mainstream, and everyone has heard about it. The field of Natural Language Processing (NLP) is now under the scope of researchers and enterprises.

Large Language Models can acquire skills like text generation and text translation, where the LLM is prepared to produce text on demand, or skills, enabling them to act as chatbots that maintain an engaging conversation. This last option fits the vision of the thesis, as it encompasses the ability to answer user questions within the context of agricultural management assistance.

The use of chatbots within a company environment, especially for user service, needs a very controlled and specific training environment. It also shows good upsides, like providing the user with contact for 24 hours a day, being a perfect addition, maintaining the model accuracy.

Transformer Language Models

Transformer-based Language Models (Figure 4), use the transformer architecture (encoder-decoder architecture) introduced by (Vaswani et al., 2017). This architecture is now a fundamental component for NLP tasks. This method is based on a self-attention mechanism that allows the transformer architecture to understand distant correlations between words, enabling it to predict the next word in a sequence by analyzing the previous context and exploring multiple possible continuations. This approach makes the transformer significantly more effective than traditional neural networks like RNNs and CNNs.

This type of architecture is based on encoder and decoder components:

- **The Encoder:** this component processes the input, usually a sequence of tokens used to create a simpler representation of the information. To this simplification, we call the embedding, containing a multi-dimensional feature array (Vaswani et al., 2017). An example of the encoder is represented on the left side of Figure 4.
- **The Decoder:** this component uses the encoder´s hidden state and the output's generation to create a new generation, considering the input while predicting the next output token (Vaswani et al., 2017). A representation example of the decoder is on the right side of Figure 4.

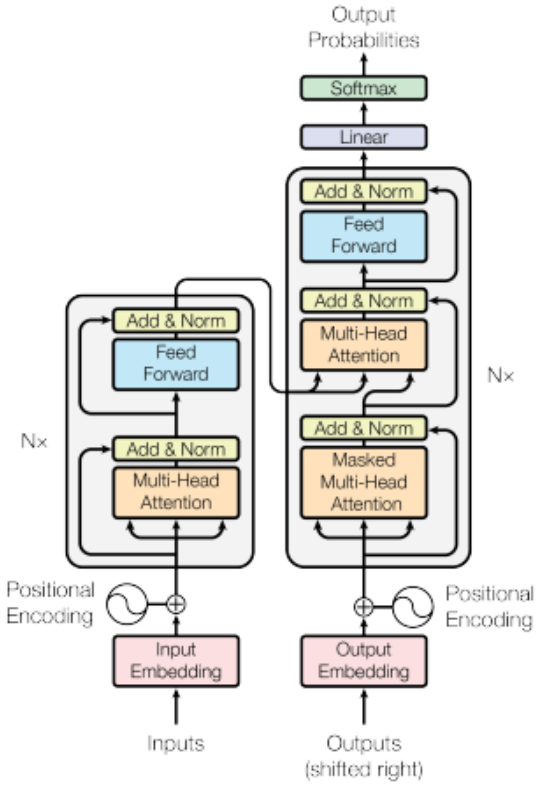

Figure 4 Transformer model architecture (Vaswani et al., 2017).

The original architecture, as shown on Figure 4, was based on an encoder-decoder model, however, the transformer model architecture served as a starting point for many well-known LLMs. and some architectures showed up with encoder-only or decoder-only models.

Examples of the encoder-only models are Bidirectional-Encoder-Representations from Transformers (BERT) (Devlin et al., 2019) and Robustly Optimized BERT Approach (RoBERTa) (Liu et al., 2019), both relying on a Bidirectional self-attention (Seo et al., 2017).

For the decoder-only we have the Generative Pre-trained Transformer (GPT) case and BigScience Large Open-science Open-access Multilingual Language Model (BLOOM) (Scao et

al., 2022), where the models predict each token in a sequence based on the tokens before, this method exceeds on activities that require generative and creative writing.

As for the encoder-decoder method, in addition to the original transformer there is the Text-to-Text Transfer Transformer (T5) (Xue et al., 2021), both using the advantage of having the encoder and decoder working together to extract complex correlations found in the normal day-to-day language.

Bidirectional Encoder Representations from Transformers (BERT)

While new models continue to emerge alongside OpenAI's decoder-only ChatGPT, Google's BERT, an encoder-only model, remains a foundational natural language model that is widely supported through open-source libraries, making the BERT model accessible for researchers all around the world.

This architecture makes sure to break the information with contextualization and tokens. The training of the NLM consists of removing words from sentences, and BERT tries to predict them, giving possibilities and assigning probabilities to them. This method helps to create relations between the words (Devlin et al., 2019).

This variation of the transformer architecture is based on a bidirectional attention flow (Seo et al., 2017). where the computed embedding analyzes the text before the token and the text after the token to better understand the context. BERT is also trained on large text chunks and fine-tuned for specific tasks.

BLOOM Open-Access

BigScience Large Open-science Open-access Multilingual Language Model (BLOOM) originated from a big collaboration of hundreds of researchers, with the objective of providing open-access to large language models.

The Bloom architecture (Figure 5) is based on the original Transformer architecture (encoder-decoder architecture), but some changes were proposed (Scao et al., 2022), like alternative positional embeddings and novel activation functions.

After a series of experiments two architectural deviations were adopted for BLOOM. The Alibi Positional Embedding, which showed good results, not only with is capability of extrapolating to longer sequences, but also good results on training and downstream performance. The Embedding LayerNorm which improved training stability but ended up penalizing the zero-shot generalization.

On top of the architectural updates, the BLOOM project explored a training dataset where many natural and programming languages were used to turn it in a more adaptable tool. To ensure data privacy, BLOOM method involves changing the dataset, removing personal identifiable information.

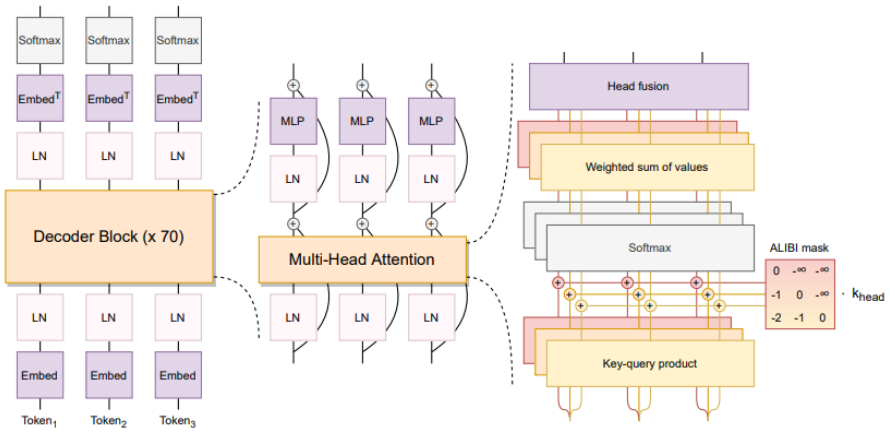

Figure 5 BLOOM full architecture representation (Scao et al., 2022).

Large Language Model Meta AI (LLaMa)

LLaMa is a model developed by Meta, formerly known as Facebook. They are committed to using an open approach to AI and have established partnership with many organizations to guarantee the future for open AI.

The (Touvron et al., 2023) introduces the LLaMa model, proposing a training method based exclusively on open-access data, marking a big step compared to other models dependent on private and inaccessible data.

The model LLaMa 2 is an auto-regressive language model using a transformer architecture derivation. The 70 billion parameter uses grouped-query attention to speed up the process of prediction, without sacrificing quality. These LLaMa 2 models are available free of charge for commercial use, being available for anyone that agrees with the LLaMa 2 license agreement.

Other Transformer derivations

Since the BERT breakthrough, many Transformer-based models surged and comparing these models can be a challenging task due to their architecture differences and training strategies that most of the time focus on very specific tasks. The (Pearce et al., 2021) study, used Transformers as a starting point and talked about some derivations.

RoBERTa training method was compared to BERT and the key modifications implemented were that RoBERTa is trained longer and on a larger dataset, changing the

masking pattern applied to the dataset and removing next sentence prediction. This model reached a very high rate of effectiveness on multitasking datasets, establishing state of the art results, it also matched top results on other datasets.

A Lite BERT (ALBERT) is a more compact and lighter version of BERT, made for users with less time or not as good computational capacity available. This model has less parameters than the original BERT, a task accomplished through cross-layer parameter sharing. This version is now very popular for corporation usage, because of its lower time and lower costs associated with training, while maintaining a similar accuracy.

XLNet comes to fix a few problems. With BERT, the dependency between masked positions is ignored, additionally to that, BERT may achieve worse results related to its fine-tunning strategies. XLNet uses permutation-based training objectives, which allows it to learn the relative positions of words in a sentence (H. Li et al., 2020). It also uses a new method called "memorizing transformer" to improve performance.

Some other well-known methods include Efficiently Learning an Encoder That Classifies Token Replacements Accurately (ELECTRA), ConvBERT, DistilBERT, BART, DeBERTa, and many others that are yet to come.

Benchmark strategies

LLMs can act in unpredictable ways and assessing their performance can be a very complex and uncertain task. For this problem, some ways of testing LLMs are explored in this section.

Hugging Face ranking Methodology. The community around LLM development uses a platform called Hugging Face to collaborate on models and datasets. On this platform there are beginner tutorials and task guides to help newcomers, there is also a platform library available to use.

Hugging Face runs a frequent updated leaderboard system that envisions ranking the most popular/capable chatbots, the ranking system is based on three benchmarks:

- A public voting system where two anonymous chatbots are presented side-by-side and the user has the opportunity to chat with both bots to decide which bot to vote. This voting system is then used in conjunction with the MT Bench and the MMLU method to create a leaderboard.

- The MT Bench is a method proposed by (Zheng et al., 2023), it involves using a LLM as a judge, basically evaluating the responses on open-ended questions. The way this method works is by preparing a sequence of queries designed to evaluate the conversational and instruction-following capability of the models.

This method has proved to achieve good results with the human votes discussed in the previous benchmark, but it also addresses some limitations. The LLM judge may show some bias related to a side when comparing A side with B side. Another issue is that the judge could lean towards a longer response even if it is not as clear or accurate. In some cases, the judge may also favor the responses generated by themselves and show limited math and reasoning capability.

- The third benchmark method used is the Massive Multitasking Language Understanding (MMLU) explored in the (Hendrycks et al., 2021) paper. This method measures the level of knowledge absorbed by the models during pre-training, trying to replicate the way humans are evaluated.

For the final testing, a multitasking and multidisciplinary test with multiple-choice is applied to the models.

General Language Understanding Evaluation (GLUE). The General Language Understanding Evaluation method is proposed in (Wang et al., 2018), GLUE is a platform designed to benchmark the performance of LLM using a variety of Natural Language Understanding (NLU) tasks.

The GLUE method includes a training dataset to ensure the understanding of certain linguistic phenomena. Some tasks are evaluated with a large dataset for training data, while other tasks are evaluated in an environment where the model is provided with limited and mismatched test data. This approach encourages the models to use simpler sample-efficient learning.

The GLUE method groundwork was used as a foundation to create SuperGLUE (Wang et al., 2019), this new method introduced an adaptation for GLUE, expanding for new languages, and using a selected number of tasks proposed on an open call, expanding the task format previously used.

General Ai Assistant benchmark (GAIA). The GAIA approach uses a method as simple as possible. It uses queries that are expected to have straightforward answers, this way, the evaluation can focus on the root tasks that the model should be able to do with high precision, like adapting fast to certain types of requests and understanding the focus of the conversation (Mialon et al., 2023).

The second focus of GAIA benchmark is the clarity of thought process, so with the simple answer that is expected, an additional reference or origin of the information is given. This way, it is easier to trace the thought process and analyze how the information was received and transformed into the output.

In some benchmarks is remarkable the possible opening for cheating, like coming up with a correct answer for the question but using the wrong method or even completing tasks in the wrong way, like brute-forcing. So, GAIA implemented tasks constructed to be more complex and needing the completion of many steps to ensure that the evaluation occurs as expected, analyzing the thought process of the model.

Dataset Generating

With the necessity of training new models, the idea of using LLM to create new datasets arises. This idea is explored by (Z. Li et al., 2023). The results were not as good as expected in this case, and the reasons why may be related to the generation of open text, which may also include misleading information. However, this idea should not be discarded, for this case scenario, where we need to generate questions from the fed information, it may not be as good or effective as expected, but LLM has a big potential in creating information based on other information, and therefore, the possibility of using literature is open. Also, the generated questions will be used for model evaluation. Therefore, the questions can also be classified as irrelevant questions (if the case) and can be ignored when classifying the final results. In the future, some strategies that involve summarizing information on schematized data could be explored and used in adaptive learning to create new models capable of performing specific tasks.

Data Grounding

An LLM model must be adapted to retain domain-specific knowledge, and in this case specifically, to develop an agronomic assistant. The approach envisioned involves choosing a versatile model that can be adapted to the system architecture. This choice is based on various

factors, like the possible pricing, the possibility of local hosting, and the security of data used in the model, which guarantees user privacy.

To complement a good LLM, a strategy for data grounding must be defined. Data grounding is the process of basing the output of the LLM on context/information. This process can be divided into two approaches, Static Data Grounding and Dynamic Data Grounding.

Static Data Grounding techniques involve predefined knowledge that requires pre-training of the model on a curated and vast dataset. **Dynamic Data Grounding** on the other hand allows for a flexible base of knowledge achieved by separating the Model and the Data. This separation needs to be complemented by some form of information-storing and fetching system, such as Vector Store or distributed Locality-Sensitive Hashing.

Fine-tuning. The LLM is pre-trained on an extensive dataset, acquiring general world knowledge and developing speech and language capabilities. The process of finetuning comes in a second phase and aims to specialize the model on task-specific or domain-specific knowledge. This technique is proposed by (Howard & Ruder, 2018).

Many variations of the finetuning methodology have been proposed, some alter the input, some focus on particular tasks, and others aim to simply change the pre-trained model weights. As (Hou et al., 2022) mentions that the method used for these specific tasks training is usually based on fine-tuning, but the paper explores an alternative. This method envisions a more efficient use of data, separating models in general and adapted pre-trained language models, adding task-adaptive reparameterization (TARP) and task-adaptive model structures (TAMS), components that individually improve the model system.

With the continuous increase of model size and need for power, other scaling finetuning techniques could be implemented, such as prompt tuning, explored by (Lester et al., 2021), which prepares a group of prompts, each one specific to a task, and directs the input to a relevant prompt.

Retrieval-Augmented Generation (RAG). The RAG (Retrieval-Augmented Generation) creates a system where two main components work together, the retriever and the generator. This way, we have a much more versatile base of knowledge and separate it from the LLM. It processes data and creates a ranking system where similar content is ranked together, maintaining a correlation. The content is compressed into vector embeddings, representing the data in a compressed, high-dimensional space. These embeddings can be achieved by encoding the

base information with models like Dense Passage Retrieval (DPR), as explained by (Lewis et al., 2020).

Many models and methods exist to create these vector databases that are crucial to storing, indexing, and retrieving the generated embeddings.

- Store and Retrieval

For the Vector Store component, many options are available for use. This is where tools like FAISS, ChromaDB, Annoy, Pinecone, Milvus and ElasticSearch come into play, some providing advantages to light and local systems, others proving more effective for scalable and cloud-based systems, and others proving better at dynamic RAG workflows.

- Dense Passage Retrieval (DPR)

To support the embedding and retrieval tasks of the Store component, various specialized models can be employed. General-purpose embedding models include "nomic-embed-text-v1.f16", available on GPT4AllEmbeddings and mpnet-base-v2 available on Hugging Face Transformers. In addition, DPR-BER, a dual-encoder architecture specifically designed for open-domain question answering and retrieval (Lewis et al., 2020). Unlike general embedding models, DPR jointly learns query and passage representations to improve semantic matching.

Large-Scale Alignment. In the context of chatbots, there is a novel methodology envisioned to overcome the difficulties created by the scalability, creating a different but more efficient way to train the models. The LAB (Large-scale Alignment for chatBots) (Sudalairaj et al., 2024) methodology explores the applications of taxonomy-guided data generation without relying on human or LLM, combining it with a multi-phase training framework.

The taxonomy component classifies the data to organize different samples into task groups, like knowledge, scientific skills and compositional skills and smaller groups inside. This separation is then used to enhance each group and applied in the second phase.

RAG vs Fine-tuning. Fine-tuning achieves highly precise and concise outputs, demonstrating a strong capacity to learn specific information. However, this method is costly due to its extensive training process. On the other hand, RAG offers an alternative approach. The RAG method can significantly accelerate the overall process by leveraging information retrieval, thereby bypassing the initial heavy training step required in fine-tuning. However, RAG may produce outputs that are more complex and challenging to manipulate (Balaguer et al., 2024). The comparison of the two alternatives is described in Table 1.

Table 1 - RAG and Fine-Tuning comparison.

Aspect	Fine-Tuning	RAG
Cost	High-cost due to extensive training process	Lower cost by leveraging existing data
Manipulation	Outputs are easier to manipulate	Creates complex outputs, hard to manipulate
Precision and Succinctness	Achieves highly precise and succinct outputs	Outputs may be more complex
Learning Specific Info	Good capacity to learn specific information	Relies on information retrieval
Training Speed	Slower training time	Faster due to skipping training step
Answer Speed	Faster answer generation	Slower answer generation due to information gathering

While fine-tuning stands out in precision and learning specific details, RAG offers faster implementation and cost efficiency, although with potential complexity in outputs. The choice between these methods depends on specific thesis requirements and trade-offs between precision, complexity, and cost.

3

MODELING

The evolution of new AI technologies and applications has been growing at a fast pace. This evolution leads to a new necessity of adaptation to the world to take advantage of these new capabilities. This means that not only do people have to show some flexibility to create space where it can be applied/tested, but also, the technology has to be adapted to the current world to start gaining credibility.

With all this in mind, the proposed architecture embraces a comprehensive design where all the components work together so that the final model can be integrated into diverse environments and adapted to different uses, maintaining the capability of processing modern dimensions of information.

This chapter outlines the characteristics of the RAG system in question, explaining every component and how they interact, keeping an evolutionary view of the concept envisioned, and addressing the principal effects of the changes on the final product.

3.1 Approach evolution

In the first stages of formulating the conceptual architecture for the envisioned system, the chosen method of data grounding was the fine-tuning approach. This process begins with the selection of a suitable pre-trained model. For the initial envisioned approach, a supervised training method was chosen. As represented in Figure 6, a domain-specific dataset with a question/answer format was prepared, consisting of examples that align with the agronomical use case. This approach is characterized by a heavy process of reiteration, which consists of adjusting the model parameters using human feedback. This iteration process is what mainly differentiates the fine-tuning system and the RAG system.

This iterative process is called Reinforcement Learning with Human Feedback (RLHF). While traditional fine-tuning adjusts the model based on static datasets and predetermined outputs, RLHF hypothetically introduces a more dynamic and adaptive learning process. The way this process works is that human agronomy experts provide qualitative feedback by ranking or scoring multiple generated responses. This human feedback is then used to train a reward model, which guides the reinforcement learning phase by encouraging the system to produce outputs that align more closely with human preferences and domain-specific standards.

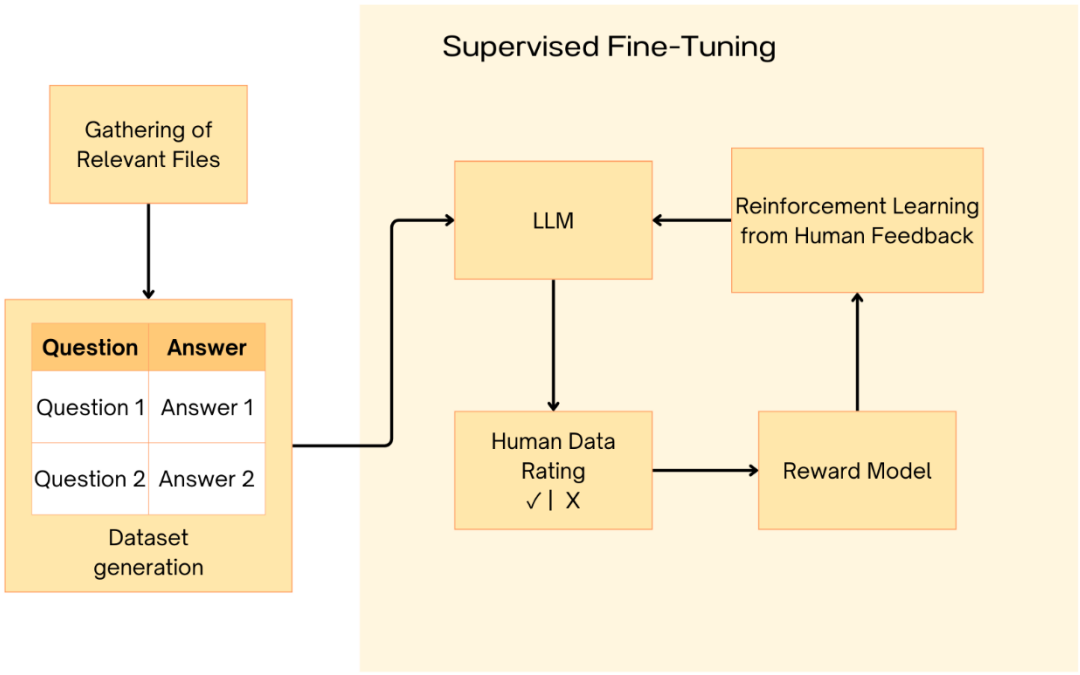

Figure 6 Fine-tuning training process overview.

The research for this initial approach proves to be relevant for a better understanding of the advantages and disadvantages of the RAG approach. On top of that, the dataset generation method can be reused for a validation phase of the system being developed, such as a form of generating example questions based on the files that are fed to the Vector Database of the RAG system. These generated questions can then be used as user input examples to analyze the capabilities of the final system.

As more research was done, some risks of the fine-tuning approach became too significant to ignore, such as the need for frequent retraining and difficulty in integrating new information dynamically. With this in mind, the architecture transitioned to a RAG-based system, which can be used as a more versatile approach, as it allows for the update of the

knowledge base without the need to fully retrain the model, while also presenting a better cost-effective solution, as previously discussed.

3.2 Conceptual Architecture

This section explains in depth how the components work, providing an insightful vision of how they are prepared before integration. This way, we create a vision of complete architecture and all the components that contribute to generating an enriched answer. The focus will be on the vector database and the LLM, which represent the principal processes.

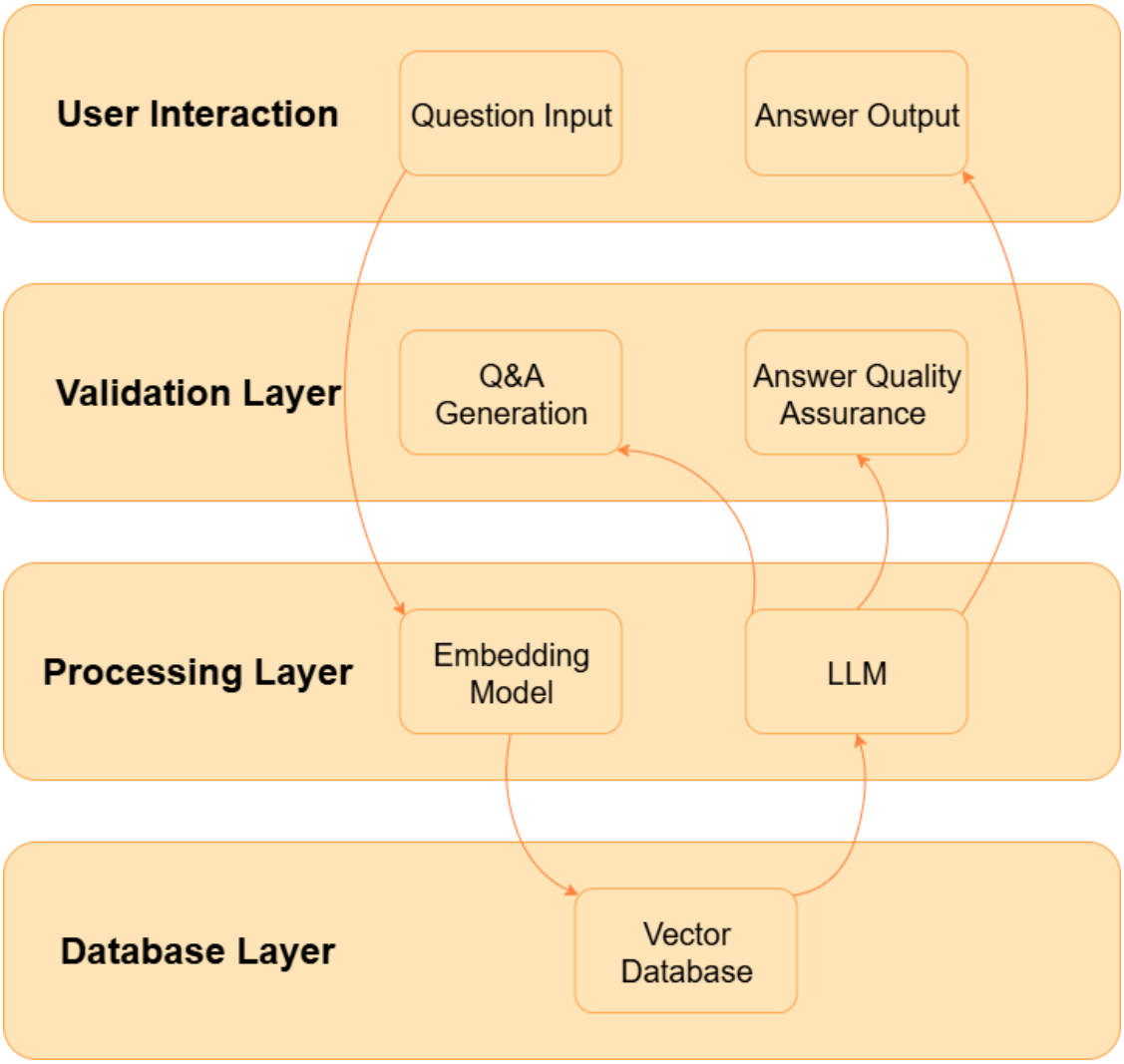

Figure 7 Retrieval-Augmented Generation High-Level Process Overview.

The main idea in RAG, as shown in Figure 7, is that there are two main components, one in charge of storing and fetching the data, and the other in charge of processing all the relevant data and creating an answer with it, maintaining a conversation with the user.

Therefore, in Figure 7, it is possible to see a High-level representation of the proposed RAG process. With this representation, we can see that the model bases the answer on relevant information and how it flows until it reaches the user

3.2.1 Vector Database

The Vector Database system is already integrated into the world of technology and business, having a variety of uses associated with similarity search. It can be seen in similar recommendation systems, such as social media, where the connections and content presented are chosen based on the activity and interests of the user, streaming platforms where the shows or songs recommended to the user are similar to previous interactions, or even in other users' similar tastes.

In this thesis, the vector database will work as an information database where the user information is stored and ready to be retrieved. The primary role of the vector database is to extract data based on its similarity to the user's query. This process creates a context of extended information, making it possible for the LLM to generate an insightful and relevant answer.

The vector database organizes and stores the data using vectors as a multidimensional mathematical representation. These representations are generated from raw data sources, which, in this specific case, consist of valuable PDFs and text documents containing useful information. The transformation of textual data into vector embeddings is a crucial step, as it ensures that important features and characteristics of the original content are retained. This process is illustrated in Figure 8.

This process allows the database to maintain meaningful relationships and structural similarities between different data points. By preserving these connections, the vector database enables efficient retrieval, comparison, and analysis of information, making it a powerful tool.

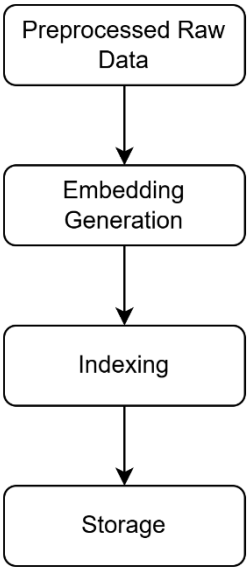

Figure 8 Information storage process in the Vector Database.

Figure 8 supports the understanding of the steps involved in creating the whole process before the data is stored. The data is commonly broken into smaller chunks, using separators (".",".\n", "\n\n", ...). This technique has two objectives, on the one hand, it separates different contexts and not related data, by separating different phrases and paragraphs, but on the other hand, it keeps related data together by using preassigned separators, and not braking in the middle of sentences. It starts from the point where the data is already extracted from its original format and is ready to be stored in the Vector Database.

Once the data is chunked, the process starts from the point where the data is already extracted from its original format and is ready to be transformed. In the Embedding Generation (Vectorization) step, a chosen pre-trained model is used to convert the selected data into the vector embeddings, (format ex. [0.30, -0.13, ..., -0.67]). These vector embeddings are a good and efficient way of storing the data, but we still need to store them with an organized technique.

Here is where the Indexing step starts, the technique used in this step involves creating a structure that allows for fast access to the data based on similarity measures. A relevant technique to mention is HNSW (Hierarchical Navigable Small World). This technique constructs an advanced multilayered graph that reconstructs itself when new data is added. This way, the top layers contain less data, creating a simple and faster path to the desired information when making queries.

After the indexing phase, comes the final storage, where the vector embeddings are stored, maintaining the Universal Unique Identifier (UUID) that associates the vector with its respective actual information.

The Vector Database is the core component that differentiates the RAG approach. Unlike a fine-tuning approach, which requires updating model weights through iterative training on new data, a vector database enables real-time adaptability. This approach also allows dynamic access to relevant data at inference time and without requiring expensive and time-consuming retraining. Consequently, the RAG system can easily integrate new knowledge, proving to be advantageous in domains where information is in constant evolution.

3.2.2 Large Language Model (LLM)

Large Language Models are being integrated as a whole new way of processing data, making it possible to consume large amounts of information and use it when the context is relevant. These models present big capacities in text generation/analysis tasks, revolutionizing the AI world implemented in chatbots to serve as personal assistants.

For this thesis, the capabilities that the LLMs have are valuable. In the context of a chatbot based on an RAG system, the LLM has the main function of transforming the gathered data into conversational text that answers the question asked and maintains the flow of the conversation, understanding the context and helping the user. It works basically as a bridge between the information and the user.

There are two key points where the application of the model proves to be essential. First, the model efficiently processes the prompt along with the user query and the retrieved context. This involves an understanding of both the task proposed and the information gathered. Then, LLM proves its capabilities to summarize information, creating a well-formed answer to the question with all the information available.

The effectiveness of the LLM in this task relies not only on its capabilities, but also on the input it receives. The model requires structured and clear guidance, with simple and direct instructions, this is where the prompt comes into play. As illustrated in Figure 9, the prompt consists of an initial task description characterized by separate guidelines to follow. It also includes a reserved space where the user's question and the context are expected.

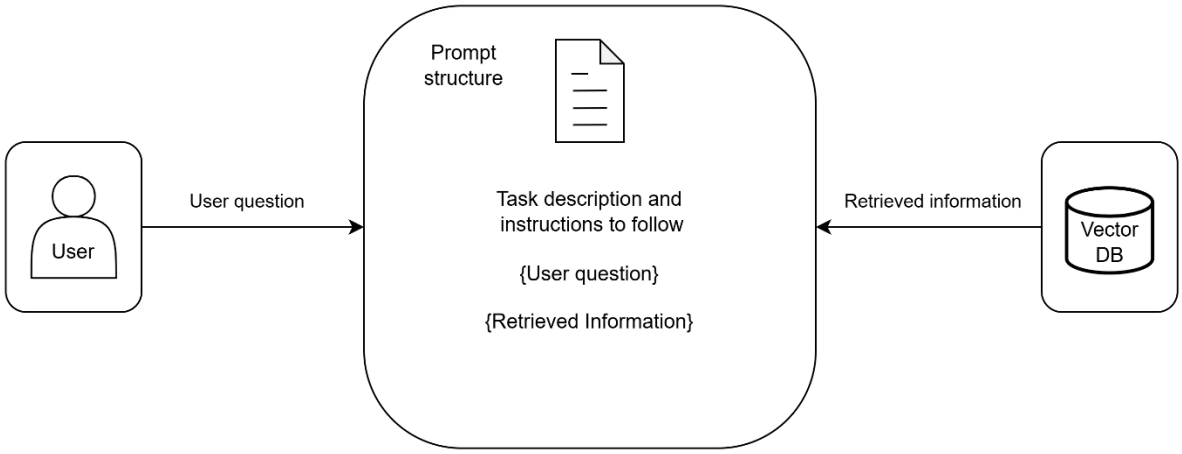

Figure 9 Prompt content.

By structuring the prompt this way, we equip the LLM with all the necessary details to process the input effectively and generate a well-informed response. The LLM's effectiveness relies not only on its capabilities but also on the way that its task is explained and how relevant the context gathered is.

3.2.3 Validation Component

The validation component is a non-essential but highly valuable part of the system. This means that, while the system can operate without it, the validation component serves as a quality-check mechanism, assessing the relevance and reliability of both the retrieved context and the generated answer. In the final implementation, the validation mechanism could be employed either actively during the generation process or as a post-user interaction validation tool. If utilized for post-interaction purposes, the validation component would serve as a means of evaluating the output after the user has received the response, helping ensure the accuracy and appropriateness of the information provided, but in a way that doesn't affect the generation time.

The validation component operates at two key points in the answer generation pipeline. The first stage occurs after retrieving context from the vector database. At this point, the system has identified and extracted relevant documents or knowledge snippets based on the user's query. The validation mechanism then assesses the retrieved information, analyzing its relevance to the question asked. Its function is to provide an evaluation of the quality of the gathered data before it is passed to LLM.

The second stage of validation takes place after the language model has generated a response. Here, the validation component examines the final output, comparing it to both the

original query and the retrieved context. The purpose of this evaluation is to determine how well the generated response aligns with the provided information and whether it stays within the boundaries of factual correctness. This way we can also keep track of the origin of the content that is used.

The information provided by the validation component is used to recursively update model variables, like its temperature (which affects the generation creativity) or even the database content and how it is stored/retrieved, like changing the size of the chunks that are stored in the database or changing the number of chunks retrieved when generating an answer.

3.3 Functional Requirements

To properly develop a chatbot that serves as an assistant providing tailored advice and actionable recommendations to farmers and agronomists, it is important to specify the main functional requirements for such a system. When designing the pipeline, the primary goal is to create a model that can be easily adaptable to new technologies by changing some steps while keeping the main architecture intact. With this objective in mind, the functional requirements are established to allow for flexibility in the architecture, ensuring that the system can evolve while still achieving the initial set of goals. Therefore, to develop an effective chatbot system

for providing tailored advice and actionable recommendations to farmers and agronomists, the functional requirements gathered in Table 2 should be considered:

Table 2 - Defined Functional Requirements.

Functional Requirement No.	Functional Requirements Description
FR 1	**Data Ingestion and Processing**
FR 1.1	Text data extraction
FR 1.2	Text chunking and embedding
FR 2	**Vector Database Management**
FR 2.1	Storage of Embeddings
FR 2.2	Similarity Search
FR 2.3	Context injection
FR 3	**LLM Interaction**
FR 3.1	Full understanding of the task proposed and questions asked
FR 3.2	Relevant and accurate response generation
FR 4	**System Characteristics**
FR 4.1	Application versatility
FR 4.2	Run the system locally

The functional requirements were divided so that each category aims to create objectives in different modeling phases. The FR1 category highlights the process of preparing the data for the system. As someone is in charge of gathering the data for the system, the system must be able to receive that raw data and process it. The system must be capable of extracting text from PDFs (FR1.1). Once the text is extracted, it must be processed into manageable pieces. This involves text chunking, where large documents are divided into smaller segments, separating different information to maintain retrieval accuracy. These chunks are then converted into embeddings (FR 1.2), in a format ready to be analyzed and stored by the vector database.

In the Vector database management category, we discuss some important capabilities that the system must have to create accurate answers. The system must store these embeddings in a specialized vector database (FR 2.1), allowing quick lookups based on relevance rather than simple keyword matches. When a user submits a query, the system performs a similarity search (FR 2.2), identifying the most contextually relevant chunks from the stored database. After retrieving the most useful pieces of information, the system injects this context together with the query into the prompt before passing it to the language model. This context injection process (FR 2.3) ensures that responses are well-grounded in the available data, improving both accuracy and informativeness.

The LLM interaction category (FR 3) imposes measures to ensure that the system correctly interprets and responds to user queries. To generate meaningful responses, the system must fully understand the task or question submitted by the user (FR 3.1). This requires advanced natural language processing capabilities that can comprehend intent, context, and subtleties in phrasing. This requirement forces a comparison between multiple LLMs and an analysis of the chosen LLM to ensure that the model meets the expectations. An equally important factor in ensuring high-quality responses lies within the quality of the proposed prompt. How a query is formulated significantly impacts the model's ability to generate relevant and accurate answers. A well-structured prompt should provide clear instructions for the task and clear guidelines, following an adequate format for the model. Once the intent is established, the system should produce relevant and accurate responses (FR 3.2) based on the retrieved data, minimizing hallucinations and ensuring factual correctness. This requirement is also affected by the quantity of the information used to generate an answer, forcing the right amount of information to be determined, this limit being the maximum amount of information that can be fed to the LLM before it starts hallucinating or taking too much time to generate an answer.

Finally, the System characteristics (FR 4) define the usability and flexibility of the RAG system. It should be versatile (FR 4.1), meaning it can be applied across different domains, and not limited to an agronomical environment, maintaining the possibility of the adaptation of the system for other uses that involve a RAG system. Moreover, to enhance privacy and accessibility, the system should have the capability to run locally (FR 4.2), allowing the system to operate in a controlled environment. This ensures that sensitive data remains secure while still leveraging the power of retrieval-augmented generation, being useful for the user and the company's data.

By meeting these functional requirements, the chatbot system is suited to assist the agricultural community effectively, offering valuable insights and recommendations that are trained in the latest research and tailored to individual needs.

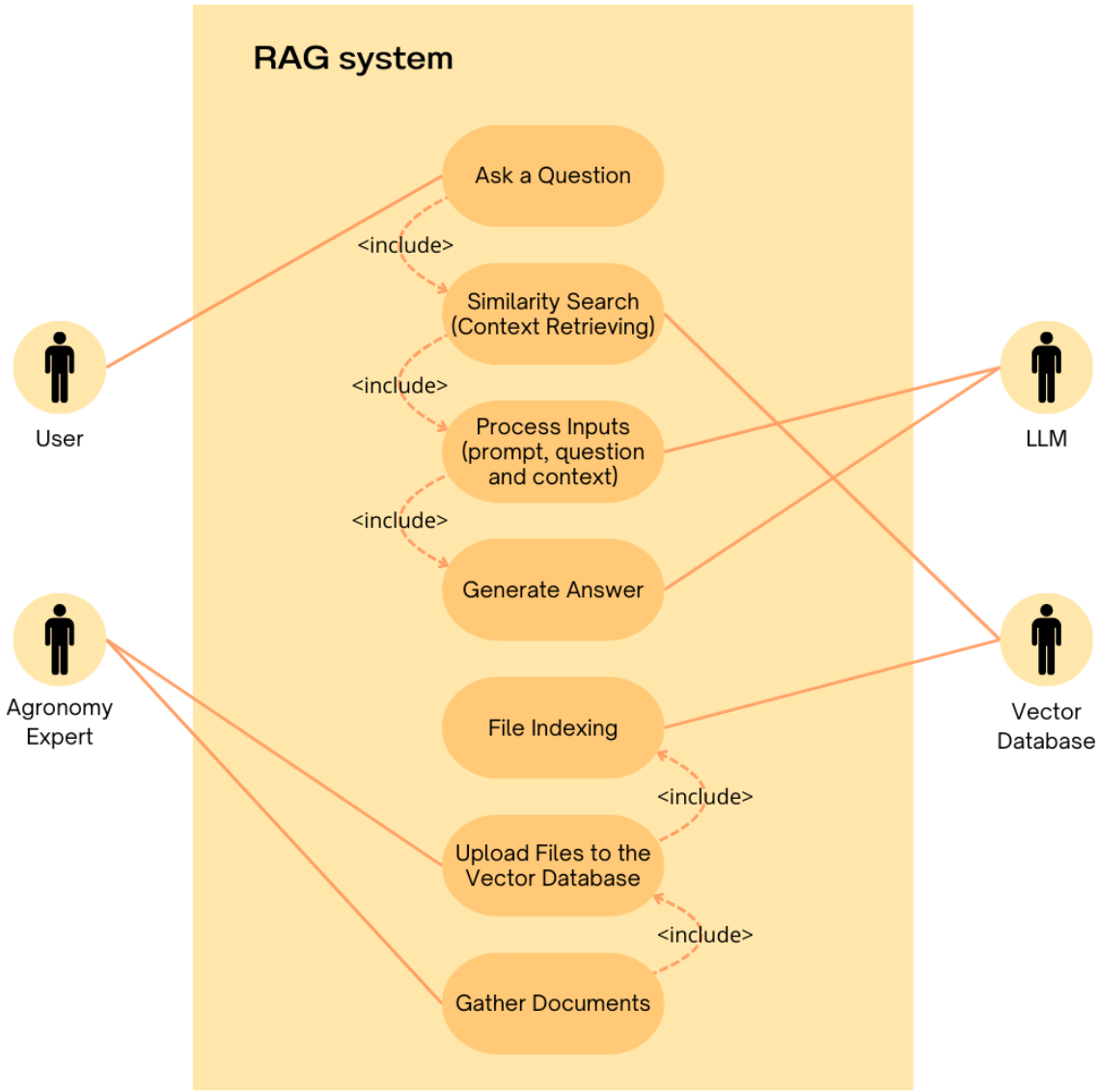

Figure 10 Use case diagram of the designed RAG system.

Figure 10 represents the main functionalities of the actors in the designed RAG system. The actors highlighted in this system are the User, the LLM, the Vector Database, and an agronomy expert. At the top of the diagram is the User, who initiates the process by asking a question. This user interaction triggers the entire system, requiring relevant context retrieval and answer generation. LLM takes on the core responsibility of processing the user's question, combining it with any retrieved contextual data, and generating an appropriate answer. This retrieved contextual data is based on external knowledge stored in the Vector Database.

The Vector Database is in charge of two main functions, it performs similarity searches, locating the most relevant information that matches the user's query, and handles the indexing of these files, so that they are easy to retrieve when needed.

An additional actor is highlighted in Figure 10, the agronomy expert. This actor is responsible for gathering high-quality domain-specific documents. These documents form the foundational knowledge base that fuels the RAG system. Once collected, the expert uploads these files to the Vector Database, where they are indexed and made available for retrieval.

The system was designed with the main idea that the worlds of AI and agriculture are always evolving. That's the reason why it is so important to maintain this easy-to-access information in the system. With this characteristic in mind, an additional actor could be represented in the image. This actor would be an AI/LLM expert who would be in charge of choosing and changing the LLM when needed or relevant. Like the agronomy expert, this actor analyzes the possibilities and chooses what is needed for the system, but in this case, the expert is in charge of keeping the technology up to date instead of the base knowledge.

3.4 Information Pipeline

The previous section presented an overview of the system, representing the main components needed and their functions to help understand how the information flows. This section builds on that previously established view by providing a more detailed examination of the information flow. While the previous discussion focused on the interactions between system components and their individual functions, this section breaks down the pipelines step-by-step, starting with the user input until the answer is finally generated.

The validation component will be explained in detail further in the document, while the focus will remain on the practical final user pipeline. For the system's final usage, the vector database has already processed and stored all the information needed to function properly.

As represented in Figure 11, the question needs to be processed into a vector embedding format before searching for relevant information in the vector database. With the question ready, the vector database finds similar information stored and returns the relevant context to generate the answer. This gathered context, now in text format, is fed into the LLM together with the question and the prompt. The LLM then generates an answer and sends it back to the user who made the question.

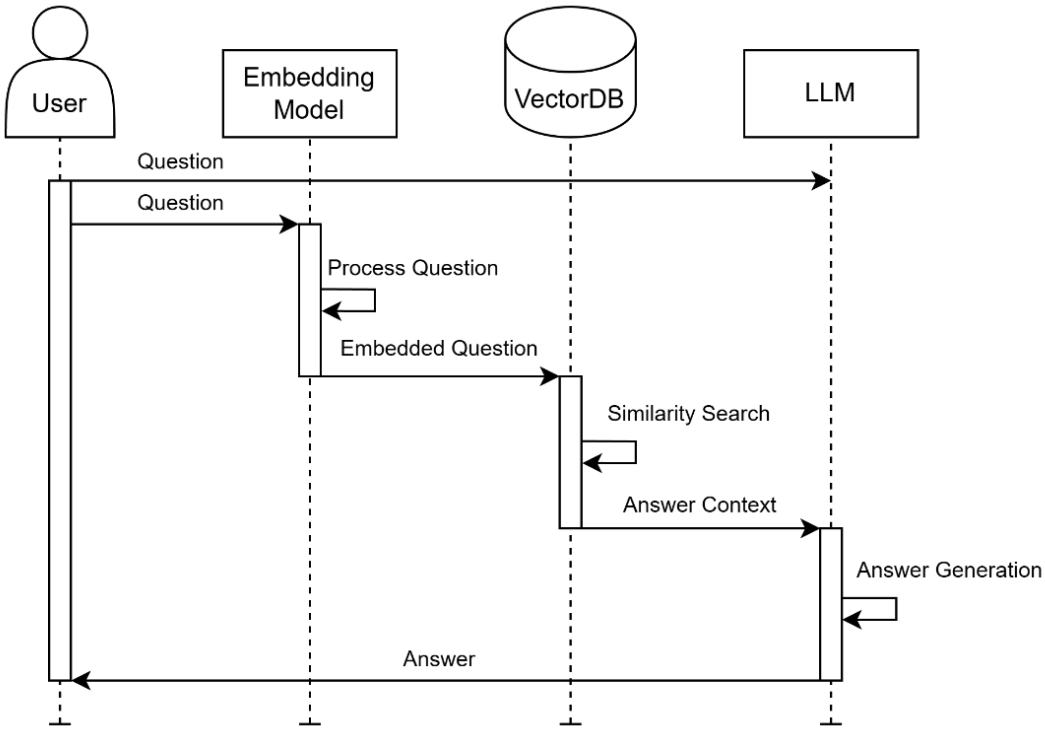

Figure 11 Operational Sequence Diagram of the RAG Pipeline.

4

IMPLEMENTATION

This chapter provides a comprehensive overview of the implementation phase of the system that was modeled and conceptualized in the previous chapter. It discusses the specific decisions made during the implementation. The chapter elaborates on the selection of the main framework, the models for embedding and generation, and the tools used for the previously discussed technologies. In the discussion of these decisions, the advantages of the tools selected are highlighted, while presenting other options and why they were considered less appealing for the context of the thesis.

4.1 Framework

For the chosen framework, many factors were considered, as the actual environment that involves the thesis development could affect its direction. The scope was to develop a thesis that could keep a wide range of options throughout its evolution. On top of that, the framework had to be helpful in the prototyping step, with valuable tools to experiment with, but, if possible, also being reliable in the production step, so that the research done could be used and put into effect later.

Like many other technologies, it is always helpful to have an active community to provide support. This is where LangChain (LangChain, n.d.) comes in handy, with the new boom on AI and LLM research, this framework became indispensable. It already has a good range of features and the team behind it keeps on adding new features, keeping it up-to-date, and providing more compatibility with new tools.

At the beginning of the implementation, one other option was also considered, this framework was the HuggingFace library (Hugging Face, n.d.), the main advantage of this library

it’s the wide range of models to choose from, and it was actually the framework to put into testing at the start of the thesis, because of its beginner-friendly tutorials that help with the introduction to LLM world. However, after some understanding of the technologies available and the comparison with the LangChain toolkit, LangChain was considered more effective with an RAG system implementation. But for further exploration with some LLM testing, the HuggingFace platform contains many features and tutorials for a fast and beginner-friendly LLM introduction.

As previously mentioned, the system is designed to run locally, to maintain the security of the user's and the company’s information, the main factor in the rag system that affects this matter is the processing of data into the vector database and the processing of the question of the user and the context retrieved when generating a new answer. LangChain proved effective in maintaining this condition amongst these processes offering the necessary tools, or the compatibility with external tools when needed.

The LangChain framework proved very reliable during the development of the thesis, this environment was particularly beneficial due to its modular design and clear separation of tools, a very good example of this feature, is the way how the vector databases, it is possible to simply save a vector database into a particular type of file that can be later accessed on another script, this way allowing for a well divided, structured and easy to evolve process of implementation and experimentation. Moreover, the framework provides many more functions for its tools that help with the modification and configuration of previously implemented features.

4.2 Environment

As explained earlier in this thesis document, the whole system is orchestrated by a Python script. This way, the system is revealed to be easy to adapt and reimplement in other environments. Therefore, if the fundamental tools can be used in different environments, the system can be replicated with a few simple steps of installation.

In recent years, cloud services have become indispensable tools for computational tasks, especially when running code for some features of companies' and organizations' products. One of the key advantages of utilizing cloud services lies in the highly controlled and standardized environments they provide. These environments are specifically designed to deliver reliable and consistent performance, ensuring that the system operates as expected across different scenarios. This level of reliability is crucial for businesses that depend on the continuous and predictable execution of their software solutions while providing a wide range

of options that cater to different needs and budgets, going from highly cost-effective solutions suitable for small-scale projects, to cutting-edge, high-performance configurations that leverage the most advanced technology available.

Moreover, cloud services offer built-in features that help with the scalability of systems. As an organization grows or its user base expands, the ability to seamlessly scale resources up or down becomes vital. Cloud platforms support this by enabling dynamic allocation of computational power, storage, and network capabilities, allowing businesses to adapt to changing workloads without significant infrastructural changes or manual intervention. To extend on this matter, these platforms provide features to keep clients' and organizations' private data safe, like communication encryption, access control, and compliance with privacy standards.

The feature of a closed environment is very appealing to a research thesis such as the one on this thesis. The closed environment helps in gathering consistent and reliable results that can be compared, such as running time and input size limit. Not only that, but it is also helpful to overcome limitations such as local machine computational resources.

For these reasons, AWS SageMaker was the chosen platform for the development of this thesis work. It provides a managed environment that aligns with the characteristics discussed, offering an option that aligns with the required computing power while remaining cost-effective. Moreover, this environment helps with the logistics of the validation step, keeping an organized environment with the results of tests conducted, where the different Vector databases are created, contexts retrieved, and answers generated are easy to access and manipulate.

The developed work in this thesis was carried out under the guidance and support of Hidrosoph, whose previous experience with AWS Sagemaker ensured an effective integration of the platform’s capabilities.

4.3 Tools and Technologies

This section provides a holistic description of the used technologies, comparing similar tools and their alternatives. The purpose is not only to provide a guide on all the used technologies, but also to consolidate the choices made during the thesis development and explain the reasons behind each choice.

The structure of the section follows the logical order of the data pipeline, detailing each technology or tool as it is introduced and integrated into the system's workflow. This approach

maintains a clear view of the global system while explaining the use of the tools, where they apply, and how they interact. Figure 12 illustrates the system architecture, with the tools used for each technology.

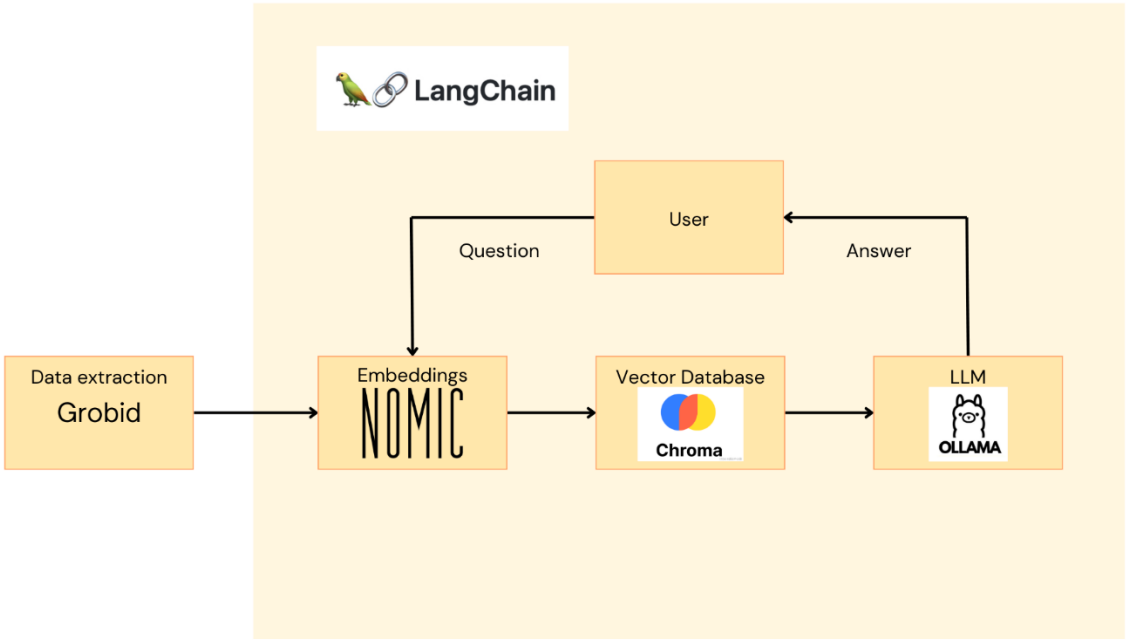

Figure 12 System Architecture with Tools Used.

4.3.1 Data extraction

As explained earlier in section 3.3 the information that serves as the knowledge database is gathered by an agronomy expert. This information is gathered in the format of PDFs but this format is not suitable for the vector database to store. For that purpose, the GROBID (*Gobrid*, n.d.) (GeneRation Of Bibliographic Data) was used. This tool extracts data with machine learning techniques from raw PDF documents, particularly scientific publications. GROBID is the main tool for separating important technical information from the documents.

GROBID works as a local web app where the settings are changed according to preference, like maintaining authors in the XML output file. This information can be used to create meta documents with the author when feeding the vector database, although this feature was not explored in this work. The output XML file is filtered to extract the body of the document, where all the text information of the document is located.

This GROBID tool was the most reliable form of PDF data extraction explored, but new technologies that are surging are worth mentioning at this stage, and that could turn out to be useful in the future.

The rise of AI has influenced the development of many technologies and revolutionized the capabilities of old processes. This data extraction from PDFs is no exception. In this context of processing PDFs, a new technique that analyzes the images present in the files has become popular. This process is a relevant way of not losing any relevant information and describing the images present in the file. While this technique was not reliable enough when it was explored and researched for implementation, it may have evolved and can be a good alternative to the GROBID tool, in the near future.

4.3.2 Embeddings and Vector Database

With the data extracted from the PDFs, it can be stored. Nevertheless, before storing the data, as explained before in the section 3.2.1, the data has to be simplified into an embedding format. For this process, we need an embedding model. The choice of an embedding model can be influenced by many factors. Once again, a principal factor to have in mind is the capacity to fully run the model locally. This condition led to the company Nomic AI. This company is an open-source AI organization focused on AI Data with the Nomic Atlas platform and on local AI applications with the creation of some models. This second ramification is relevant to the thesis.

Nomic AI developed the GPT4ALL platform and a Python library adaptation. This GPT4ALL tool is one of the main resources available to the public to run LLMs locally. It works as an API that provides the models to the users, and upon request, the models get downloaded to the requester's machine, and there it can be used as needed.

For further usability, the LangChain framework has created an environment where local models, like the ones provided by GPT4ALL, are supported in the many workflows available. With this option available the only thing left is to choose the model for our needs. In this case, an efficient and lightweight model was the obvious choice, not only because of the limitation of resources available but also because the nature of this thesis leads to an iteration of the processes where the embedding of documents could become very extensive. As such, the chosen model was the "nomic-embed-text-v1.f16" an open-source light model of the family of the nomic-embed models created by Nomic AI.

After discussing the embedding model, the next component in the workflow would be the vector database. In the context of the similarity search integrated with LangChain framework, two main options were explored, the Chroma DB, and the FAISS (Johnson et al., 2021) (Facebook AI Similarity Search).

The FAISS vector database is a tool developed by Meta team. On its own, this tool may prove to be highly efficient for the task at hand, however, its integration into the LangChain framework is less complete and requires a more low-level handling, with certain features needing to be managed separately. The FAISS also indicated the need for more computational power, which also contributed to the final decision.

In comparison, the ChromaDB tool presented better characteristics for the developments, meeting the computational power needs and, at the time of implementation, it had been deeply integrated with LangChain, providing the tools needed not only to manage the text input into the documents but also filter with metadata, a feature that could be useful with the scaling of the developments. Furthermore, ChromaDB provides built-in persistence, a feature that turned out to be very useful for the testing phase, which helped to apply different settings and prompts of the LLM to the same Vector Database, not to mention that this persist feature is indispensable for the thesis where we want to save the progress in the Vector Database, making it possible to access the knowledge when needed, and updating the information stored if needed.

4.3.3 LLM

The next component to discuss in the pipeline is the LLM. Here the LLM is in charge of understanding the task proposed in the prompt, the question asked by the user, and the context retrieved to enhance the answer. For this matter there are numerous options available in the market, so the first step is to filter them. The primary requirements for the model are to ensure it is safe to use with personal and organizational information and preferably be open source. This initial filter narrows down the options, but several choices remain. In the second filtering round, key characteristics to consider include the support provided by the model developers, community support such as available tutorials and experiments validating the model, and the variations available, which can assist with specific uses like chat-driven LLMs and lighter versions for testing purposes. Given these criteria, the LLMs that best meet the requirements, at the time of implementation, are the META LLaMa model, including the recently launched LLaMa 4, which offers a range of options to choose from.

As already mentioned, this AI market is still in an initial phase, which means now and then a new model arrives and debunks the previous favorite model, we could see this as a problem for our system, but instead we took it as an advantage, making the model completely autonomous and easy to change, this characteristic as already proven to be advantageous, because during the thesis many new versions of the LLaMa have been released, as we started

with the LLaMa 2 (released in 2023), going through LLaMa 3 (released in April 2024), LLaMa 3.1 (released in July 2024), LLaMa 3.2 (released in September 2024), LLaMa3.3 (released in December 2024), and the latest version LLaMa4 (released in April 2025).

To support this volatility of the model, a very helpful tool was explored, the Ollama platform, this platform works as a support tool, being able to run the model locally and saving it in a personal machine and even in the personal environment created in the AWS sagemaker. The platform runs parallel to the system, and when needed, the platform provides the system with the model, this process is also orchestrated by the LangChain framework, once again proving to be an indispensable tool for this thesis. This Ollama tool is very helpful for LLaMa models but can also provide other models, like DeepSeek, Mistral, and Gemma3. These models that surged after the implementation of the thesis with the LLaMa model could prove to achieve better results in this RAG context.

4.4 System Interface

To conclude the implementation chapter, a functional user interface illustrates how end-users are expected to interact with the system. This interface was developed using the Streamlit framework, which enabled a fast and simple implementation with all the previous Python work.

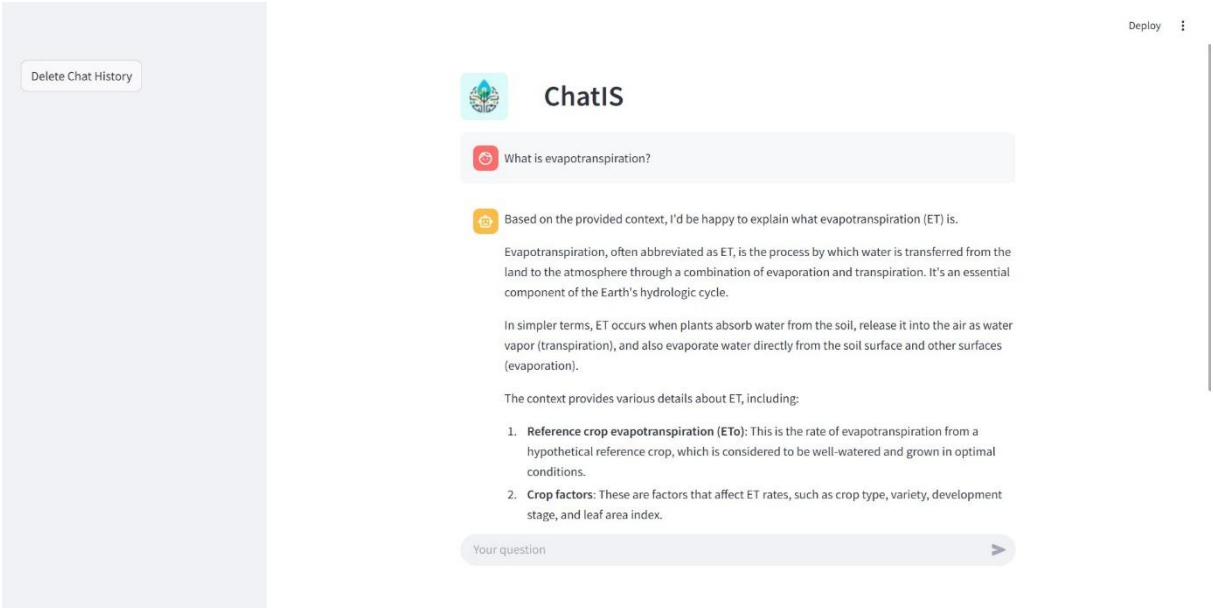

Figure 13 Developed Interface Using Streamlit.

This simple design has the main objective of functioning as an early-stage prototype. As shown in Figure 13, the interface allows the user to submit a question and then receive an answer from the previously developed chatbot. The implemented interface serves as a solid

foundation that can leverage future enhancements, such as the possibility of saving previous chat history and offering the possibility of presenting various models that can be chosen in the user side.

4.4.1 Streamlit

The Streamlit framework was chosen for building the user interface due to its simplicity and easy integration. The framework was designed to create web applications, and unlike traditional web development frameworks, Streamlit allows developers to build fully functional and visually appealing interfaces using only Python.

While implementing and testing the interface, the framework creates an illusion that it is using multi-threaded or parallel functions, but in its core, it works by executing the script from top to bottom each time the application is updated, while saving the state of the page, this way creating a smooth experience for the user.

This line of implementation for the prototype was chosen with the direct thought of creating a prototype. While this implementation may prove efficient within the scope of a prototype, the method may prove to be limited on a larger scale, because all the content in Python script, that is not related to the front end, may also be reloaded.

With all this in mind, the Streamlit framework offers a perfect environment for fast and simple implementation, but with a downside that offers small potential from a scalable point of view, serving primarily for a prototyping purpose.

5

VALIDATION

The validation chapter explains the methodology selected to evaluate the implemented system presented in the previous chapter. This evaluation aims not only to validate the reliability of the generated answers but also the system's performance. To ensure a comprehensive analysis, numerous tests were conducted, systematically varying minor configuration settings in each run and other possible details that could affect the outcome, like the prompt given. These tests compare the answers obtained in a certain dataset of questions, comparing the performance of the system and the quality of the answers. This process helped to validate the design choices, made during the modeling phase and the implementation phase.

The chapter also provides a detailed description of the Amazon SageMaker environment used during the validation process, including its specifications, to draw a baseline of comparison for the performance observed. The chapter also describes the adopted methodology to generate the dataset utilized during the validation phase.

With the whole process explained and the results obtained, a discussion is made at the end of the chapter, aiming to offer a clear perspective of the advantages and disadvantages observed during the validation, describing the expected results of the RAG system.

5.1 Validation system

In the validation step, more than one system was applied to evaluate the obtained answers. So, the whole system contains many levels of actual analysis, going from a systematical way of comparing the performance of the system, given slightly different configurations, to a final human analysis of the question-answer pairs.

Dataset Generation

To train the dataset in a realistic context, a dataset was generated to perform consistent tests and achieve reliable results. The generated dataset consists of a group of questions related to the files stored in the vector database, and for this matter, the FAO-56 (Allen, 1998) file was used as the knowledge database for our tests.

5.1.1

The dataset generation system is built upon the original system, with several modifications implemented. Specifically, adjustments were made to the LLM and the text-feeding process, with the prompt completely redesigned to redefine the task. This system works by breaking the input file (FAO-56) into large chunks, after which the LLM generates questions for each chunk. The vector database component is removed as the new system's sole purpose is to generate the questions dataset.

The developed system takes the extracted text file generated with GROBID and divides it into chunks (chunking process explained in section 3.2.1), every chunk is then fed into Llama with a prompt explaining that the model should generate some questions that would be able to test the comprehension of a reader for the fed chunk. At the end, a text file is used as output, containing a set of questions for the utilized file. This process creates a vast set of questions, so some of the questions are hand-picked by a human and are then used for the validation scenario. Here is a sample to represent the contents of the generated dataset:

- What is evapotranspiration?
- What factors influence the transpiration rate in plants?
- What are some factors that can limit crop development and reduce evapotranspiration?
- What method is used to determine evapotranspiration by measuring the various components of the soil water balance?
- What procedure is recommended when some meteorological data are missing or cannot be calculated?
- What is the purpose of the crop coefficient (Kc) in calculating crop evapotranspiration?

Testing Methodology

The process of validation was done by changing the characteristics of the system that could affect the results. These characteristics range from simple model settings, like temperature that affects the model's creativity when answering, or vector database input settings like the chunking method, and the context retrieval size affecting the available information to the model when generating the answer.

5.1.2

The tests were done by fixing all the parameters and changing only one at a time to check the effect of the target parameter and what would be the best value for the tested target. The tests run by choosing a set of possibilities, and for each one, the system ran 3 times, repeating the test. The information obtained from the tests is very helpful in finding the most reliable configuration for our system.

The conducted tests save the information used and generated during the process. This means that not only are the answers obtained saved, but also intermediate processes like the information retrieved from the vector database, the question asked, the prompt used, and all the settings being tested. The tests are enhanced with complementary information, like the running time of certain processes. An additional simple classification was experimented by feeding the data that requires an interpretative analysis to the LLM. This classification is used in many cases, such as evaluating if each chunk retrieved from the vector database is relevant to the question asked, classifying if the answer generated aligns with the context retrieved, and assessing if the answer generated contains the information to answer the question. It is important to note that this generated information may contain inaccuracies, but it was decided to experiment with it because it helped to identify errors in the implementation.

5.2 Experimental Testing

The tests made on the system served two main objectives: as established, one of them was to validate the system and its reliability; and the other was to test the performance capacity of the system. For this matter, as the tests were run, an ideal configuration of the system was settled, this way obtaining an optimized system and defining some limitations in its capabilities.

This section provides a detailed presentation of the results observed from the experimental tests conducted throughout the course of the thesis. The data gathered during these tests will then be discussed to evaluate the system's performance, capabilities, and limitations.

System Configuration

The system configuration analyzes all the settings tested by exploring a set of possible values for each setting. To provide a better understanding of the system configuration, Table 3 represents the main settings explored in the tests. These settings are organized by the component affected and provide an additional description of their impact.

5.2.1

Table 3 - System Settings.

Setting	Purpose
Vector Database	
Chunk Size	Defines the maximum number of characters in a chunk
Retrieved Context Size	Number of chunks retrieved to be used as context for answer generation
LLM	Asda
Prompt*	The prompt contains the instructions for the LLM

The chunk size is the setting that affects the start of the pipeline, right before feeding the texts into the vector database, and it defines the maximum of characters contained in a chunk, in this case, a set of possible values was chosen (250, 500, 1000). This setting works with two other settings, the separators which define the characters that split the text (in this case {".",""\n","\n\n"}), and the chunk overlap setting that was set to 0, which affects how much of a chunk would be repeated into the next chunk.

A set of values was chosen for the retrieved context size setting (5, 10, 15, 20, 50, 75, 100, 150). These values represent the number of chunks provided by the vector database into the LLM when generating an answer, this setting may impact the speed of the answer generation, so it works as a trade of, more context, but slower generation.

The prompt is a very important configuration of the system, as it can get very hard to get right and even to test if it is being fully understood and followed by the LLM. During the developments, the prompt went through a process of reiteration as errors were identified. This evolution was mainly focused on the structure of the prompt and the format of the instructions given. On the LLM component, the LLaMa 3.1 model was used. For its configuration, there was also a relevant setting to mention, which was the temperature of the model, this setting affects the creativity of its generations. The model's temperature was set to 0 because it helps to create

a controlled generation where the model follows the instructions given, and the origin of the information used in the generation is known and truthful.

5.2.2 System Performance

In the system performance section, the direct results of tests are discussed, drawing a perception of the most optimized configuration of the system, and determining some of its limitations. The tests evaluated the system on many steps of the pipeline, so the section is divided into the diverse focus of the tests and following the order of action in the pipeline.

Table 4 - Overview of conducted tests and evaluation focus

T 1	**Vector Database**
T 1.1	Impact of number of chunks on Vector Database generation time
T 1.2	Retrieval grading and grading impact on generation time
T 2	**LLM**
T 2.1	Impact of answer length on time for different number of chunks retrieved
T 2.2	Impact of context size on performance
T 3	**Answer**
T 3.1	Agronomy expert grading on context retrieved and answer generation

Table 4 serves as a small introduction to the tests that were conducted and lists what tests were made and which part they were intended to evaluate/access.

Vector Database

For the vector database component, two tests were applied. The first one (Test 1.1) was run by simply feeding the document and recording the time that it takes to generate the Knowledge database. The second test (Test 1.2) was conducted by maintaining the system that evaluates the gathered context and evaluating its impact on the whole process of generating an answer. For both tests the question asked was "What is evapotranspiration?".

Table 5 presents the times obtained by feeding the FAO-56 document, which contains approximately 418000 characters. The times observed in Test 1.1 lead to the conclusion that the size of the chunks doesn't affect the time of generation of the vector database.

Table 5 - Test 1.1 Results (impact of number of chunks on Vector DB generation time).

Chunk Size	T1	T2	T3
250	0:09:42	0:09:38	0:09:45
500	0:09:14	0:09:14	0:09:10
1000	0:09:27	0:09:21	0:09:16

For the second group of tests, the vector database utilized was divided with the chunk size of 500 characters, the temperature of 0, and the number of documents retrieved from the vector database was varied with the values of 5, 10, 15, and 20. The prompt fed to the model is represented in Figure 14. This prompt is written to create a grading model that can identify whether the retrieved chunks are relevant to the question asked.

```
prompt_grade = PromptTemplate(
    template="""<|begin_of_text|><|start_header_id|>system<|end_header_id|>
    You are a grader assessing relevance of retrieved documents to a user
    question. If the documents contain keywords related to the user question,
    grade it as relevant. It does not need to be a stringent test. The goal
    is to filter out erroneous and useless retrievals. \n
    Give a binary score 'yes' or 'no' score to indicate whether the documents
    are relevant to the question and add no preamble or explanation.
     <|eot_id|><|start_header_id|>user<|end_header_id|>
    Here are the retrieved documents: \n\n {documents} \n\n
    Here is the user question: {question} \n
    <|eot_id|><|start_header_id|>assistant<|end_header_id|>
    """,
    input_variables=["question", "documents"],
)
```

Figure 14 Grading LLM Prompt.

Table 6 represents the times observed for retrieval plus grading. Initially, we can observe a proportional increase in the time taken with the number of chunks fed. This increase in the

time observed from 10 chunks to 15 chunks seems to be less than expected. With this disproportion in time, we can identify a problem, which is that the model starts to get overwhelmed and ignores part of the input. This conclusion is further consolidated with the results observed in Table 7.

The decrease in time observed from 15 chunks to 20 chunks is totally unexpected and raises the alert that something went wrong. The problem is identified when observing the outputs of the model.

Aside from the problems identified, this system also takes a very long time to grade the retrieved chunks. This problem leads to the conclusion that the system would not be a good feature to maintain when the model is taken to a production phase, maintaining the possibility of using it as a post execution validation or optional validation tool that the maintenance team could use to grade the retrieved context used to generate an answer. This eliminates the need for computational power when generating an answer, requiring only the storage of the elements used to generate an answer, which could be later accessed and graded.

Table 6 - Test 1.2 Results (grading system impact on generation time).

Nº of retrieved chunks	T1	T2	T3
5	0:04:55	0:04:50	0:04:39
10	0:09:57	0:10:07	0:09:52
15	0:12:12	0:11:56	0:11:54
20	0:09:20	0:09:28	0:09:23

Table 6 demonstrates the generated output of the grading LLM. The outputs observed remained consistent across T1, T2, and T3 (result expected as the temperature of the model was set to 0). In the first test with 5 chunks, the model provides the expected answer, but in further tests with 10 and 15, the model doesn't grade all the chunks retrieved. With 20 chunks, we can observe that the model completely ignores the task provided. This happens because the model may be unable to manage the growing complexity or volume of the input.

Table 7 - Test 1.2 Grading Response.

Nº of retrieved chunks	Generated Grading Example
5	{'Document1': 'yes', 'Document2': 'yes', 'Document3': 'yes', 'Document4': 'yes', 'Document5': 'yes'}
10	{'Document 1': 'yes', 'Document 2': 'yes', 'Document 3': 'yes', 'Document 4': 'yes', 'Document 5': 'yes', 'Document 6': 'yes', 'Document 7': 'yes'}
15	{'Document 0': 'yes', 'Document 1': 'yes', 'Document 2': 'yes', 'Document 3': 'yes', 'Document 4': 'no', 'Document 5': 'yes', 'Document 6': 'yes', 'Document 7': 'yes', 'Document 8': 'yes', 'Document 9': 'yes'}
20	{'answer': "Evapotranspiration (ET) is the process by which water is transferred from the land to the atmosphere through evaporation and transpiration. It's a critical component of the Earth's water cycle, and it plays a key role in regulating the amount of water available

The results lead to the belief that although it may be useful in an implementation phase, where it can identify some errors, the system is not applicable in a real context, therefore, the grading system is dropped from this point of the tests.

LLM

For test 2.1, small changes were applied to the prompts to check the response from the system. The base prompt is presented in Figure 15. This prompt gives no instructions lengthwise, providing only guidelines for the task to be carried out.

Figure 16 and Figure 17 represent the changes applied to the base prompt. In Figure 16, the instructions to create a concise answer within three sentences are added to the prompt. This prompt is created with the main objective of observing the model's behavior when instructed to create an answer of a predefined length. In the third prompt, presented in Figure 17, is added the instruction to create an extensive response and to use all the information available gathered by the Vector Database. This modification of the prompt aims for complete and insightful answers.

The test is created not only to compare the model change of behavior when presented with different length instructions, but also to compare the model performance, comparing the generation duration.

```
prompt = PromptTemplate(
    template="""<|begin_of_text|><|start_header_id|>system<|end_header_id|>
    You are an assistant for question-answering tasks.
    Use only the following pieces of retrieved context to answer the question.
    Don't base the answer on information not provided. If you don't have enough
    information, just say that you don't know.
    <|eot_id|><|start_header_id|>user<|end_header_id|>
    Question: {question}
    Context: {context}
    Answer: <|eot_id|><|start_header_id|>assistant<|end_header_id|>""",
    input_variables=["question", "context"],
)
```

Figure 15 Prompt with no length guideline.

```
prompt_short = PromptTemplate(
    template="""<|begin_of_text|><|start_header_id|>system<|end_header_id|>
    You are an assistant for question-answering tasks.
    Use only the following pieces of retrieved context to answer the question.
    Don't base the answer on information not provided.
    If you don't have enough information, just say that you don't know.
    Use three sentences maximum and keep the answer concise
    <|eot_id|><|start_header_id|>user<|end_header_id|>
    Question: {question}
    Context: {context}
    Answer: <|eot_id|><|start_header_id|>assistant<|end_header_id|>""",
    input_variables=["question", "context"],
)
```

Figure 16 Prompt for short answers.

```
prompt_long = PromptTemplate(
    template="""<|begin_of_text|><|start_header_id|>system<|end_header_id|>
    You are an assistant for question-answering tasks.
    Use only the following pieces of retrieved context to answer the question.
    Don't base the answer on information not provided. If you don't have enough
    information, just say that you don't know.
    Provide an extensive response, using all the information available
    <|eot_id|><|start_header_id|>user<|end_header_id|>
    Question: {question}
    Context: {context}
    Answer: <|eot_id|><|start_header_id|>assistant<|end_header_id|>""",
    input_variables=["question", "context"],
)
```

Figure 17 Prompt for long answers.

- Answer for base prompt (Figure 15): "Evapotranspiration is the combination of two separate processes: evaporation from the soil and transpiration from plants, resulting in the loss of water from a system to the atmosphere."
- Answer for short answer instruction prompt (Figure 16): "Evapotranspiration (ET) is the process by which water is transferred from the land to the atmosphere through evaporation and transpiration. It is calculated as the sum of evaporation from the soil and transpiration from plants, and is denoted as ET or ETo. The reference crop evapotranspiration (ETo) is a specific type of ET that refers to the rate at which water is evaporated from a hypothetical grass surface with specific characteristics, not short of water."
- Answer for long answer instruction prompt (Figure 17): "Evapotranspiration (ET) is the process by which water is transferred from the land to the atmosphere through evaporation and transpiration. It is a critical component of the Earth's hydrologic cycle, and understanding ET is essential for managing water resources, predicting droughts and floods, and optimizing crop yields.

  In this context, evapotranspiration refers to the rate at which water is lost from a surface, including crops, soil, and other vegetation. This process involves two main components: evaporation (the transfer of water from the soil or surface to the air) and transpiration (the release of water vapor by plants).

  The calculation of ET can be complex, as it depends on various factors such as climate, soil type, crop characteristics, and weather conditions. However, a commonly used method for estimating ET is the reference evapotranspiration (ETo), which represents the maximum amount of ET that would occur from a hypothetical grass surface with specific characteristics.

  The ETo value is calculated using the FAO Penman-Monteith method, which takes into account various climatic parameters such as temperature, humidity, wind speed, and solar radiation. This method has been widely adopted due to its accuracy and simplicity.

  In summary, evapotranspiration (ET) is a critical process that involves the transfer of water from land to atmosphere through evaporation and transpiration. The reference evapotranspiration (ETo) value represents the maximum ET rate for a hypothetical grass surface and is calculated using the FAO Penman-Monteith method."

As we can observe from the outputs generated, the first answer appears overly direct and leaves key points unexplained. This answer can be suitable for a general audience, but that's not the aim of the thesis. The second answer provides a moderate amount of information viable for a professional and practical context, maintaining the answer concise. However, the answer will always lack contextual details, application examples, and certain insights. The third answer presents the biggest amount of detail by far, and although it provides a holistic answer, the length of the answer can be an opening for hallucination occurrences and can present less correct and unwanted details when in a very general topic.

The analysis reveals a possible trade-off between conciseness and depth, highlighting how prompt formulation must carefully balance informativeness with reliability. These characteristics observed contribute to a further understanding of the behavior of the model when faced with different prompts, helping with the optimization of prompt strategies that support an accurate, context-rich output in this specific application.

Table 8 - Test 2.1 Results (prompt/short_prompt/long_prompt times).

Nº of retrieved chunks	T1	T2	T3
5	0:03:55/0:04:42/0:04:13	0:03:48/0:04:39/0:04:10	0:03:47/0:04:34/0:04:09
10	0:08:29/0:09:44/0:11:23	0:08:37/0:09:58/0:11:04	0:08:19/0:09:34/0:10:47
15	0:10:38/0:11:20/0:11:21	0:10:35/0:11:30/0:11:06	0:10:35/0:11:20/0:11:04
20	0:17:21/0:18:47/0:22:35	0:17:12/0:18:33/0:22:53	0:18:16/0:19:31/0:24:09

The same question was asked while the context size was changed. T1, T2, and T3 correspond to three repetitions of each test configuration to assess consistency in response generation time. The empirical results demonstrate that while the requested answer length has minimal impact at low context levels, its influence grows substantially as more retrieved chunks are included.

The combination of a long expected output and a large number of retrieved chunks lead to significantly higher generation times. However, this growth appears proportional to the increased volume of text being generated and processed. These findings highlight the importance of balancing context size and output length in prompt design, particularly in scenarios where system responsiveness and efficiency are critical.

Test 2.2 focuses more on testing the limits of the model, providing it with increasing numbers of context chunks to assess its behavior.

The generation times results presented in test 2.2 were obtained using a different computational environment than those in test 2.1. This change was intentional, aiming to leverage greater computing capacity to handle more complex or resource-intensive testing scenarios. As a result, direct comparison of generation times between these tests is not valid. Nonetheless, the data is included for completeness, while the primary focus remains on evaluating the model limits. This change of environment also provides an insight into the impact of different resources available to the model.

Table 9 - Test 2.2 impact of context size on performance.

Nº of retrieved chunks	Generation time	Focus maintained
5	0:03:37/0:03:31/0:04:08	yes
10	0:02:55/0:02:31/0:02:32	yes
15	0:03:16/0:03:27/0:03:57	yes
20	0:04:07/0:03:34/0:03:33	yes
50	0:15:48/0:22:42	yes
75	0:10:19	No
100	0:09:39	No
150	0:09:40	No

As shown in Table 9, the model maintained question focus consistently up to 50 retrieved chunks, despite increasing generation time. However, beyond this point, the mode was no longer capable of retaining the question asked, answering something along the lines of **"It seems like you provided a large block of text related to evapotranspiration, but didn't ask a**

specific question. ...". This behavior indicates a threshold where excessive context began to degrade performance.

The threshold found can be reached by two different means, the first one is associated with the model architecture, in this case, the Llama 3.1, which can process a window of 128k tokens. The second way is by the available computational power available to the model. In this case, the token limit is not reached. With the rule of thumb that says that one token generally corresponds to ~4 characters of text for common English text (Which can be translated as roughly 0.75 of a word, so 100 tokens ~= 75 words) (OpenAI, 2025). By rounding up the 75 chunks to 250 characters (which never happens because the splitter uses specific characters before reaching 250), plus ignoring the tokens of the question (what is evapotranspiration?), which would add no more than 10 tokens. All this would add to a total of 4*250*75=75000 tokens, still inferior to the 128000 tokens. With this, we can conclude that for at least 75 chunks, the model is limited by the available computational power.

On another note, it is also possible to identify that for 10 chunks, the model generates an answer faster. This is possibly due to the relevance of the chunks provided to the model. While it seems logical to conclude that the model should always use 10 chunks, this needs further testing, because for different matters, there are different amounts of available information, which can result in a different optimal number of retrieved chunks. Also, a faster answer doesn't always mean a better answer.

For test 2.3, a small test was developed with the intent of testing if the model would follow the instructions to clarify that it doesn't have relevant information available to answer the question, if that is the case.

The test consists of asking two different questions and swapping the gathered context between them. With this setup, we ensure that the context provided to the model doesn't correspond to the question being asked, and we can assess whether the model accurately recognizes the absence of sufficient information rather than attempting to generate an answer based on previously known information acquired from its pretraining.

First, the model was provided with 2 questions:

- **Q1:** What is evapotranspiration?
- **Q2:** What can be used as an approximation for cloudiness during nighttime periods?

To confirm the model's ability to generate accurate answers, each question was initially paired with the correct and relevant context. This ensured that the model had the necessary information to respond appropriately. Subsequently, the contexts were swapped between the two questions, representing the context as "CX". Resulting in the following outputs:

- Q1 C2: "Based on the provided context, I don't have a direct definition of evapotranspiration. However, I can infer that it might be related to the climatic parameters discussed in the text.

  From the context, I see that "evapotranspiration" is not explicitly mentioned, but there are mentions of "ET o" which seems to be related to evapotranspiration. The phrase "Precise estimates of ET o for specific hourly periods may require the use of aerodynamic stability functions and functions for modifying the value for r s based on levels of radiation, humidity and temperature" suggests that ET o is a measure of evapotranspiration.

  However, without a clear definition or explanation in the provided context, I can only make an educated guess. If you have any additional information or context about evapotranspiration, please let me know!"
- Q2 C1: "The context provided does not mention cloudiness at all. It discusses evapotranspiration, its measurement, factors affecting it, and related concepts, but there is no information about cloudiness or how to approximate it during nighttime periods.

  Therefore, I don't know the answer to your question."

In both test cases, the model rightfully recognized when the provided context lacked explicit information and didn't generate an answer based on prior knowledge as instructed. While this behavior fulfills the intended function for controlled testing, it may appear slightly unnatural in a conversational setting and final product scenario, because the model mentions the context that is never provided to the user directly.

Following the same line of thought as the previous test (test 2.3), Test 2.4 also assesses the capacity of the model to say that it doesn't have enough information to formulate an answer, when it's the case. For this matter, an empty context was provided to model, and this is the outcome:

- "I'm happy to help! Unfortunately, it looks like there's no context provided for me to draw from. But I can still give you a straightforward answer.

  Evapotranspiration (ET) is the process by which water is transferred from the Earth's surface into the atmosphere through evaporation and plant transpiration. It's an essential component of the Earth's hydrologic cycle, playing a crucial role in regulating the planet's climate and weather patterns.

  In simple terms, ET occurs when plants absorb water from the soil and release it as water vapor into the air through their leaves (transpiration), while also evaporating water directly from the surface of the soil and bodies of water. This process is driven by solar energy and can be influenced by factors such as temperature, humidity, wind speed, and atmospheric pressure.

  ET is a critical factor in understanding various environmental processes, including climate modeling, agriculture, and hydrology. It's also an important aspect of managing water resources, as it helps us estimate the amount of water that's being lost from the system due to evaporation and plant transpiration.

  Would you like me to elaborate on any specific aspect of ET?"

As we can see in the output, the model identifies that there's no context to pull data from, but instead of restraining from generating an answer, the model proceeds to generate an answer based on its pretraining data. In certain cases. While this behavior may be advantageous in certain applications where answers based on previous knowledge are acceptable, it doesn't align with the goals of the current thesis, which prioritizes context-based generation.

Agronomy Expert Evaluation

For the final test, experts evaluated both the retrieved context and the generated answer, scoring each on a scale from 1 to 3. The results generated used a vector database with a max chunk size of 250 and splitter characters ("\n", "\n\n", "."). The temperature of the model was set to 0, and the number of retrieved chunks used was 40, because with 50, the model showed some evidence of forgetting the question asked. This configuration was chosen to push the

model to its limit, not focusing on the system's time efficiency. This way, aiming for the best generation performance.

Table 10 - Test 3.1 Expert Evaluation.

Question asked	Context evaluation	Answer evaluation
what is evapotranspiration?	2	2
What factors influence the transpiration rate in plants?	3	3
What are some factors that can limit crop development and reduce evapotranspiration?	2	2
What method is used to determine evapotranspiration by measuring the various components of the soil water balance?	3	3
What procedure is recommended when some meteorological data are missing or cannot be calculated?	3	2
What is the purpose of the crop coefficient (Kc) in calculating crop evapotranspiration?	3	3
how does the percentage of the ground surface is typically covered by vegetation correlated with K c values?	1	1
150What is the main difference between the single Kc coefficient approach and the dual crop coefficient approach in computing evapotranspiration?	3	3
What are the steps involved in calculating crop evapotranspiration (ETc)?	3	3
What is the readily available soil water, and how does it relate to the crop's ability to extract water from the root zone?	1	1
What are some effects on evapotranspiration of salts in the soil water solution?	2	3
What determines the value for irrigation depth?	1	1
Can you list some ways of increasing crop growth rates and vegetable yields?	1	1

The classification method idealized for the context rates it with how accurately and concisely the retrieved information addressed the question, ranging from entirely incorrect (1), to partially relevant with either excess or insufficient detail (2), to correct and concise (3). For the generated answers, a score of 1 indicated an incorrect response, 2 reflected mostly correct answers with some lack or surplus of detail, and 3 denoted a fully correct answer.

Table 10 presents the results of the evaluation conducted. The answers scored with 3 on both context and answer quality were considered good answers. Answers with a score of 2 in either classification were considered mainly correct, but with some flaws. For example:

- In the case where the context is evaluated with 2 and the answer with 3, the context was too long, which led to a less concise answer.
- In the case where the context is classified with 3 and the answer with 2, the answer itself is considered correct, but vague.
- In cases where both criteria were classified as 2, the answers confuse some concepts (mainly evapotranspiration, possibly because it is a concept that is explored throughout the whole FAO 56 document).

For the cases where both criteria were classified as 1, the answers were considered wrong, either for confusing similar concepts or simply not answering the question asked.

The number of answers analyzed is admittedly a small range of possibilities, limiting how conclusions and generalizations can be made from the results. However, the test size was constrained by the nature of the task, which consisted of a manual analysis of extended contexts retrieved, prioritizing the analysis's quality over quantity.

5.3 Results discussion

The results observed in the various tests provide important information that leads to key insights distributed throughout all the components assessed.

Starting with the vector database component, experiments demonstrated that the initial time required to build the vector store does not depend substantially on the chunk size (observed in Test 1.1). This characteristic may also be observed because smaller chunks imply more chunks, which leads to less time in each embedding phase, but more embeddings, counterbalancing itself.

In Test 1.2, the system was tested for the possibility of implementing a real-time grading system that would assess the context retrieved for the question asked, grading each chunk and, subsequently, grading the capacity of the vector database to retrieve relevant chunks. The idea deserves better attention in the future, but only as a test phase grading system, because, as observed, the grading system would maintain many flaws like ignoring part of the chunks,

not having enough credibility, and hallucinating with an extensive number of chunks which would turn the system not reliable for a deployment phase of the full system. Furthermore, it shows a sign of slowing the answer time generation, possibly duplicating the overall time, becoming even less reliable in the final deployment phase.

Turning to the LLM component, the results of Test 2.1 demonstrate that even minor changes in the prompt can significantly affect the model's output. The base prompt produced an overly general answer, omitting details necessary for professional contexts. Introducing the short-answer guideline led to an improvement in the technical content of the answer while maintaining conciseness, suggesting that explicit length instructions help focus the model without overwhelming detail. However, instructing the LLM to generate an extensive answer resulted in far richer and more comprehensive responses, including technical explanations and background. While in this case, the longer answer proved to be valuable, the long-answer prompt can be harmful in other cases because the longer outputs also risk introducing irrelevant or hallucinated content, particularly when the available context is limited or inconsistent. This phenomenon is observed in Test 3.1, where some answers were classified as bad. More tests would be needed to prove the relation between a long answer and hallucination.

Furthermore, the results show a trade-off between the context size and the generation time of the model. At a small number of chunks retrieved, there isn't a noticeable impact from the length instruction. Yet as the number of retrieved chunks increased, long-answer prompts caused a substantial rise in generation duration, scaling proportionally with the text volume being processed. This suggests that for applications where responsiveness and efficiency are critical, prompt design must carefully balance output length and context size to avoid latency issues without sacrificing informational quality. This balance is hard to achieve because the ideal number of chunks to retrieve changes from question to question, and the relevant information available will vary depending on the topic of the question.

For Test 2.2, the model was stressed to its limit. As the results show, the model reaches its limit, forgetting completely the question asked. This "loss of memory" is caused by the vast and increasing number of tokens fed to the model, which happens way before the model's theoretical maximum. This leads to the belief that the available computational power also limits the model.

Tests 2.3 and 2.4 may seem irrelevant, but they proved very helpful in the early stages of the thesis and added another layer of certainty about the system's behavior. The tests examined what happens in cases where the retrieved context doesn't have relevant information

for the question asked. Importantly, the tests uncovered cases where the model was being overfed with information, losing the instructions of the prompt and the context retrieved, leading to the generation of an answer based on previous knowledge of the model.

Finally, looking at the data of Test 3.1, we can, firstly, observe that the quality of the answer is correlated with the quality of the context retrieved. For an overall view, the model can generate a satisfactory answer roughly 70% of the time. But in the other 30%, the model generates answers that are based on weak or incomplete context, leading to inaccurate or misleading answers. This emphasizes the crucial role of high-quality context retrieval.

For an overall review of the system's performance, there are 2 main negative points to highlight. Firstly, the system takes more time than ideal to generate an answer. This may be due to the computational power available in the testing environments. Therefore, for further investigation, the system should be tested in an environment fully dedicated and envisioned for LLMs, alongside efficiency strategies such as RAGCache, which caches and reuses relevant information from previously retrieved documents rather than recomputing it for each query (Jin et al., 2025).

Secondly, although the system shows a high rate of satisfactory answers, there is still plenty of room for improvement. The limitation seems to derive from the context retrieved, as a worse context leads to a worse answer. This challenge may be harder to tackle, possibly meaning a whole modification/addition to the database module. New strategies for generation quality could be explored, such as a state-aware dynamic retrieval mechanism that continuously adapts document selection during generation through joint optimization of retrieval and generation modules (He et al., 2025), as well as Agentic RAG, where autonomous agents filter the information retrieved (Singh et al., 2025).

6

FINAL REMARKS

6.1 Conclusion

Despite the big evolution seen in the branch of Artificial Intelligence, there is still a long path to follow. New applications are being explored and new ways of making the technology more efficient and accurate are surging. This evolution leads to the opportunity of creating decision-making knowledge assistants and accident prevention assistants.

This thesis explored the development and evaluation of a local RAG LLM system designed to process and store knowledge from PDFs and then generate relevant insights for questions asked by the user, using the knowledge previously given to the system. The system was built on the integration between a vector database and LLM, using mainly the LangChain framework that enabled the collaboration between tools like Ollama and ChromaDB. The system created fits within the first steps that lead the transition from Agriculture 4.0 to Agriculture 5.0. The system enables academic and complex knowledge to be understood by everyone by simplifying it and focusing on the topics separately.

While the current system demonstrates promising results, it requires further refinement to achieve optimal performance. Specifically, it shows drawbacks in the generation time, taking too long, and in the quality of the context retrieved from the database, possibly because the database contained either too little or too much information in certain topics for the number of chunks selected for retrieval.

6.2 Future Work

Besides tackling the previously discussed drawbacks of the system, by trying new strategies such as RAG derivations, more computational power, or even a test with cloud computing for time and quality improvement, the system serves as a base and opens the door for many new projects and applications.

One important direction for future work involves conducting additional testing to assess its performance with multi-query scenarios, a very important capability to maintain a natural conversational flow with human users. This feature would mean less question contextual size availability for the model, because its memory would also be filled with chat memory. Overall, a positive tradeoff if the objective is to keep it natural.

Another essential aspect of improvement is the potential new adjustments in its core prompt to then evaluate its performance when talking in languages other than the base language of its stored knowledge. This is a crucial matter for the thesis because its core idea revolves around the democratization of knowledge, and it needs to be available in the user's native language.

In addition, the system was designed to be updatable, and this aspect goes beyond the knowledge database. The modular architecture of the system allows for the update or replacement of the components as technology advances. In particular, the LLM can be exchanged for newer or more specialized ones, as happened during the development. This way, ensuring the chatbot remains aligned with the latest developments in Ai while preserving its structure.

In the scope of a final product there's also a flag system to explore. This system would identify areas of knowledge where the database would be missing relevant information. This way the database could be in constant improvement as it was projected to be.

Finally, the initial development aimed to develop a chatbot that assists in irrigation optimization. The prediction model had already been researched and developed in a parallel organization project. Therefore, the next step is to integrate the prediction model into the chatbot, making it possible to ask the chatbot itself for help in irrigation optimization. If this step proves feasible, then there's a whole new world of possibilities to explore, such as integration with computational vision to detect pests or diseases, crop growth monitoring with NDVI, yield forecasting, and much more.

7

BIBLIOGRAPHY

Aceto, G., Persico, V., & Pescapé, A. (2019). A Survey on Information and Communication Technologies for Industry 4.0: State-of-the-Art, Taxonomies, Perspectives, and Challenges. In *IEEE Communications Surveys and Tutorials* (Vol. 21, Issue 4, pp. 3467–3501). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/COMST.2019.2938259

Allen, R. G. ; P. L. S. ; R. D. S. M. (1998). *Crop evapotranspiration : guidelines for computing crop water requirements - FAO Irrigation and drainage paper 56*. Food and Agriculture Organization of the United Nations. https://www.fao.org/4/x0490e/x0490e00.htm

Ameen, A., & Raza, S. (2017). International Journal of Advances in Scientific Research Green Revolution: A Review QR Code *Correspondence Info. *International Journal of Advances in Scientific Research*, *3*(12), 129–137. https://doi.org/10.7439/ijasr

Balaguer, A., Benara, V., Cunha, R. L. de F., Filho, R. de M. E., Hendry, T., Holstein, D., Marsman, J., Mecklenburg, N., Malvar, S., Nunes, L. O., Padilha, R., Sharp, M., Silva, B., Sharma, S., Aski, V., & Chandra, R. (2024). *RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture*. http://arxiv.org/abs/2401.08406

Cui, Y., Zhou, F., Wang, J., Liu, X., Lin, Y., & Belongie, S. (2017). Kernel Pooling for Convolutional Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3049–3058. https://doi.org/10.1109/CVPR.2017.325

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Fasciolo, B., Panza, L., & Lombardi, F. (2024). Exploring the Integration of Industry 4.0 Technologies in Agriculture: A Comprehensive Bibliometric Review. In *Sustainability (Switzerland)* (Vol. 16, Issue 20). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/su16208948

Fountas, S., Espejo-Garcia, B., Kasimati, A., Gemtou, M., Panoutsopoulos, H., & Anastasiou, E. (2024). Agriculture 5.0: Cutting-Edge Technologies, Trends, and Challenges. *IT Professional*, *26*(1), 40–47. https://doi.org/10.1109/MITP.2024.3358972

Gers, F. A., Urgen Schmidhuber, J. J., & Cummins, F. (1999). Learning to Forget: Continual Prediction with LSTM. In *IEE* (Vol. 2). http://www.idsia.ch/http://www.idsia.ch/

Gobrid. (n.d.). Retrieved May 6, 2024, from https://github.com/kermitt2/grobid

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

He, J., Liu, G., Zhu, B., Zhang, H., Zheng, H., & Wang, X. (2025). Context-Guided Dynamic Retrieval for Improving Generation Quality in RAG Models. *2025 IEEE 7th International Conference on Communications, Information System and Computer Engineering (CISCE)*, 939–943. https://doi.org/10.1109/CISCE65916.2025.11065272

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). *Measuring Massive Multitask Language Understanding*. http://arxiv.org/abs/2009.03300

Holzinger, A., Fister, I., Fister, I., Kaul, H. P., & Asseng, S. (2024). Human-Centered AI in Smart Farming: Toward Agriculture 5.0. *IEEE Access*, *12*, 62199–62214. https://doi.org/10.1109/ACCESS.2024.3395532

Hou, Z., Salazar, J., & Polovets, G. (2022). Meta-Learning the Difference: Preparing Large Language Models for Efficient Adaptation. *Transactions of the Association for Computational Linguistics*, *10*, 1249–1265. https://doi.org/10.1162/tacl_a_00517

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 328–339. https://doi.org/10.18653/v1/P18-1031

Hugging Face. (n.d.). *Hugging Face*. Hugging Face. Retrieved February 12, 2025, from https://huggingface.co/

Jin, C., Zhang, Z., Jiang, X., Liu, F., Liu, S., Liu, X., & Jin, X. (2025). RAGCache: Efficient Knowledge Caching for Retrieval-Augmented Generation. *ACM Transactions on Computer Systems*. https://doi.org/10.1145/3768628

Johnson, J., Douze, M., & Jegou, H. (2021). Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, *7*(3), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

LangChain. (n.d.). *LangChain*. LangChain Documentation. Retrieved February 12, 2025, from https://python.langchain.com/docs/introduction/

Lester, B., Al-Rfou, R., & Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3045–3059. https://doi.org/10.18653/v1/2021.emnlp-main.243

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. https://doi.org/10.48550/arXiv.2005.11401

Li, H., Choi, J., Lee, S., & Ahn, J. H. (2020). Comparing BERT and XLNet from the Perspective of Computational Characteristics. *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, 1–4. https://doi.org/10.1109/ICEIC49074.2020.9051081

Li, Z., Zhu, H., Lu, Z., & Yin, M. (2023). Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 10443–10461. https://doi.org/10.18653/v1/2023.emnlp-main.647

Liu, Y., Ma, X., Shu, L., Hancke, G. P., & Abu-Mahfouz, A. M. (2021). From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges. *IEEE Transactions on Industrial Informatics*, *17*(6), 4322–4334. https://doi.org/10.1109/TII.2020.3003910

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. http://arxiv.org/abs/1907.11692

Lowenberg-Deboer, J., & Erickson, B. (2019). Setting the record straight on precision agriculture adoption. In *Agronomy Journal* (Vol. 111, Issue 4, pp. 1552–1569). American Society of Agronomy. https://doi.org/10.2134/agronj2018.12.0779

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., & Scialom, T. (2023). *GAIA: a benchmark for General AI Assistants*. http://arxiv.org/abs/2311.12983

OpenAI. (2025, June 10). *OpenAI Tokenizer*. https://platform.openai.com/tokenizer

Pearce, K., Zhan, T., Komanduri, A., & Zhan, J. (2021). *A Comparative Study of Transformer-Based Language Models on Extractive Question Answering*. https://doi.org/10.1007/978-3-031-08473-7_45

Pilevari, N., & Yavari, F. (2020). Journal of Industrial Strategic Management Industry Revolutions Development from Industry 1.0 to Industry 5.0 in Manufacturing Industry Revolutions Development from Industry 1.0 to Industry 5.0 in Manufacturing. In *Journal of Industrial Strategic Management* (Vol. 2020, Issue 2).

Razak, S. F. A., Yogarayan, S., Sayeed, M. S., & Derafi, M. I. F. M. (2024). Agriculture 5.0 and Explainable AI for Smart Agriculture: A Scoping Review. In *Emerging Science Journal* (Vol. 8, Issue 2, pp. 744–760). Ital Publication. https://doi.org/10.28991/ESJ-2024-08-02-024

Saiz-Rubio, V., & Rovira-Más, F. (2020). From smart farming towards agriculture 5.0: A review on crop data management. In *Agronomy* (Vol. 10, Issue 2). MDPI. https://doi.org/10.3390/agronomy10020207

Scao, T. Le, Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., … Wolf, T. (2022). *BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. BigScience Workshop*. http://arxiv.org/abs/2211.05100

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Seo, M., Kembhavi, A., Farhadi, A., & Hajishirzi, H. (2017). *Bidirectional Attention Flow for Machine Comprehension*. http://arxiv.org/abs/1611.01603

Singh, A., Ehtesham, A., Kumar, S., & Khoei, T. T. (2025). *Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG*. http://arxiv.org/abs/2501.09136

Smagulova, K., & James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. In *European Physical Journal: Special Topics* (Vol. 228, Issue 10, pp. 2313–2324). Springer Verlag. https://doi.org/10.1140/epjst/e2019-900046-x

Sudalairaj, S., Bhandwaldar, A., Pareja, A., Xu, K., Cox, D. D., & Srivastava, A. (2024). *LAB: Large-Scale Alignment for ChatBots*. http://arxiv.org/abs/2403.01081

Teixeira, J. P., & Fernandes, P. O. (2012). Tourism Time Series Forecast -Different ANN Architectures with Time Index Input. *Procedia Technology*, *5*, 445–454. https://doi.org/10.1016/j.protcy.2012.09.049

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). *LLaMA: Open and Efficient Foundation Language Models*. http://arxiv.org/abs/2302.13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. ukasz, & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. Von Luxburg, S. Bengio, H.

Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2019). SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., & Raffel, C. (2021). *mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer*. https://pypi.org/project/langdetect/

Zhao, X., Fang, Y., Liu, L., Xu, M., & Zhang, P. (2020). Ameliorated moth-flame algorithm and its application for modeling of silicon content in liquid iron of blast furnace based fast learning network. *Applied Soft Computing Journal*, *94*. https://doi.org/10.1016/j.asoc.2020.106418

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in Neural Information Processing Systems* (Vol. 36, pp. 46595–46623). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf