KU LEUVEN

FACULTY OF SCIENCE

Exploring Large Language Models for analyzing Aviation Safety Data

Jade LEJEUNE HERMAN

Supervisor:
Prof. Jan De Spiegeleer

Co-supervisor:
Leopold Viroles (EASA)
Guillaume Soudain (EASA)

Master's Thesis presented in
fulfilment of the requirements
for the degree of Master of Science
in Statistics and Data Science

Academic year 2023-2024

Preface

This Master's Thesis represents an important stage of the Master of Science in Statistics and Data Science at KU Leuven. It presents the exploration of Large language Models for analyzing Aviation Safety Data.

The completion of this Master's Thesis would not have been possible without the support and encouragement of many individuals to whom I wish to express my gratitude.

First and foremost, I would like to express my gratitude to my supervisor, Professor De Spiegeleer, for agreeing to supervise my Master's Thesis, for his insightful advice, his support throughout this project, enabling me to develop my own ideas and pursue my own path.

I would also like to warmly thank EASA for providing me with the opportunity to conduct this research within their organization. I am particularly grateful to Leopold Viroles and Guillaume Soudain for their collaboration, valuable assistance and comments. I also thank the Data4Safety team for allowing me to conduct tests on the platform at the end of my Master's Thesis and the service provider ALG-Global. A big thank you to the various individuals at EASA who contributed to the completion of this Master's Thesis, including Dominique Fouda, Luc Tytgat... Thank you also to François Triboulet from Eurocontrol for his precious advice and feedbacks on the implementation, results and validation parts.

I wish to express my gratitude to KU Leuven and the faculty members for their academic and administrative support. Special thanks to Professors Van Keilegom and Carbonez for their guidance and help throughout my studies.

I will not forget my friends and fellow students I met during my studies for their friendship, moral support, and the many enriching discussions we had together.

Finally, my thanks go to my family for their love, support, and understanding throughout this journey. Their encouragements have been a constant source of motivation and strength.

Thank you to everyone who, near and far, contributed to the completion of this Master's Thesis.

Contribution Statement

This Master's Thesis is the fruit of my labor and has been written under the guidance of my supervisor. Here are the key aspects of my Master's Thesis.

- Original thesis topic idea: The Master's Thesis proposal was initiated by EASA. All aspects and approaches, unless otherwise specified, are my own initiative.
- Bibliography: Several bibliographical sources were consulted for the completion of this Master's Thesis. References are cited in the text or in the figure and table captions.
- Database: The safety data used are from the EASA Data4Safety initiative.
- Figures and Tables: Captions for all figures and tables clearly indicate ownership.
  - Original figures and tables created by the author: Labelled as "Created by author" including code details if necessary.
  - Figures and Tables created by the author using external data: Labelled as "Created by author from [citation]" or "Created by author with the data from [citation]".
  - Figures and Tables adapted by author from external sources: Labelled as "Adapted by author from [citation]".
  - Figures and Tables extracted from external sources: Labelled as "Extracted from [citation]".
  - Figures and Tables reproduced by author from external sources: Labelled as "Reproduced by author from [citation]".
- R, Python Codes, `Jupiter` Notebooks and `Databricks` Notebooks: The codes were created and implemented by the author.
- Service Provider ALG-Global: This is the service provider appointed by EASA to develop the Data4Safety platform. They provided the `Databricks` development environment, including the database catalog, blank notebooks for development, and serving endpoints for accessing AI models.
- Evaluation Results: All results were produced by the author.

Abstract

The Master's Thesis, conducted in collaboration with the European Union Aviation Safety Agency (EASA), aims to develop an AI tool for natural language interaction with aviation safety data on the platform called Data4Safety. The research focuses on implementing Large Language Models (LLMs) to address the challenges posed by the volume and complexity of the occurrence reporting database ECCAIRS2, a key component of the platform.

The Master's Thesis starts by providing the contextual background, the methodology and the research's objectives. Then, an analysis of the database is conducted, including data flow assessments, database structure explorations and data profiling exercises. This analysis is needed for understanding the database's functionality, identifying its structure and exploring the data it contains. The resulting knowledge will facilitate the implementation of the LLMs solution and allow future improvements. Following this, a state of art of the generative AI and the Large Language Models is performed, in particular, by reviewing their models, detailing their characteristics, presenting mathematical representations and developing use cases to illustrate their principles and enhance understanding.

Then, the implementation of LLM solutions to interact with the database is outlined and analysed against industrial constraints. Two AI framework and their architectures are examined. The first uses local AI models with a Flask-powered interface, while the second employs AI models via notebooks on a `Databricks` server. Details of both implementations are discussed. Additionally, two techniques are explored: Text2SQL, which generates SQL queries from natural language prompts, and the Retrieval Augmented Generation (RAG) technique, which retrieves optimal outcomes from data, providing fine-tuned context to a pre-trained model.

Next the different solutions implemented were tested and compared using a created validation set with about forty test prompts to evaluate fundamental and reasoning capabilities. Execution accuracy was chosen as the metric, assessing the full pipeline's performance for database analysis. Validation techniques are reviewed including the EASA learning assurance process and adaptations are proposed for LLMs. Lack of repeatability and evaluation metrics are also discussed. Finally, the Master's Thesis concludes with a summary of the key findings and outlines potential avenues for future research and improvements based on the insights collected from the study.

List of abbreviations

AI Artificial Intelligence

AIGC Artificial Intelligence Generated Content

API Application Programming interface

ATM Air Traffic Management

biLSTM bidirectional Long Short-Term Memory

BERT Bidirectional Encoder Representations from Transformers

BLEU Bilingual Evaluation Understudy

COTS Commercial Off-The Shelf

D4S Data4Safety

EC European Commission

EU European Union

EASA European Union Aviation Safety Agency

ECCAIRS European Co-ordination Center for Accident and Incident Reporting Systems

ELMo Embeddings from Language Models

Gen-AI Generative AI

GPT Generative Pre-trained Transformer

GPT-2 Generative Pre-trained Transformer 2

GPT-3 Generative Pre-trained Transformer 3

GPT-4 Generative Pre-trained Transformer 4

IT Information Technology

LaMDA Language Model for Dialog Applications

LLM Large Language Models

LSTM Long Short-Term Memory

ML Machine Learning

MLM Masked Language Modelling

MS Member States

NAA National Aviation Authority

NLP Natural Language Processing

NSP Next Sentence Prediction

ODD Operational Design Domain

PLM Pre-trained Language Models

RAG Retrieval Augmented Generation

ROUGE Recall Oriented Understudy for Gisting Evaluation

RRN Recurrent Neural Networks

SIA Safety Investigation Authority

SLM Statistical Language Models

SQL Structured Query Language

t-SNE t-Distributed Stochastic Neighbour Embedding

T5 Text-to-Text Transfer Transformer

Word2Vec Word to Vector

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Context

The aviation is widely recognized as a leader in safety and many other industries look to aviation for its practices. This reputation can be explained by the expectation of achieving high safety standards but also by the certification process that every aircraft undergoes before being allowed to fly under the oversight of the aviation safety authorities [1].

In Europe, the European Union Aviation Safety Agency (EASA) has been created in 2003 to ensure that all European citizens can travel in the sky with the highest level of safety. This is an independent and neutral body that ensures confidence in safe air operations in Europe and world-wide by proposing and formulating rules, standards, and guidance; by certifying aircraft, parts and equipment and by approving and overseeing organisations in all aviation domains.

In this context, Safety information plays a fundamental role in the detection of potential safety hazards. Specific Regulations, in particular Regulation (EU) No 376/2014 [2], have been developed in order to improve aviation safety. This requires that relevant civil aviation safety information should be reported, collected, stored, protected, exchanged, disseminated and analysed and appropriate safety action should be taken on the information collected.

In practical terms, this Safety information is disseminated throughout the aviation domain encompassing aircraft manufacturer, airlines, maintenance organisations, Air Traffic Management system, airports and more. At the heart of this process of gathering safety information is the ECCAIRS program [3]. This program aims to capture, share and facilitate analyse of safety data as safety reports from various stakeholders across Europe. Those data are collected and aggregated into a common database.

Additionally, EASA has initiated the Data4Safety programme (D4S) [4] to organise a massive collection of aviation data and manage, through collaborative tools, the analytical capacity amongst all European aviation safety partners. Although, currently in the development stage, this programme should significantly increase the safety intelligence capacity in Europe to anticipate potential issues, proactively identify trends and take timely actions.

Considering the developments of the D4S platform and the current advancements in Large Language Models with the emergence of tools like ChatGPT or BARD, EASA has identified potential enhancements to explore and implement in order to streamline safety analysis. In essence, three components are interacting: data capture via ECCAIRS, data analysis through Data4Safety, and the utilization of artificial intelligence solutions, not yet developed on the platform. It is within this context that the Master's Thesis was formulated.

The Master's Thesis project aims to make a contribution to the field. It seeks to address the complex interaction between occurrence reporting regulations, the Data4Safety database, the data analysis through statistical methods and the innovative capabilities of Large Language Models.

The proposed topic is also a clever blend of the various disciplines covered within the Master of Science in Statistics and Data Science as illustrated on figure 1.1.

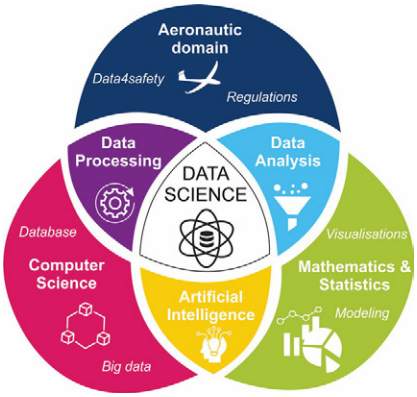


Figure 1.1: Overview of the disciplines addressed by the proposed topic - *Created by author*

1.2 Rationale for conducting this research

The core of the Master's Thesis involves the analysis of data extracted from the safety reports database. In the current development phase of the D4S project [4], collaborative tools have been introduced to conduct safety analyses. However, the safety database presents a significant challenge due to its extensive and heterogeneous data volume. The recent developments in AI technology emerge as a promising solution to handle this challenge.

This opportunity was well identified by EASA [5] in the recently published version of the AI roadmap where it has been determined that AI could provide solutions to deal with D4S data and could offer solution to infer knowledge through:

- Comprehensive Data Understanding: achieved by conducting ad hoc analysis of large amounts of historical data;
- Unveiling hidden correlations: AI's capacity to unveil correlations between different silos of data is amplified through data fusion techniques;
- Enhancing the vulnerability discovery capabilities;
- Anomaly Detection: Through incremental analysis of data flowing into expansive big data architectures, AI serves as a sentinel for anomaly detection.

With the recent progress in Large Language tools, novel techniques have arisen for handling large database. The integration of LLMs could facilitate the extraction of insights from the data and open new perspectives [6], [7]. That's why this research needs to be conducted and has the potential to benefit to the D4S Project.

The scope of this Master's Thesis extends to providing support for EASA experts engaged in safety data analysis and it may also have implications for analysts within the EU Member States.

1.3 Methodology

This research proposes a methodology that integrates Large Language Models (LLMs) with the D4S database to enable natural language querying and response generation.

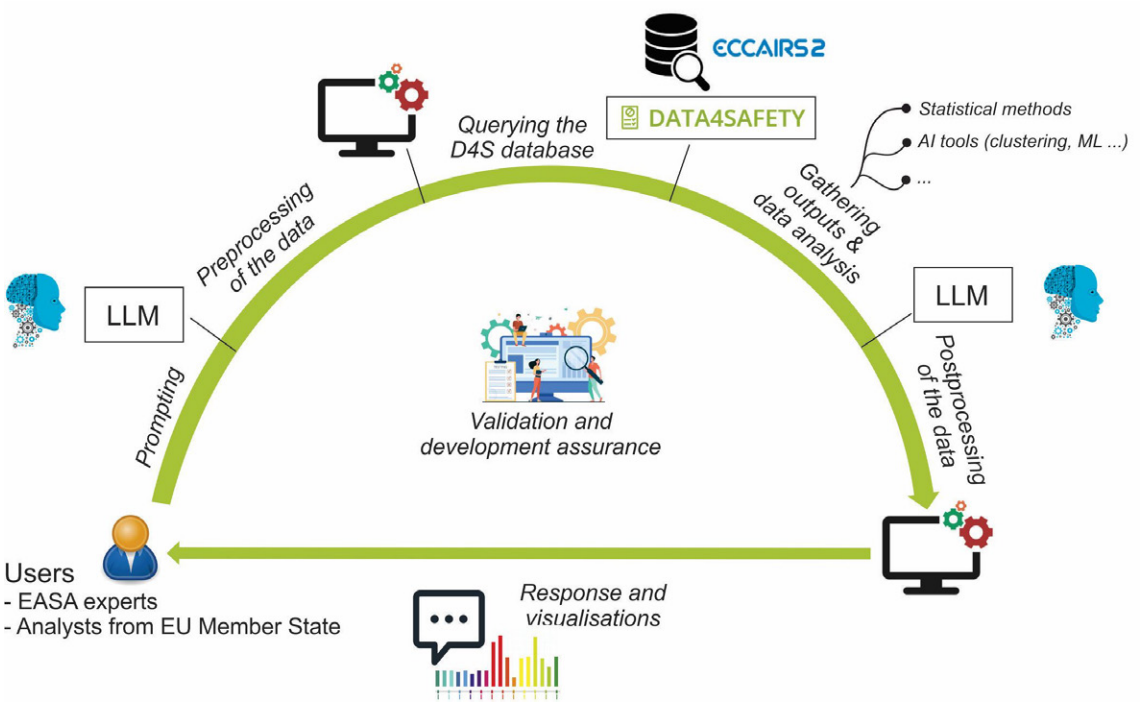


Figure 1.2: Overview of a methodology employing LLMs - *Created by author*

As illustrated on Figure 1.2, the principle is the following: the user submits queries to the D4S platform in natural language. The LLM processes the inputs and queries the database. The resulting outputs are gathered and sent back to the LLM to generate responses in natural language.

In this Master's Thesis, Large Language Models will be implemented in order to process the query provided by the user in natural language. A programming language like Python will be used to interact with existing LLM API [1]. This API will then link to the database and create SQL statements to get the searched data. The output data extracted from the database is mixed with the initial request to guide the LLM's response generation. Other methods or AI techniques (clustering, machine learning...) could be used to analyse the output data but they will be outside of the scope of this work. Dedicated activities are moreover under development in other Agency projects.

While LLMs offer powerful versatile capabilities, they also have limitations. In the context of D4S dealing with Safety data, a thorough exploration of methodologies (fine-tuning, critical analysis...) becomes essential to ensure the production of accurate, complete and meaningful outcomes. A specific part of this work will be allocated to their study. The assurance levels provided by the LLMs will be reviewed in order to ensure that the system performs as expected.

1.4 Objectives and structure

The objective of this Master's Thesis project is to explore large language models for analyzing aviation safety data.

The first step involves an analysis of the Data4Safety database, as detailed in Chapter 2. Specific objectives include identifying the core components of the database, evaluating data quality, analyzing data flow, and detecting potential biases or inconsistencies. Reviewing applicable regulations for occurrence reporting and conducting in-depth studies of the database aims to understand its functionality and structure. Challenges may include data quality issues and complexity of the database structure. The insights gained will aid in implementing the LLMs solution. Key observations and recommendations from this analysis are highlighted for future improvements.

Following this, Chapter 3 presents a literature review focusing on generative AI and Large Language Models. The objective is to review existing models and their underlying architectures, analyse the state of the art in LLM research and identify the most appropriate methods for the analyzing the database. To achieve these objectives, a wide range of models are examined, detailing their characteristics and mathematical representations. Use cases are developed to illustrate their principles and enhance their understanding. The rapidly evolving nature of the field pose the main challenge of this part.

[1] API or Application Programming interface is a set of defined rules and protocols that allow different software applications to communicate and interact with each other.

Chapter 4 transitions from theory to practice by focusing on the development and implementation of LLM solutions tailored to safety data analysis. Considering the industrial constraints, the specific objective is to develop and deploy LLM based systems capable of querying and extracting valuable insights from the database. To achieve this, the chapter outlines the construction of an architecture and AI framework. This involves experimenting with diverse LLM models and architectures to optimize performance and exploring techniques like Text2SQL and fine-tuning. Two AI frameworks are considered: a local implementation and a cloud based solution. Challenges include integrating LLMs with the existing infrastructure, achieving optimal LLM performance and ensuring overall solution effectiveness.

Next, in Chapter 5, the performance of the developed LLM solutions are evaluated and compared. The goal is to asses the accuracy and reliability of the implemented models. To achieve this, the chapter develops an evaluation framework to measure LLM performance against predefined metrics. The results are discussed to gain insights into the implemented models. Validation methods are then reviewed and analyzed, including a learning assurance approach. Adaptations to this approach are proposed, and recommendations are formulated to create a validation method suitable for LLM. The aim is to ensure that the outputs are accurate, meaningful, and trustworthy. A key Challenge is developing an appropriate evaluation framework that effectively compares the developed models.

Finally, the Master's Thesis concludes by outlining potential avenues for future research and improvements based on the insights collected from the study.

Chapter 2

Analysis of the database

2.1 Chapter structure

The objective of this chapter is to study and analyse the database and the data used throughout this Master's Thesis. This analysis is needed for understanding the database's functionality, identifying its structure and exploring the data it contains. Observations and recommendations originated from this analysis are highlighted throughout the chapter to allow future improvements. The resulting knowledge will facilitate the implementation of the LLM solution.

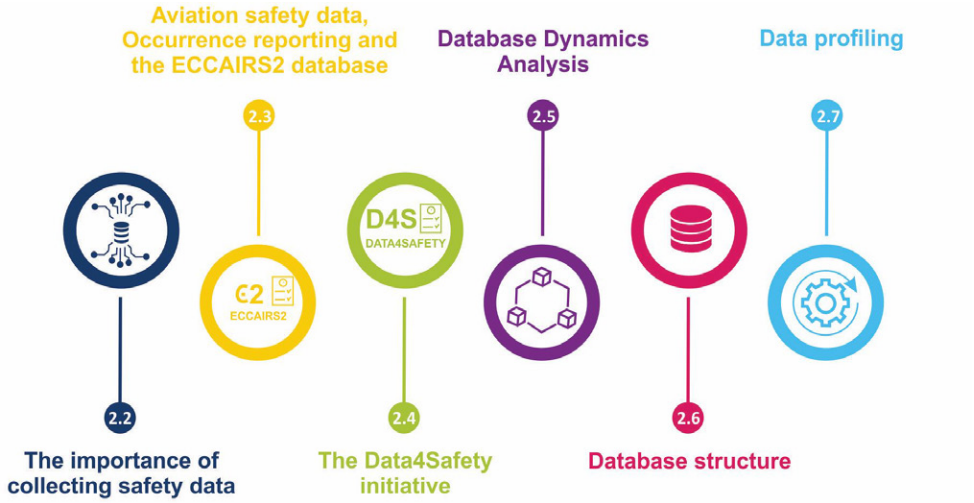


Figure 2.1: Structure of the chapter - *Created by author*

First, it presents the importance of collecting safety data and outlines the framework and context around the database. Then, the analysis delves into the examination of data flows, including the identification of stakeholders and the creation and analysis of data flow diagrams. Further exploration extends to the study of the database structure, essential for understanding the complexities of data organization and facilitating the creation of efficient queries. Finally a data profiling exercise is conducted. This process involves an in-depth analysis of data characteristics, a high-level statistical examination, and an assessment of data quality. The outcomes of this profiling section, conducted and analysed, lead to valuable insights, which are then used to establish a set of recommendations for improving the database (Figure 2.1).

2.2 The importance of collecting safety data

The aviation industry is perpetually evolving, seeking to deliver the most efficient and comfortable flight experience while ensuring the safety of the millions of passengers who travel worldwide each year. Aviation accidents can result in significant number of fatalities and catastrophic consequences, highlighting the need to reduce their frequency especially as an increase of air traffic is expected in long term [8] as displayed on Figure 2.2. In today's aviation safety systems, safety improvements are mainly coming from technological progresses, compliance with prescriptive regulations and lessons learned from aircraft accidents. Yet, experience has demonstrated that many accidents could have been prevented if early warning signs and observations from similar aircraft had been taken into account [8].

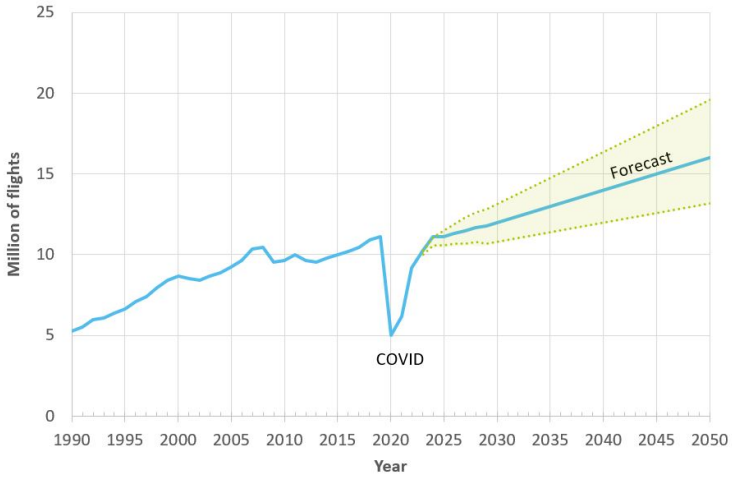


Figure 2.2: Expected increase of air traffic - *Created by author with the data from [9]*

Figure 2.3: Principle of a "proactive" safety system - *Adapted by author from [3]*

To address these challenges, a more proactive system is needed, one that learns from occurrences, risks and hazards as displayed on Figure 2.3. This necessitates a continuous collection of safety information in order to identify, assess and mitigate potential safety hazards [3]. This proactive data collection and analysis should drive advancements in technology, procedures, regulations and address systemic issues.

This need has been duly recognized in the aviation industry worldwide, and the European Union and its Member States have initiated the implementation of a more proactive system. On 3^{rd} April 2014, the European Union has adopted the regulation (EU) No 376/2014 on the reporting, analysis and follow up of occurrences in civil aviation [2], [10]. The regulation focuses on enhancing aviation safety by mandating the reporting, analysis, and follow-up of occurrences within the European Union. The regulation also establishes a structured reporting system to ensure a flow of information across aviation stakeholders, including airlines, authorities, and safety organizations.

2.3 Aviation safety data, Occurrence Reporting and the ECCAIRS2 database

Aviation safety data encompasses a wide range of data related to the safety aspects of aviation. This can include data concerning incidents, accidents, maintenance records, weather data, flight data and various other factors that contribute to ensuring the safety of air travel. These data are collected by various sources and means: incident and accident data are collected through a process of occurrence reporting, weather data are sourced from weather stations and airports and flight data are collected from Air Traffic Management systems [3]... In this Master's Thesis, only the safety data extracted from the occurrence reporting systems will be used.

Focusing on the safety data reported from occurrences through the mandatory reporting system, this information is collected into a database called ECCAIRS2. Individuals fill out dedicated forms provided by aviation authorities at the location where the occurrences took place to populate the database. Figure 2.4 shows the occurrence reporting form provided by the Belgian Aviation Authority[1] as example.

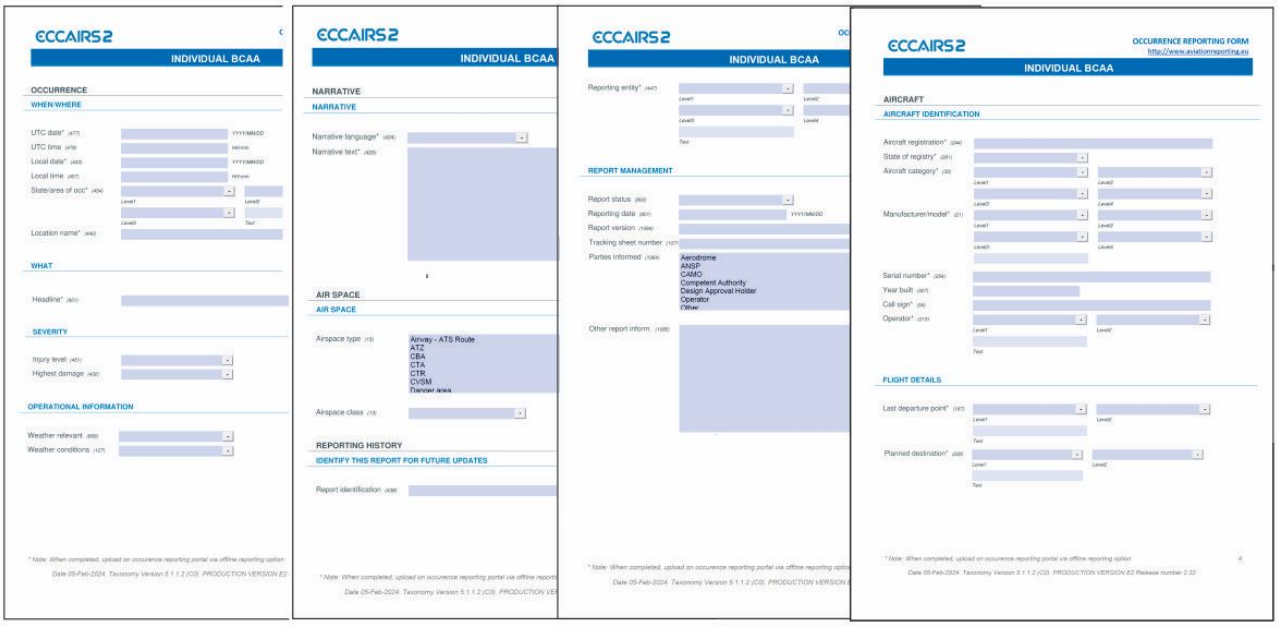


Figure 2.4: Belgian Form to collect safety data from the occurrence reporting - *Extracted from [3]*

The ECCAIRS (European Co-ordination Center for Accident and Incident Reporting Systems) project was indeed created to integrate safety information data from existing sources and provide a solution to Member States that did not have a system in place [3]. This solution was established by the European Commission since 1989 and in 2017, this activity was taken over by EASA. At that moment, the database has been reworked into a web-based and centralized architecture called ECCAIRS2.

[1]The complete form can be consulted on [3].

2.4 The Data4Safety initiative

The challenge to collect the safety data described in the section 2.3 (data from occurrence reporting, maintenance records, weather data...) stems from their heterogeneous nature and from the fact that they are scattered throughout the aviation domain. With recent advancements in Big Data technology, vast amounts of highly diverse data can now be gathered within single Big Data platform [11]. This forms the foundation of the Data4Safety program [4].

Data4Safety is a collaborative and voluntary initiative within the aviation community aimed at sharing and analyzing safety data. This platform constitutes an extensive repository of aviation data continually augmented by the inclusion of data from safety reports, airline flight data, air traffic management system data, weather data, and any other information that enhances the understanding of the aviation landscape. Additionally, this program also organises the analysis capabilities, empowering the EU Member States and Industry to work together on the collaborative analysis platform.

Figure 2.5 illustrates the principle of the Data4Safety platform where the inputs are collected from various data sources in various format. Some possible outputs such as analytics tools, studies, performance assessment are also displayed as proposed to be developed on the collaborative platform.

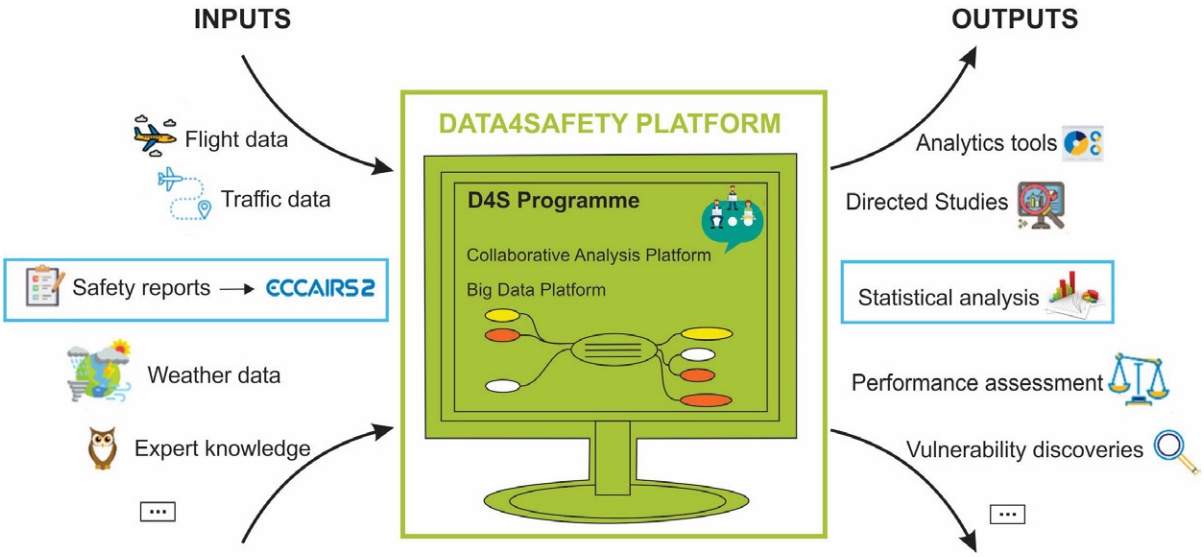


Figure 2.5: Principle of the D4S Programme - *Created by author from [4]*

The input and output used in this Master's Thesis are respectively safety reports sourced from the ECCAIRS2 database and statistical analysis. The focus will be placed on the utilization of LLMs for the analysis of data extracted from the safety reports. The tool developed will be integrated into the Data4safety platform, which is in its preparatory phase with analysis tools still in development. No such AI tool is implemented yet.

2.5 Database Dynamics Analysis

A study of the database dynamics is conducted by analyzing the key stakeholders and the data flow involved within the system. As the Master's Thesis focus on the Safety reports database within the framework of the Data4Safety platform, the analysis is conducted on both the ECCAIRS2 database and the Data4Safety platform.

2.5.1 Stakeholders

The main stakeholders associated with the ECCAIRS2 database include any entity (individuals and/or groups) that has an interest or is involved in either using the database or developing tools to analyse it. These stakeholders were identified by studying the ECCAIRS2 database [3], [12] and are explained hereafter.

- **EASA**: The agency is a key stakeholder as it is responsible for coordinating the aviation safety in Europe. The agency uses the ECCAIRS2 database for the collection and analysis of safety-related data. ECCAIRS2 is also used in research programs developed by EASA.
- **National Aviation Authorities (NAAs)**: They actively receive and assess occurrence reports, which form the foundation of the ECCAIRS2 database. NAAs may also utilize the database as a resource to develop their national safety initiatives.
- **Regulatory Authorities**[2]: Involved at national or international level, they use the database to monitor and improve regulatory compliance.
- **Aircraft Operators, Manufacturers, Maintenance Organisations**: They provide and utilize safety-related data to enhance their operations and products.
- **Air Traffic Management (ATM) Organizations**: Entities responsible for managing air traffic are also stakeholders. They may contribute by providing data related to incidents in airspace and airports.
- **Pilots and Aircrew**: By their direct involvements in operational activities, they contribute to the database by reporting firsthand incidents and accidents. The ECCAIRS2 data enables them also to learn from the past, shaping improved training and operational procedures. The feedback loop provided by the database allows pilots and aircrew to understand the impact of their reports on overall safety initiatives.
- **Researchers and Analysts**: Professionals involved in aviation safety use the ECCAIRS2 database to study trends, identify patterns, propose safety improvements or develop specialised tools for data analysis.

[2]These authorities are tasked with overseeing and regulating various aspects of civil aviation safety. Examples include the International Civil Aviation Organization (ICAO) and the Directorate-General for Mobility and Transport (DG-MOVE) at the European Commission.

- **Developers**: As the ECCAIRS2 database is involved in the data4safety program, developers may interact with the database to create and develop tools to analyse data.

At the level of the Data4Safety Platform, additional stakeholders are related to the system [4]. This includes:

- **Third party Service Providers**: as contractors, they provide support, tools and service to develop and maintain the database. The ECCAIRS2 database is one of the various flow of data collecting by the Data4Safety platform. The data are converted by the service provider into a SQL model that can be used by the users.
- **Members of the Data4Safety initiatives**: they can use and interact with the Big Data platform including the data from the ECCAIRS2 database.

2.5.2 Data flow diagram

This section delves into the data flow process, explaining how data is initially generated and the different steps it undergoes before ultimately contributing to aviation safety.

The ECCAIRS2 database

A high level data flow diagram of the ECCAIRS2 database is created as shown on Figure 2.6 [3].

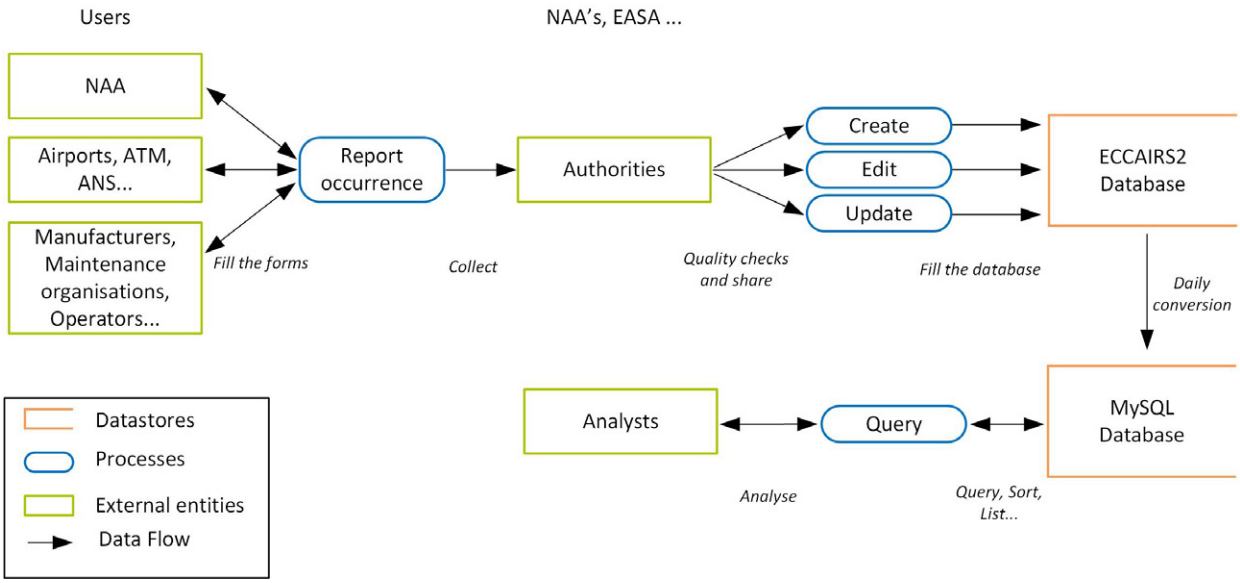


Figure 2.6: Data Flow diagram - *Created by author from [3]*

Three primary external entities play key roles in this process: a group of *Users* (including NAAs, Airports, ATM, Air Navigation System (ANS), Manufacturers, Operators, Maintenance Organisation, inspectors...) responsible for reporting occurrences when identified, *Authorities* (EASA and NAAs) that collect, perform quality checks and share the data in the central ECCAIRS2 database and the *Analysts* who monitor safety risk portfolios and develop reports about safety issues.

The central hub of data within the system is the ECCAIRS2 Database serving as the main datastore. To facilitate interactions with the database, a Structured Query Language (SQL) model has been developed. This secondary datastore is updated daily with data from the ECCAIRS2 database.

In terms of flow, everything begins when a *user* needs to report an occurrence. They are required to complete a dedicated form on the ECCAIRS2 reporting portal. Depending on the Aviation Authority State to which the *user* is affiliated, the form varies. There are 32 distinct forms, each customized according to the Member Sate where the report is submitted. Once the *user* has fill out the form with the necessary data, it is sent to the dedicated aviation authority. The form includes all the data necessary to create a valid entry in the ECCAIRS2 database and specifies the taxonomy to be followed.

In each Aviation Authority, the received data undergoes a review and check, often performed manually. Additional taxonomy is applied and then the data is shared on the ECCAIRS2 central database. The ECCAIRS2 database is synchronised with the SQL database every night so that both databases are identical. Through the SQL database, analysts can easily query and conduct various types of analyses.

Based on the findings and analysis conducted, safety action plans can be defined to enhance global aviation safety. These plans are deployed at national levels and at European level. Furthermore, a continuous monitoring system to assess the safety performance can be further developed and expanded by using the findings gained. Regularly updating and enhancing this monitoring mechanism will enable the system to stay proactive in identifying potential safety issues and adapting the safety plans accordingly.

The Data4Safety Platform

The occurrence reports gathered in the ECCAIRS2 database contribute to a Big Data repository hosted on the Data4safety platform. These reports serve as one of the diverse sources of data utilized within this platform. A simplified representation of the data flow is created and illustrates the processes involved. In this simplified data flow diagram (Figure 2.7) three external entities are identified. One service provider is tasked with the responsibility of collecting the data from diverse sources such as Occurence reports from ECCAIRS2, weather data or flight data ensuring their maintenance. A service provider is also designated to process these data involving tasks such as formatting of the data or insertion in a standard model (SQL...) to enhance their usability. The resulting data are then stored in the Big Data platform facilitating interactions, utilisation and analysis by the members.

Two databases are used as data stores:

- The data collection platform serves as an internal database responsible for collecting and preparing the data;
- The Big Data platform functions as a database where all the records are stored enabling the members to access and use the data.

A simplified data flow diagram of the Data4Safety platform is created as shown on Figure 2.7 [4].

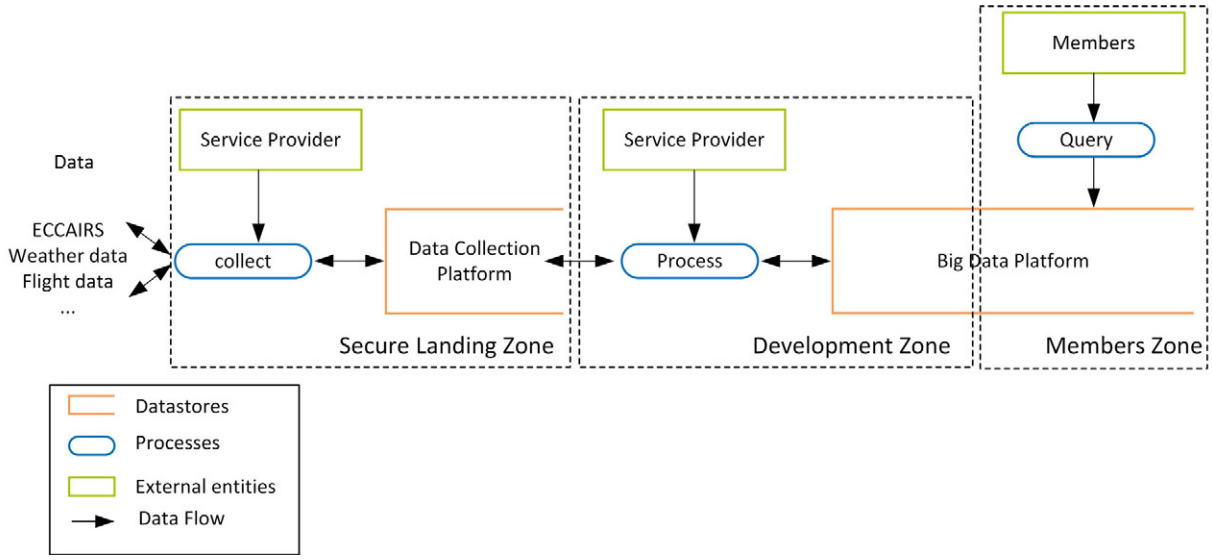


Figure 2.7: Simplified data flow of the Data4safety platform - *Created by author from [4]*

In terms of data flow, various sources are collected in the secure landing zone through the data collection platform. Subsequently, pre-processing is applied to format and convert the data. The processed data is then stored in the Big Data platform, where members of the Data4Safety platform can access and analyze it.

2.5.3 Observations and recommendations

From the data flow process (Figures 2.6 and 2.7), several observations can be made:

- Safety data related to a reporting occurrence takes time before being uploaded and published into the ECCAIRS2 database (several weeks to several months).
- The reporting of occurrences relies mainly on the willingness of users. However, the system depends on the quality and availability of data. If users, despite legal constraints, do not report each identified event, the system will not have a comprehensive view, and certain findings may go unnoticed.
- The fact that data quality control and taxonomy implementation are carried out by different Member States can lead to standardization issues.

Based on these observations, two recommendations are suggested:

- The implementation of a common interactive form incorporating extensive choice lists would improve data quality by ensuring the implementation of the defined taxonomy. This would also optimize data checks and validation by member states.
- Implementing an AI tool to verify user-provided data could significantly enhance data quality and streamline the tasks of Member States. Such a tool could be deployed not only to assess incoming data but also to analyze existing data in the database, thereby improving and completing missing fields.

The implementation of these recommendations would facilitate the integration of the LLM solution by improving data quality and streamlining verification and analysis processes.

2.6 Database structure

An overview of the database structure is presented. Practically, the occurrence data is not directly taken from the ECCAIRS2 database but they are extracted from a replicated database utilizing MySQL technology. The information is extracted regularly from the ECCAIRS2 platform and are transformed into a relational model to ease its use and accessibility. From the ECCAIRS2 database, different entity relation diagrams have been created and are analysed. The python code used to conduct this analysis is referenced in Appendix B and allows to create the Figures 2.8, 2.9 and 2.10.

Figure 2.8 shows the different entities of the model. They are organised around the central entity *Occurrence*. Each entity has various attributes which are not represented to facilitate the understanding of the general structure. They also includes 2 keys: one to uniquely identifies a specific instance of an entity and the other one to connect the entity with its parent.

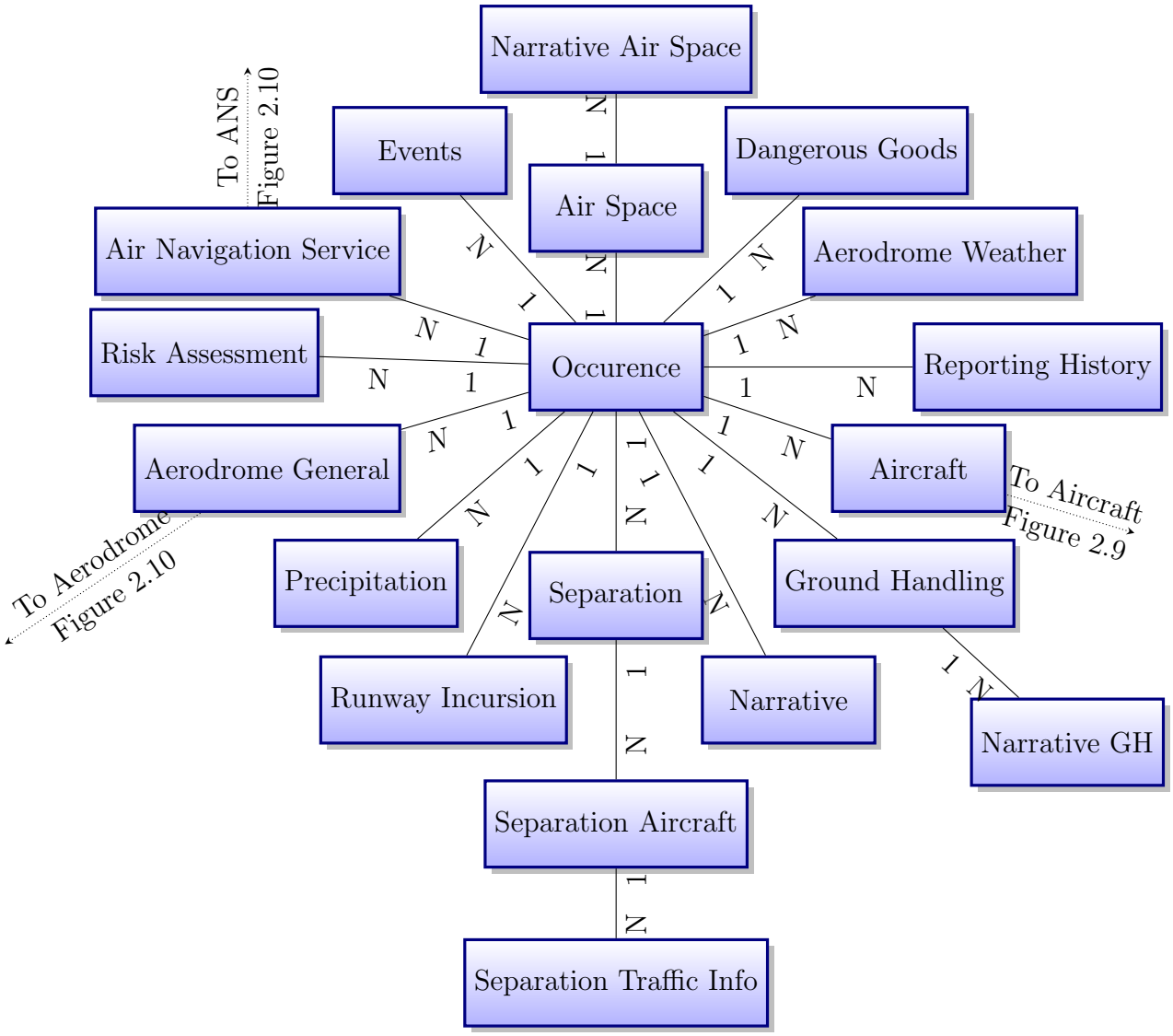


Figure 2.8: Entity relation diagram - Occurence - *Created by author (code Python: A1)*

There are a total of 33 entities, of which 14 are directly connected to the *Occurrence* entity. This central entity encompasses in addition 79 attributes. These attributes store details associated with the occurrence, spanning from the reporting entity, and basic weather information to the occurrence class date and location. They also include information on the number and types of injuries or fatalities as well as the severity and repeatability of the occurrence.

Additional details are consolidated and included within the 14 associated entities. Amongst these entities, four entities are quite broad:

- Risk Assessment: the 25 attributes of this entity provides the details on the risk level associated with the occurrence.
- Narrative: the 4 attributes of this entity provide the full narrative text provided by the occurrence reporting organisation.
- Reporting History: this entity provides in its 27 attributes the history of the reporting occurrence detailing the status, date and reporting entity.
- Events: this entity and its 10 attributes provide the event type and the phase of the related occurrence.

Two entities are related to the aircraft itself and some potential transport of dangerous goods.

- Aircraft: comprising 86 attributes and complemented by 7 additional entities, this entity provides details pertaining to the aircraft. These details encompass specifics on the engine and propeller (if applicable), flight crew members, aircraft type, destinations, installed options, aircraft age, available instruments, operation type, destination, registration, and passenger count, among others. Figure 2.9 provides the structure of the entity.
- Dangerous goods: this entity and its 15 attributes describe, if applicable, the type, origin and the volume of dangerous good transported during the occurrence event.

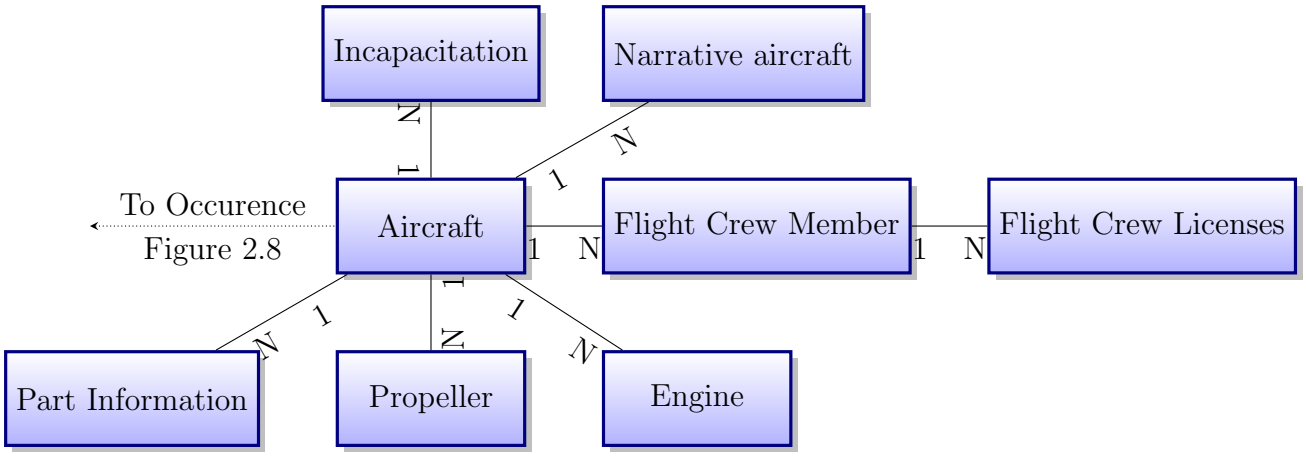


Figure 2.9: Entity relation diagram - Aircraft - *Created by author (code Python: A1)*

Four other entities are related to air space and traffic management involved in the occurrence event:

- Air navigation service: this entity with its 6 attributes and 4 related entities as described on Figure 2.10 provides the details of the air traffic management conducted during the event including the ATM staff involved and the recording device used.
- Air space: this entity and its 8 attributes describe the name, class and type of airspace related to the reported event.
- Separation: this entity provides 11 attributes and 2 complementary entities to describe, if applicable, the vertical and horizontal separation distances between the aircraft involved in the reporting and the corresponding actions taken to avoid any incidents.
- Runway Incursion: this entity has 5 attributes to indicate if the occurrence reported is associated with a runway incursion.

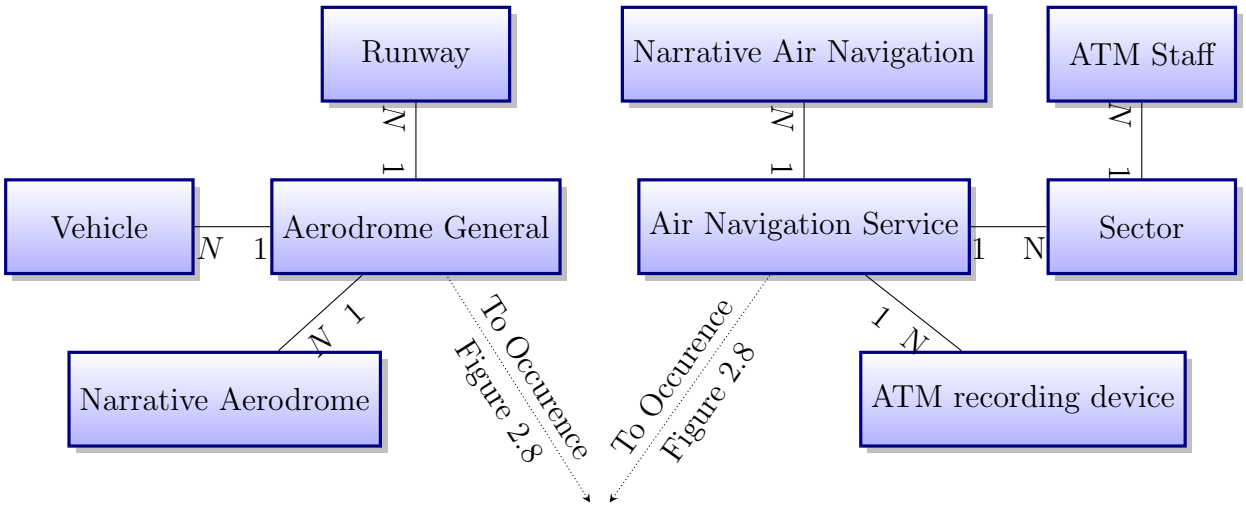


Figure 2.10: Extract of the entity relation diagram - Aerodrome General and Air Navigation Service - *Created by author (code Python: A1)*

The four last entities are related to the aerodrome, its operation and local weather.

- Aerodrome General: the information related to the aerodrome included the details on the runway, vehicles involved if any are included in the 15 attributes and the 3 complementary entities as presented on Figure 2.10.
- Ground Handling: the entity has 13 attributes and provides details about the ground services, vehicles and staff involved if any. The entity is completed by a narrative entity that provides the full text of the ground handling service report.
- Aerodrome Weather: the relevant details of the aerodrome weather are included in 4 attributes associated with this entity.
- Precipitation: the 4 attributes of this entity provide details on the precipitation type and intensity present during the reported events.

2.7 Data profiling

Data profiling involves exploring the data stored in the ECCAIRS2 database through three distinct approaches: examining the characteristics of the data, conducting high-level statistical analysis, and performing data quality assessments. This process helps to gain insights into the distribution and quality of the data, providing a comprehensive view of the data and an initial experience for further exploration and analysis. The knowledge gained will provide some in-depth insights into the data which will be valuable for future use in the developments of LLM tools.

2.7.1 Characteristics of the data

Initial insights

To gain initial insights into the data, feedbacks provided by users are examined. These reviews were gathered by the European Commission as part of their ex-post evaluation of Regulation 376/2004 [13], conducted between 2014 and 2019. Although the details of this evaluation exceed the scope of this section, these offer valuable perspectives on the database and its overall quality. From this ex-post evaluation exercises, three main concerns are listed and discussed:

- A primary concern identified arises from gaps in the quality and completeness of reports contained in the database. These gaps result from different approaches to quality checking and data transfer. The substantial volume of occurrences, coupled with the detailed reporting requirements outlined in the Regulation and its Annexes, imposes a significant workload on competent authorities to collect, process and store occurrences notably related to the fact that completeness and quality checks are performed manually. This concern of completeness of the reported data will be further studied during the data profiling.

- Another aspect highlighted in the report is the extensive list of mandatory data fields included into the database. Not all of these fields are relevant to every reporting organisations, leading to a potential lack of voluntary reporting. Many organisation tend to focus their efforts due to limited resources on compliance with the extensive requirements for reporting on mandatory occurrence types. Moreover, the multitude of data fields further complicates the collection of data, checking process and analysis work.

- Finally, the collection and analysis of data through the implementation of Regulation 376/2014 have indeed led to a noticeable enhancement in reporting occurrences. However, it also underscores that the heterogeneity of the data and its sheer volume pose significant challenges that needs to be overcome to generate valuable findings and ultimately improve overall safety.

Generalities

The analysis of the database structure revealed the presence of no less than 33 entities, each containing a multitude of attributes. Examining the characteristics of each isolated attribute individually is impractical and of little relevance. Therefore, this section will focus on studying two main entities, namely *Occurrence* and *Aircraft*. While the organization and characteristics of attributes in other tables may differ, they exhibit marked similarities with the 2 attributes under consideration. At the time of this analysis, there are $3,395,848$ observations present in the database. The codes used to perform the analysis is referenced in Appendix A2 and A3. The obtained results are discussed hereafter.

The first entity under investigation is *Occurrence*. This entity is the central piece of the database. It contains 79 attributes. Among these, 17 are numerical values, 4 serve as unique identifiers or serial numbers, 2 are binary value (True or False), 5 are associated with date or time data and the remaining items consist of text fields.

The observations are analyzed to identify potential missing data. The Table 2.1 displays the attributes from *Occurrence* with the highest number of missing values. Given the total number of observations, the count of missing data is truly negligible. However, they can impact certain analyses or machine learning processes, therefore considering a strategy to manage this type of data may be worthwhile.

Attribute	Number of Missing data
Location name	206
Longitude of Occurrence	17
ATM Contribution	5
Light condition	4
UTC Date	4
Headline	3
Damage to aircraft	2
Total Minor Injuries	2

Table 2.1: Attributes with the highest number of missing data - *Created by author (code R: A3)*

A comparable analysis can be conducted on the *Aircraft* entity. This entity contains 86 attributes among which 8 are numerical values, 4 serves as unique identifiers, 3 are date or time values and the other items consist of text fields. Various attributes are divided into multiple fields. The number of missing data is also negligible in comparison to the number of observations.

Numerical attributes

Using the codes referenced in Appendix A2 and A3 an analysis is conducted on the 17 numerical attributes.

It can be observed that they exhibit numerous entries labelled as *None*. This indicates that the value for the respective attribute is not provided or is not available. Across the entire set of observations, nearly 85% of the data for each attribute is identified in this manner. While for certain attributes, the information may not be relevant to the reported event, for others (such as the total number of passengers or the number of injuries ...), a null value would be more appropriate.

The range and distribution of available values for numeric attributes can also be studied. As an example, the distribution of *Visibility* values in meters is depicted in Figure 2.11.

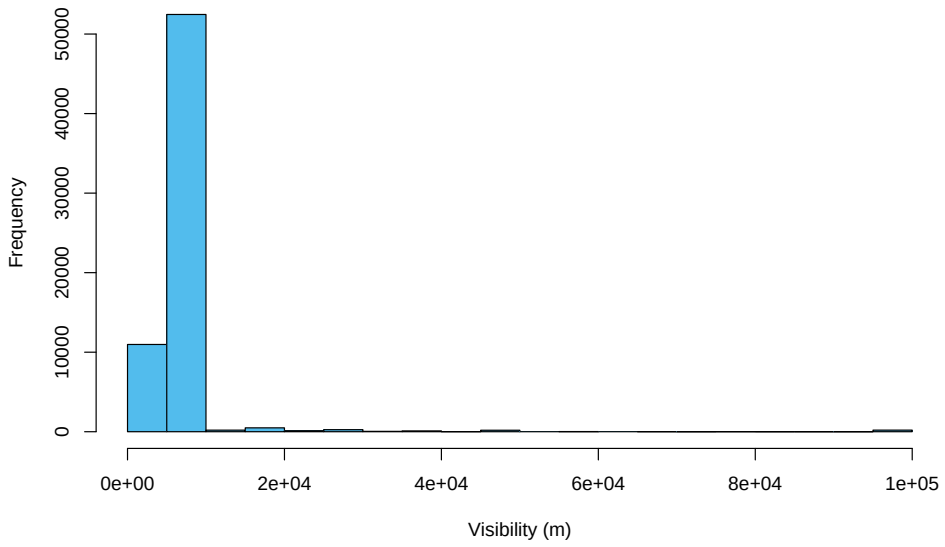


Figure 2.11: Histogram of the attribute *Visibility(m) - Created by author (code Python: A2)*

The histogram reveals that the majority of visibility values, when available, fall within the range of 0 to 10,000 meters, which is representative of typical visibility ranges in aviation. Conversely, visibility values of 100,000 meters are not typical in aviation and could be considered as errors or anomalies. These extreme values should, therefore, be analyzed carefully to verify their accuracy. Implementing a range constraint during data entry could prevent the introduction of erroneous data and draw the reporter's attention to such extreme values at the point of data entry into the database.

Similarly, an analysis of outliers reveals a certain number of them in the values of the numerical attributes. Without more information about each of these values, they cannot be excluded from the data, especially considering that the data entered into the database may be related to atypical events with extreme values. A careful analysis of these points must therefore be conducted before utilizing this data.

Comparable analyses can be carried out on the attributes with numerical values within the *Aircraft* entity. Similar conclusions can be drawn from these analyses.

Attributes with text values

By exploring the database and using the codes referenced in Appendix A2 and A3, the attributes with text values are analysed and various results are discussed.

Attributes with text values are highly diverse, encompassing lists of names (country names, manufacturer, sector...) as well as short expressions describing situations or entities (e.g., *Occurrence class*, *State area of occurrence*...). Some of these attributes are divided into multiple fields to allow for the entry of long text field. While taxonomy is well-established, certain attributes exhibit duplications. This is namely the case of *reportingEntity*, where the attribute is quite broad and flexible. Notably, similar entity names may appear in different languages.

In the text attributes, there are also discrete nominal variables that indicate the status, occurrence type, or technical severity. These variables assume distinct categories or labels (e.g., *Open*, *Close*, *Pending*) without any inherent order or ranking among them. These variables represent qualitative data, offering a quick overview of a specific parameter.

Another entity that contains only text values is *Narrative*. This type of entity encompasses 5 attributes including a unique identifier, the identifier of the parent entity associated with the narrative, the language of the narrative and a free text field containing the narrative text. This text field comprises raw text describing in details the occurrence and the corresponding reported findings. There are 6 narrative entities corresponding to aerodrome, air Navigation service, air space, aircraft, ground handling and Occurrence. Unlike the other data elements that provide detailed information on for example location, weather, aircraft and the events themselves, the narratives provide a storyline for the events.

Observations

To conclude, this short data profiling highlights a significant degree of heterogeneity and complexity in the dataset, encompassing mainly numerical and text values. The tables contain a substantial number of attributes, facilitating detailed storage of information but also resulting in numerous empty fields for each record. The numerical data exhibit a low number of missing values but potential outliers, necessitating careful study and analysis before utilizing the data. The six narrative entities offer a descriptive supplement to the database details. However, the availability of these narratives in different languages through the database introduces complexity to their analysis.

2.7.2 High level statistical analysis

To pursue the database analysis, a high level statistical analysis is conducted. The objective is to engage with the database and validate the presented previously findings. This also serves as an opportunity to draft SQL queries and Python code for extracting data. Those codes could be utilized in the implementation phase to refine the LLMs algorithm. Following data extraction, R code is employed to perform basic analyses. The python codes, SQL queries and R codes are referenced in Appendix A2 and A3.

The statistical analysis initially focuses on the main table *occurrence*. A Python script queries the database using SQL code to determine the number of data entries. The query results in 2,788,219 observations[3].

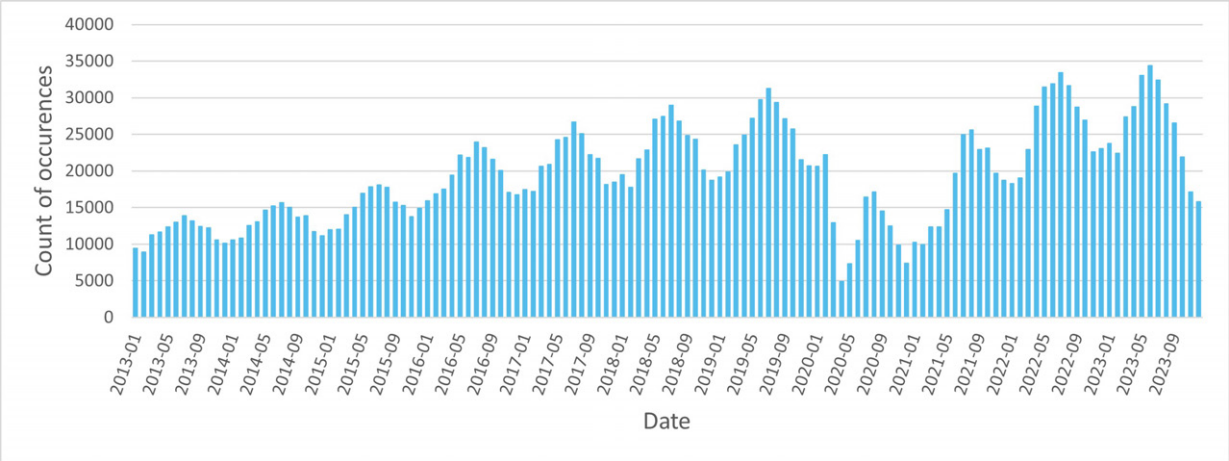


Figure 2.12: Numbers of reported occurrences per month over the last 10 years - *Created by author (code R: A3)*

Figure 2.12 illustrates the monthly distribution of the reported occurrences over the past 10 years. Notably, a consistent peak is observed during the summer season, aligning with increased aeronautical activity during that period. The impact of the Covid-19 pandemic is also well visible in the plot, contributing to large fluctuations.

Furthermore, an interesting trend emerges as the total number of reported occurrences annually shows an increase post-2014. This temporal shift coincides with the implementation of updated reporting regulations.

Next, the distribution of the occurrence responsible entity is assessed and presented in Figure 2.13. It indicates that the majority of reported occurrences are allocated to the European Commission (EC). Additionally, it can be seen that the Netherlands, Denmark, and Ireland emerge as the top three Member States (MS) with the highest number of allocated reported occurrences. Figure 2.14 presents the numbers of reported occurrences per country in Europe. France and Germany are two main contributors.

[3]The analysis was realized in December 2023.

These observations can be seen on Figure 2.13 and on Figure 2.14.

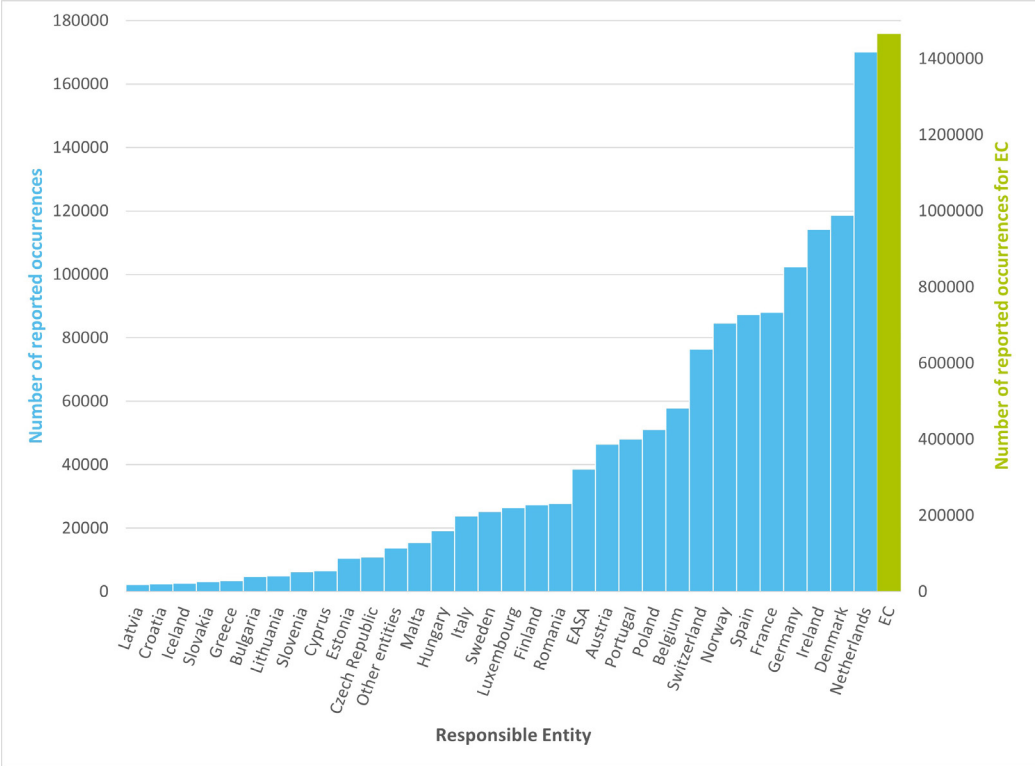


Figure 2.13: Distribution of the occurrence responsible entity per country - *Created by author (code R: A3)*

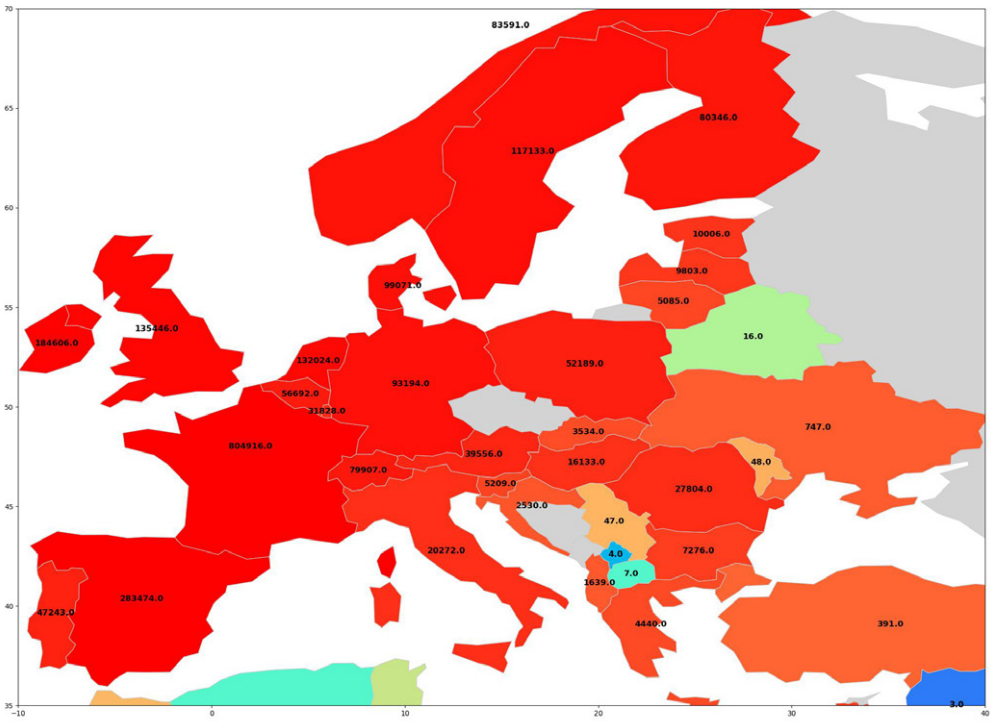


Figure 2.14: Reported occurrences per country in Europe - *Created by author (code R: A4)*

By analyzing the attributes of the occurrence table, a statistic corresponding to the proportion of each type of occurrences can be extracted.

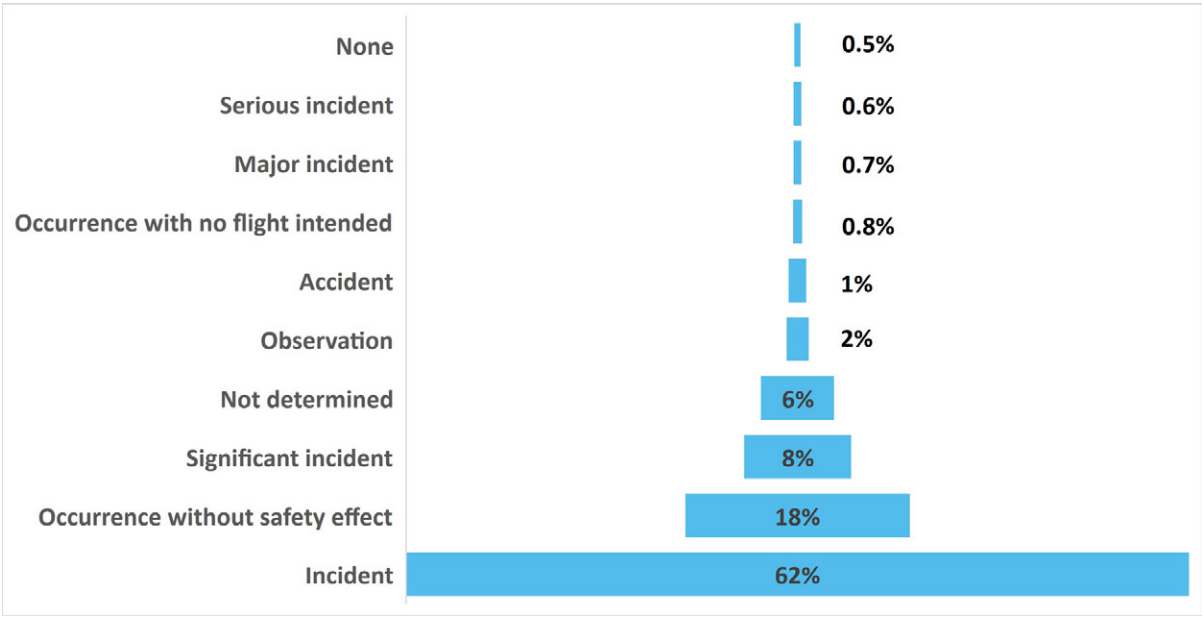


Figure 2.15: Distribution of occurrence types - *Created by author (code R: A3)*

The Figure 2.15 reveals that incidents make up about two-thirds of the reported occurrences, with the cumulative occurrence of the most severe types (Accident, Major incident, Serious incident and Significant incident) totalling 11 percent.

Another interesting statistical analysis involves identifying the sources of reported occurrences based on different reporting entities. Figure 2.16 illustrates the percentage distribution across these groups.

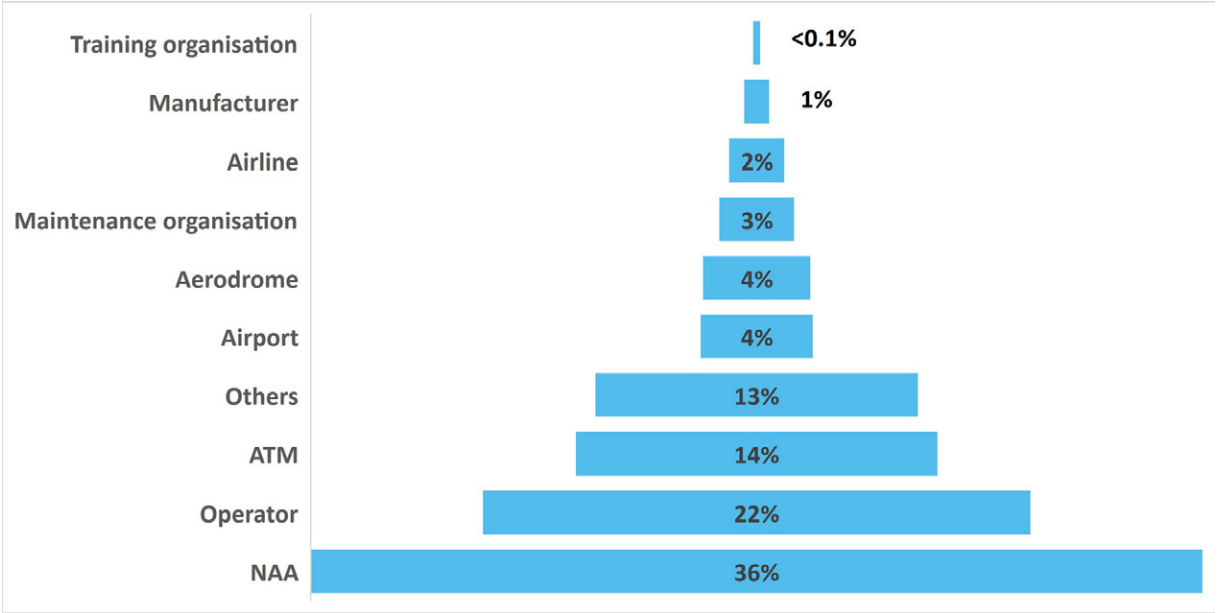


Figure 2.16: Reported occurrence sources distribution - *Created by author (code R: A3)*

It can be seen that the NAA is the predominant reporter, accounting for 36% of occurrences. The operators contribute around 22%, and air traffic controllers report approximately 14% of the occurrences.

Finally, by combining data extracted from both the aircraft table and occurrence table, it becomes possible to plot the evolution of reported injury types over time. The interpretation of the Figure 2.17 is not straightforward and may involve further analysis to determine the observed variations in the number of reported injuries.

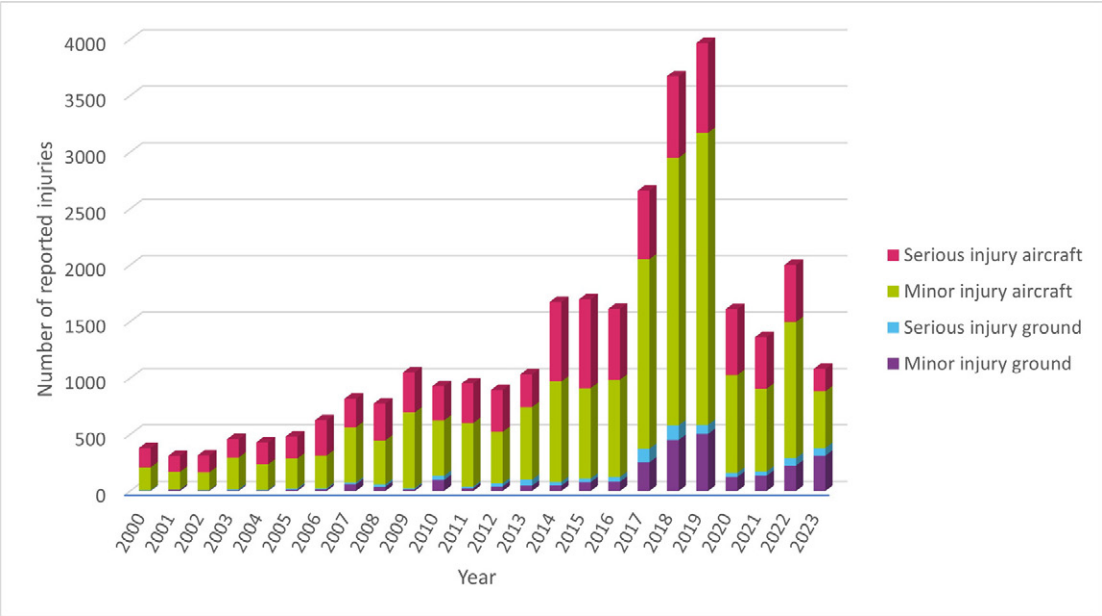


Figure 2.17: Evolution of reported injury types over time - *Created by author (code R: A3)*

This high-level statistical analysis leads to the conclusion that extracting and interpreting data from the database is not a straightforward task. It typically involves gathering information from multiple tables, and the interpretation requires additional background information that may not be directly available in the database. Addressing this challenge will be crucial during the exploration of the implementation of a LLM solution.

2.7.3 Data quality assessment

In this section, the characteristics of the previously described data are evaluated to gain insights into the overall quality, reliability, and effectiveness of the database in meeting its intended purpose.

Findings and observations

The database is analyzed on criteria used in software quality control [14] and the findings listed against those criteria are presented. They represent observations made through the use of data, queries and the results of the analyses conducted previously. They may not encompass all aspects.

Figure 2.18 presents the findings.

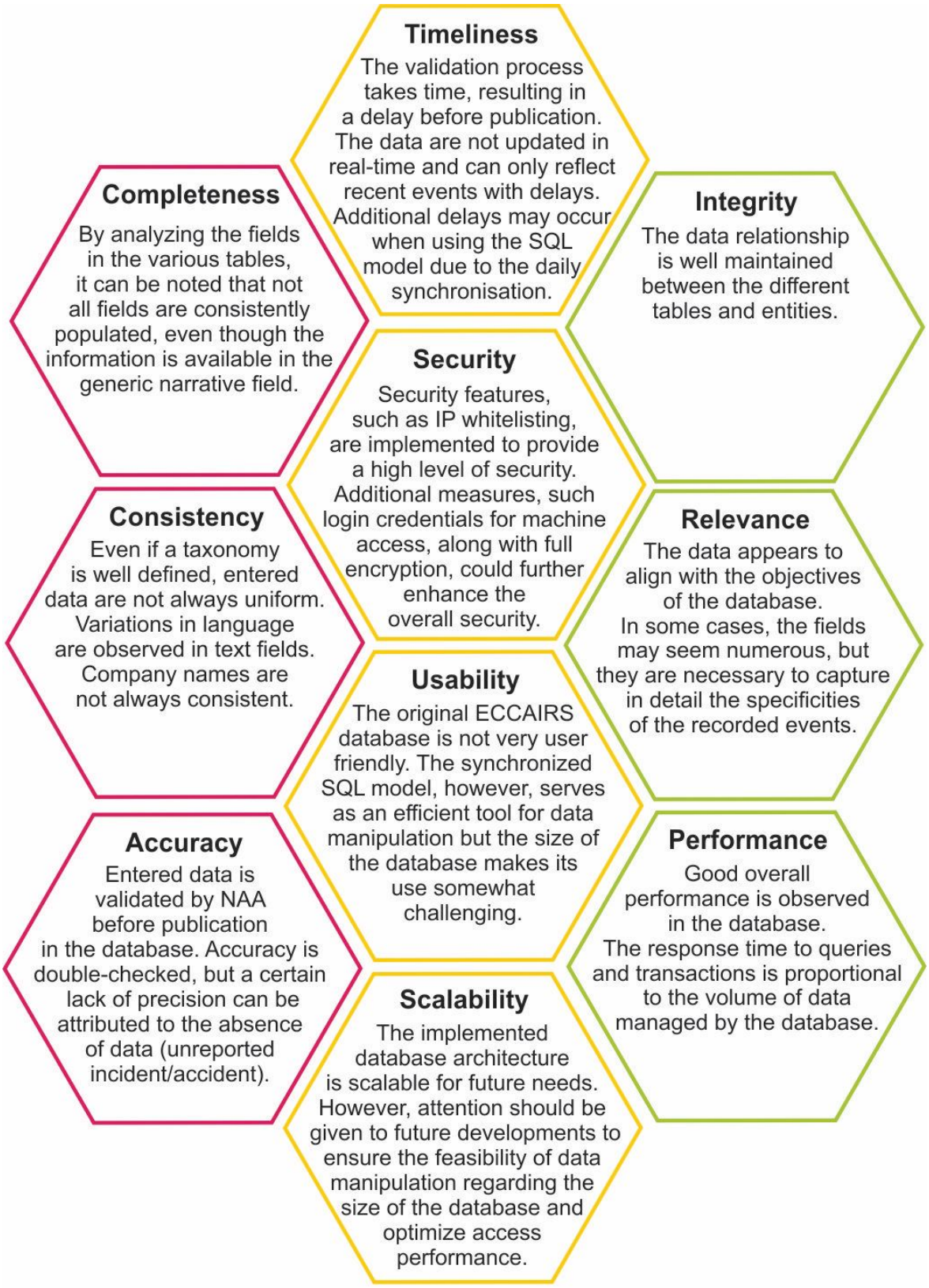


Figure 2.18: Quality assessment of the ECCAIRS2 database - *Created by author from the analysis conducted in chapter 2*

Recommendations

To enhance validation time, an interactive form incorporating already a set of attributes and their corresponding taxonomy should be provided at the time of data submission in the NAA. While the inclusion of multiple attributes allows for the introduction of detailed information into the database, numerous fields may remain empty. Hence, it could be advantageous to designate mandatory fields that must be completed systematically for all reporting occurrences, with optional attributes serving to complement the defined information.

Several inconsistencies have been identified. To avoid these and minimize the risk, interactive forms with predefined lists could prevent the generation of duplicates in names. Likewise, translating all fields into a single language within the database would help avoid redundancies and facilitate analysis.

The lack of data or imprecision significantly impacts the potential analyses. Certain fields should be identified as mandatory to ensure a minimum quality of the introduced data.

The complexity and multitude of attributes and entities make the database complex and challenging to manipulate easily. The development of specialized tools such as machine learning, AI tools or Power BI should be considered to enhance analytical capabilities and provide assistance to improve data quality in the database.

Regarding scalability, a substantial number of observations are added each month. To allow for the future development of the database and maintain its utility, a mechanism for archiving older data should be considered. The data would not be lost but stored in a secondary database, enabling easier use of the most recent data.

Chapter 3

State of the art

3.1 Chapter structure

This section explores the world of Large Language Models (LLMs), examining their current status, capabilities, and limitations. Recent research has significantly expanded our understanding of LLMs, revealing a wide array of applications and methodologies across various fields. Figure 3.1 illustrates the structure of this chapter.

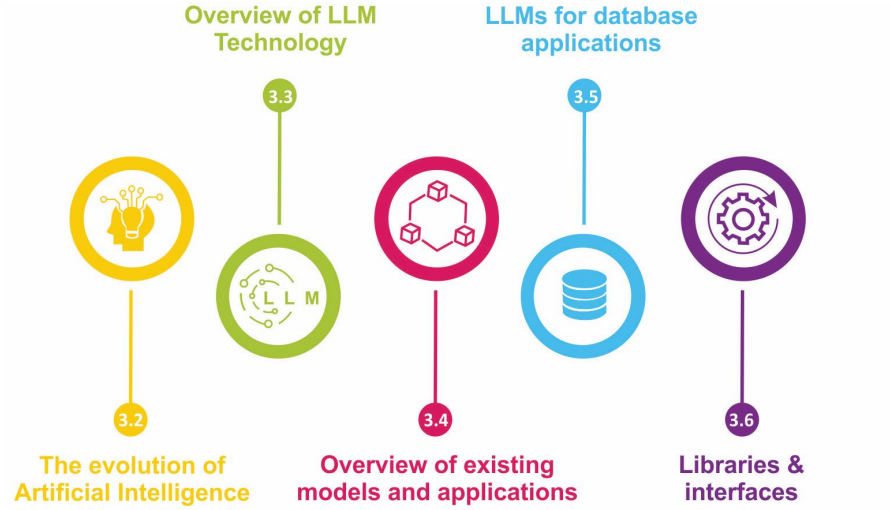


Figure 3.1: Structure of the chapter - *Created by author*

Beginning with a concise historical overview of AI and generative AI, the technology behind LLMs is introduced including mathematical abstractions and use cases. Future challenges and directions are also discussed. The evolution of different LLM models and their various applications is then explored. This exploration is followed by an investigation of different implementations of LLMs interacting with databases, focusing on common challenges encoutered in such interactions. Similar approaches could be followed for this Master's Thesis. The review encompasses examining and discussing options for accessing and utilizing LLMs, including libraries and APIs.

3.2 The evolution of AI

3.2.1 A short history of AI

Artificial Intelligence (AI) covers a vast set of techniques and a wide range of applications. Recent advancements in IT have led to a renewed surge in this domain with numerous researches. Despite the diversity of AI research, the core concept is the same : the creation of intelligent agents that can sense their environment and take appropriate actions to achieve specific goals. This basic idea is not new and have evolved over the past fifty years within the domain [15].

Historically, the birth of AI is widely attributed to the Dartmouth workshop in 1956. It was during this event that the field gained its name, its missions and first successes. The domain then underwent an initial phase of substantial investments, but it quickly became evident that the difficulty of the project had been significantly underestimated.

Faced with these challenges, investments were redirected into other areas and progress slowed for a time. AI developments experienced a resurgence with the massive investments made namely by the Japanese government in the 1980s. AI evolved into an industry promising expert systems, robots and specialized software. However, once again, investors became discouraged due to the slow pace of progress and the limited visible advancements in the field.

In the 1990's, AI began to find successful application in the technology industry. The "intelligent agent" paradigm[1] became widely accepted and new tools, such as Bayesian networks[2] and hidden Markov models[3] were also developed. It wasn't until the early decades of the 21^{st} century that new methods successfully addressed various challenges, leading to renewed substential investments in the field. The availability of large data sets, along with more affordable and faster computers, as well as advanced machine learning techniques like deep learning, were successfully employed to tackle numerous challenges.

These three milestones represent significant developments in the history of AI, with each era contributing to the progress and evolution of artificial intelligence. AI continues to advance rapidly, with new breakthroughs and milestones occurring regularly in different domains. Recently, one of these fields has gained significant attention: the generative AI with ChatGPT's breakthrough moment.

[1]The paradigm refers to a conceptual framework in artificial intelligence where an "agent" operates within an environment, perceiving its surroundings and taking actions to achieve specific objectives.

[2]A Bayesian Network is a model that represents variable and their interdependencies using an acyclic graph incorporating Bayesian inference for reasoning under uncertainty.

[3]A Hidden Markov Model is a statistical tool that models sequences of observed data assuming an underlying sequence of hidden states that generate the observed information.

3.2.2 Evolution of Gen-AI and Natural Language Processing

Throughout the history of artificial intelligence, the aspiration has always been to enable natural and seamless communication with machines or robots [15]. Generative artificial intelligence is tasked with developing models known as Artificial Intelligence Generated Content (AIGC), capable of generating content, not limited to text but also encompassing images, music, videos, and more. This transformative technology harnesses the capabilities of generative AI to revolutionize content production across industries, enabling businesses to automate content generation, enhance user experiences through personalization, and create innovative and engaging multimedia content at an unprecedented scale [16], [17]. To manage interactions between computers and human language, Natural Language Processing (NLP), a subset of AI, specifically addresses these interactions. NLP focuses on understanding, interpreting, and generating human language. When combined with generative AI, these two domains have led to the development of powerful models that can communicate convincingly with humans. In this section, a brief overview of Gen-AI history is presented. Figure 3.3 illustrates a timeline and outline the main key concepts introduced.

The origins of these technologies date back more than 90 years, and significant progress was only made recently. The foundations of NLP and generative AI can be traced to early research on automatic language translation [18]. In the 1930s, a first machine was developed to translate words encoded on punched cards (see Figure 3.2). In the 1960s, various mathematical models were created to describe the structures of human language (including grammar and semantic). Several simulations were developed with moderate success, leading to a period of reduced interest in the subject for several years.

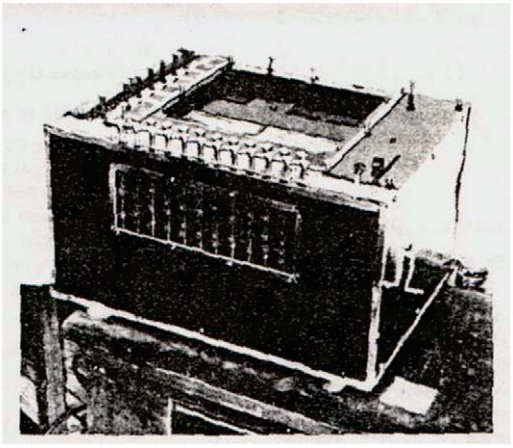


Figure 3.2: The mechanical brain from Georges Artstrouni - 1937 - *Extracted from [18]*

In 1986, Recurrent Neural Networks (RNNs) gained popularity, leading to new research into characterizing the probability of word sequences using neural networks and introducing the concept of distributed word representations [19], [15], [17]. This led to the development of a general neural network approach for various NLP tasks. In the 1990s, Statistical Language Models (SLM) emerged. These models predict the next word in a sequence based on the context of adjacent words. They had limitations and required the development of smoothing techniques to become effective, but they marked a significant advance in the field.

The introduction of the Long Short-Term Memory (LSTM) architecture addressed certain shortcomings of RNNs, offering memory support and stimulating research into techniques for analyzing longer text sequences [19].

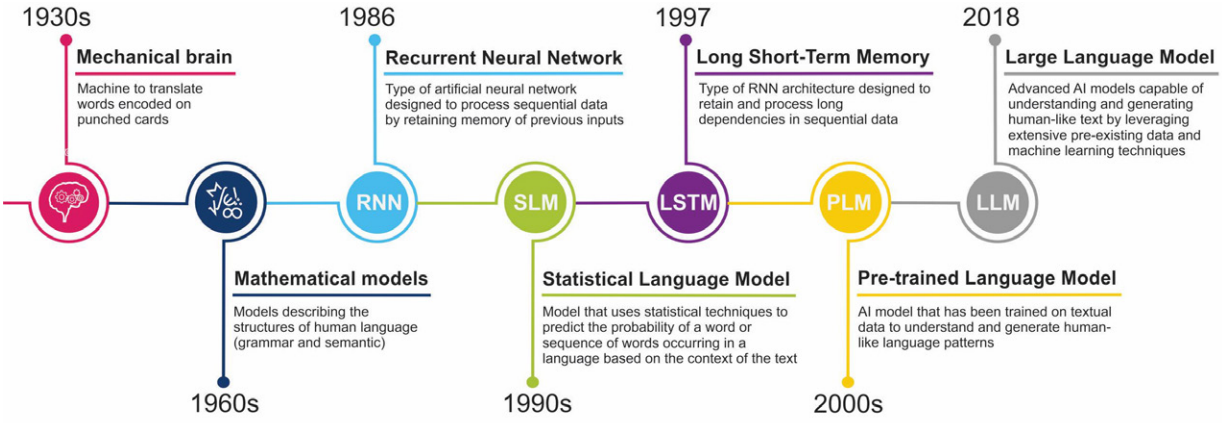


Figure 3.3: Chronology of the emergence of LLMs - *Created by author from section 3.2.2*

Then, Pre-trained Language Models (PLM) were proposed to capture context-aware word representations. These models involved pre-training bidirectional LSTM (biLSTM) networks and fine-tuning them for specific tasks. Additionally, the use of the highly parallelizable Transformer[4] architecture with self-attention mechanisms further enhanced context-aware word representations, significantly improving the performance of NLP tasks [19], [16]. Researchers discovered that scaling PLMs, whether in terms of model size or data size, often led to improved model capacity. These larger PLMs demonstrated surprising capabilities in solving complex tasks, marking the emergence of the first LLMs.

3.2.3 Emergence of Large Language Models

The evolution from larger Pre-trained Language Models to what is now recognized as Large Language Models marked a turning point in generative AI and natural language processing.

This evolution began with the development of various models demonstrating an impressive leap in natural language understanding and generation. Their successes underscored the potential of scaling models to achieve previously unforeseen abilities. Additionally, milestones in scaling and breakthroughs in training methodologies further defined the emergence of LLMs. These milestones also included advancements in computational power, innovative techniques in data processing and model architecture, and the collective efforts in fine-tuning and refining models'capabilities.

The gradual transformation from smaller-scaled models to these Large Language Models not only redefined performance benchmarks across a spectrum of NLP tasks but also highlighted the unforeseen potential for these models to address complex language challenges [20].

[4]Transformers are neural network architectures that utilize attention mechanisms to efficiently capture relationships between different elements in input data. They are studied in the next section.

However, the unexpected emergent abilities observed in these models were not entirely foreseeable by extrapolating performance improvements from smaller-scale models. Moreover, these capabilities were not explicitly included during the training phase of the LLMs. The full extent of tasks that these language models can proficiently undertake remains unknown and it seems that the scale of the model is not the exclusive factor governing the unlocking of emergent abilities.

Looking ahead, the exploration of emergent abilities in language models involves not just scaling but also innovative approaches to further empower these models. This could encompass advancements in model architectures, data scaling, improved training techniques, and a deeper understanding of prompting techniques [20], [21].

3.2.4 Step by step technological evolution

Looking more specifically to the technology behind the LLMs reveals the step by step evolution based on the progress of both Gen AI and NLP. Over 40 years, a series of advancements has occurred, transforming LLMs from specialized isolated networks to more general-purpose systems. The technological concepts introduced in the following evolution will be detailled in the next section.

In the mid-80s, experiments began with recurrent neural networks (RNNs) to understand sequential patterns. Jordan's work in 1986 introduced the concept of state units [22], creating a recurrent neural network capable of learning and predicting sequences. Later, Jeffrey Elman expanded this idea [23], training a network on language and observing its ability to understand word boundaries and hierarchical meanings.

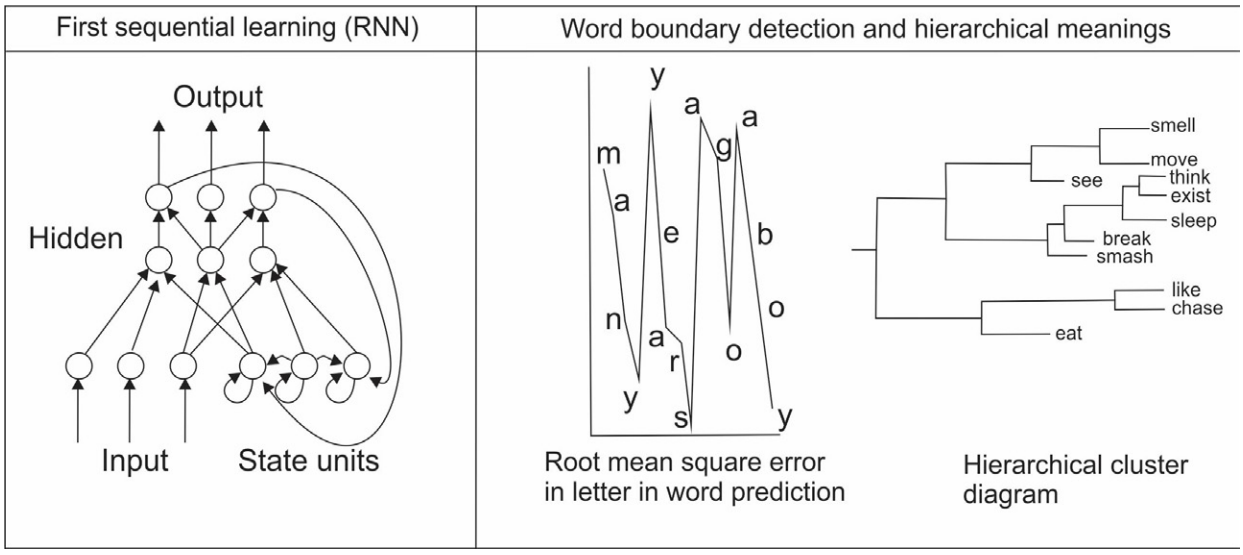


Figure 3.4: Reccurent Neural Networks concept - *Reproduced by author from [22] and [23]*

Figure 3.4 illustrates these concepts, presenting a basic recurrent neural network on the left side, including the state units. On the right side, word boundaries are highlighted from a word prediction process. A hierarchical cluster is also displayed and shows that the network classify words into groups of similarities.

Significant progress occurred in 2011 [24], with a focus on character-level prediction for text compression. The approach shifted towards larger networks[25], and in 2017, OpenAI's team built on this work, training a massive recurrent network on 82 million Amazon reviews. They discovered neurons capable of understanding complex conceptual concepts, such as sentiment, emerging from the learning process.

The breakthrough in 2017 also saw the introduction of transformers [26], utilizing self-attention layers to process input sequences in parallel. This architecture addressed the memory bottleneck, allowing LLMs to capture long-range dependencies more effectively.

The attention mechanism enabled a network to consider the entire context simultaneously, leading to the development of BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). Their success prompted subsequent iterations. BERT encompassed 330 millions parameters while GPT-2, with 1.5 billion of parameters utilized web-scraped data and a more extensive network.

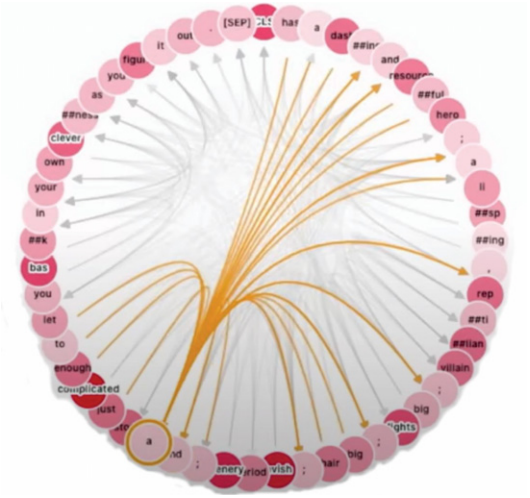


Figure 3.5: Self attention principle - *Extracted from [26]*

Figure 3.5 presents the principles of self attention. To absorb some context, each word is compared to others, allowing transformers to update the word embedding by considering the surrounding words. These mechanisms will be discussed in detail in the next section.

The results were astonishing, showcasing proficiency in various tasks, but challenges remained in maintaining coherence over extended sequences. GPT-3, however, with 175 billion connections and 96 layers, marked a turning point. Trained on the entire common web and Wikipedia, it exhibited increased performance across diverse measures.

The most remarkable aspect of GPT-3 was its capability for in-context learning, allowing the network to adapt its behavior without altering its weights. This presented a new computing paradigm where the computer operated at the level of thoughts, responding to prompts as a form of programming. Despite its potential, GPT-3 remained relatively unknown to the general public until efforts were made to refine its behavior further, making it more adaptable to human instruction.

From these foundational principles, numerous models and research efforts have refined these concepts, leading to the creation of the multitude of Models currently available.

3.3 Overview of LLM Technology

3.3.1 Principles and mathematical abstraction

General principles

Typically, LLMs refer to language models that contain hundreds of billions (or more) of parameters, which are trained on massive text data [6], [7]. They encompass various concepts and architectures including but not limited to embeddings, transformers, attention mechanisms, pre-training, fine-tuning and prompt engineering as shown on figure 3.6. Further explanations on these components will be provided in the upcoming sections. They collectively contribute to the LLMs' ability to understand, process, and generate human-like natural language responses across diverse applications and domains, making them increasingly indispensable in various fields such as customer service, content creation, translation, and data analysis.

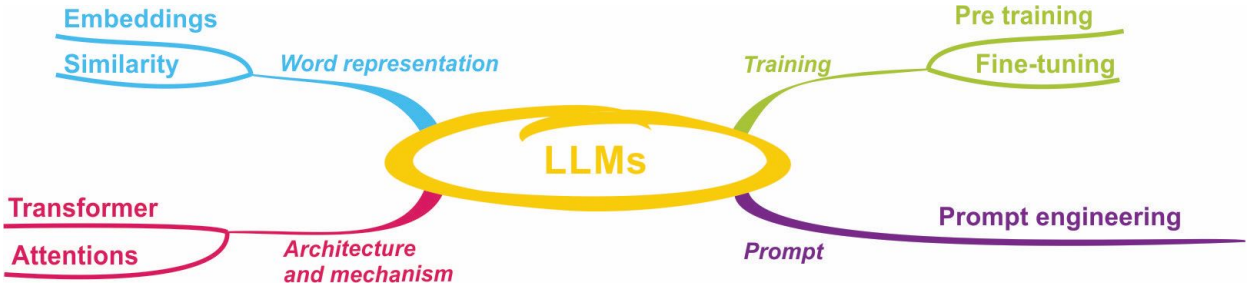


Figure 3.6: Overview of the main concepts and architectures forming the LLMs - *Created by author*

Following the presentation of a general mathematical representation of the LLMs paradigm, the subsequent sections will explore the fundamental concepts underlying LLMs including an overview of the neural network learning. Additionally, these sections will not only cover the fundamental principles of the concepts but also incorporate mathematical representations and practical use cases when feasible. This approach aims to provide an in-depth understanding of large language models.

Mathematical representation

To capture mathematically the concept of Large Language Models, a simple model is proposed [27], [28]. LLMs showcase an ability to engage and interact with humans using natural language which implies the capability to understand and predict the next word in a sequence of words. Conceptually, LLM estimates the conditional probability of the next word given the preceding words in a sequence by assessing the joint probability:

$$P(w_{1:T}) = \prod_{i=1}^{T} P(w_i|w_{i-1}, ..., w_1) \tag{3.1}$$

This joint probability represents the generation of a complete sequence of words $w_{1:T}$ by decomposing this probability into a series of conditional probabilities. Each conditional probability $P(w_i|w_{i-1}, ..., w_1)$ captures the prediction of a specific word w_i based on all the preceding words in the sequence.

By using the joint probabilities, the model can estimate the likelihood of various potential next words in a sequence. The word with the highest probability becomes the model's prediction for the next word. Figure 3.7 illustrates the core principle. It involves computing conditional probabilities of particular words and their combination to derive the joint probability. Python code (B1) was developed to compute random conditional probabilities as an example (displaying three). These probabilities are merged to calculate the joint probability, where, in this particular instance, the word "crash" is the model's predicted next word.

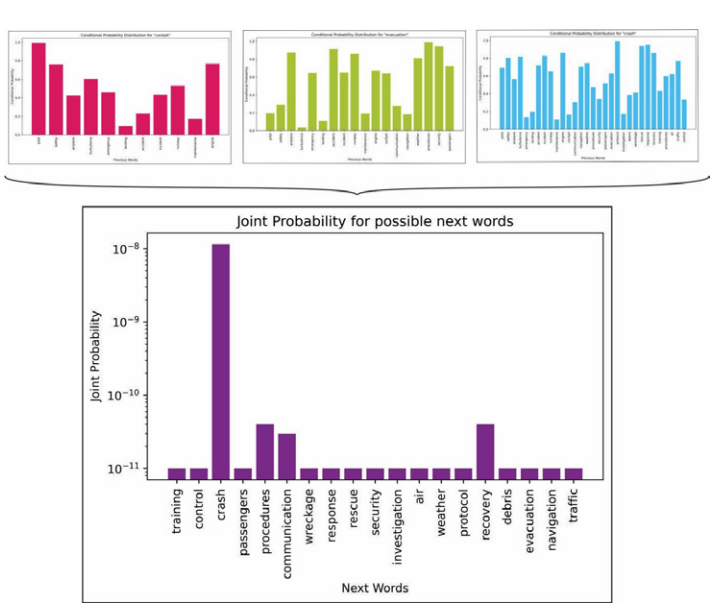


Figure 3.7: Conditional and joint probabilities for a sequence - *Created by author (code Python: B1)*

In reality, as detailed in the next section, while real words prediction mechanisms may be more sophisticated, the fundamental mathematical concept remains unchanged.

In many applications, users can interact with LLMs (as described in the next section). To do so, they provide a "prompt" as input to the model. The model analyzes the input and provides a response in natural language. This process can be modelled as follows:

$$P(w_{t+1:T}|w_{1:t}) = \int_{\theta} P(w_{t+1:T}|\theta)P(\theta|w_{1:t})d\theta \tag{3.2}$$

where the equation expresses the conditional probability of the outputs $w_{t+1:T}$ given a prompt $w_{1:t}$ in the form of an integral over the various possible underlying concepts θ. The integral captures how the model infers and utilizes the different potential concepts or themes given a specific prompt to generate coherent outputs.

3.3.2 About Neural Networks

Principles

Before introducing LLM-specific concepts, this section discusses the fundamental concept of the Neural Network, a central aspect of the learning mechanism [15], [29].

The concept involves creating a densely connected network of cells to learn and recognize patterns. A typical neural network consists of a series of artificial neurons called units arranged in layers. The first layer comprises the input layers, which receives the information to be processed by the network. Other units are located at the output of the network and provide the network's response to specific inputs. In between, there are multiple layers of fully connected hidden units (see figure 3.8). The connections between units are represented by weights that can be positive or negative, indicating the influence of one unit on another. While a simple neural network might consist of just few layers, a richer structure, called "Deep Neural Network" could involve many different layers between the input and the output.

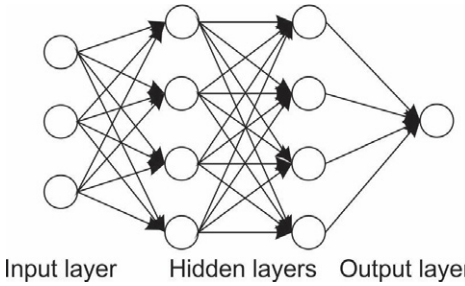


Figure 3.8: Scheme of a 4-layers neural network - *Created by author from [15]*

To be able to use this neural network, it must undergo an initial training phase. In the most basic networks, input information travels through the network in one direction, generating an output. Such a network is known as a feedforward network, where connections between neurons do not form cycles. More complex structures exist but they fall outside the scope of this short overview on neural networks.

In a basic Neural Network, each unit receives inputs from units on its left, with these inputs being weighted by the connections they traverse. Every unit sums the received inputs, and if the sum exceeds a particular threshold, the unit fires, activating the adjacent neuron to its right. During this phase, the network receives input information and in order to learn, a feedback process called backpropagation is implemented. This process involves comparing the produced output by the network with the expected response. The resulting difference is then used to update the weights of the connections among the units in the network, by propagating the update backward throughout the network. This method enables the network to learn by minimising the disparity between the provided response and the expected one. Several more complex methods, based on this core principle, have been developed. For instance, in Convolutional Neural Networks, the hidden layers perform convolutions rather than straightforward multiplications.

Once the network has been sufficiently trained, it reaches a point where it can used to be exposed to new set of data.

Mathematical representations

A basic representation of a Neural Network involves conceptualizing the network as a parameterized function [15], [29]:

$$\mathcal{N}(x, \theta) \tag{3.3}$$

where x represents the input supplied to the function, and θ is a vector of parameters governing and shaping the function. As discussed previously, this network must be trained by using training data consisting in pairs $(x, \mathcal{N}(x))$. During the process, the parameter vector is adjusted $\theta \rightarrow \theta^*$ such that the resulting network function $\mathcal{N}(x, \theta^*)$ is as close as possible to the desired function:

$$\mathcal{N}(x, \theta^*) \approx \mathcal{N}(x) \tag{3.4}$$

The procedure for adjusting the model parameters θ is called the learning algorithm. Looking inside the network, each unit consists of 2 operations:

- The preactivation $prea_i$ is the sum of the weighted incoming data to the unit $W_{ij}d_j$ and a bias b_i. For a network with n_{out} neurons, and a data vector of size n_{in}, this gives:

$$prea_i(d) = b_i + \sum_{j=1}^{n_{in}} W_{ij}d_j \quad for\ i = 1, ..., n_{out} \tag{3.5}$$

- The activation σ_i. In this process, each unit fires or not depending the value of preactivation and produces an activation. The activation function is usually chosen to be a non linear function.

The neural network comprises layers composed of n_{out} units or neurons. These units take an input of a n_{in}-dimensional vector of data d_j and produce an output of a n_{out}-dimensional vector of activations σ_i. The network structure involves connecting different neurons to form a layer, and aggregating these layers forms what is called the Multilayer Perceptron. This architecture is recursively defined as follows:

$$\begin{aligned} prea_i^{(1)}(d_\alpha) &= b_i^{(1)} + \sum_{j=1}^{n_0} W_{ij}^{(1)} d_{j,\alpha} && for\ i = 1, ..., n \\ prea_i^{(l+1)}(d_\alpha) &= b_i^{(l+1)} + \sum_{j=1}^{n_l} W_{ij}^{(l+1)} \sigma(prea_j^{(l)}(d_\alpha)) && for\ j = 1, ...n_{l+1} \\ & && and\ l = 1, ..., L-1 \end{aligned} \tag{3.6}$$

The equation 3.6 describes a network with L layers, where each layer consists of n_l units, and the outputs of one layer serve as inputs to the next layer through weighted connections and activation functions.

Use cases

To illustrate the neural network concept, a Python code has been implemented B2. It consists of a 3-layer neural network. The objective is to teach this network the XOR (exclusive or) function. The training involves presenting random data and comparing the obtained results with the expected output.

If the obtained results are incorrect, a backpropagation mechanism updates the network's weights (initially generated randomly).
Figure 3.9 shows the evolution of the difference between the expected output and the calculated output as a function of the number of iterations. In this simple model, a rapid convergence is observed, indicating that the network has "learned" the function. Tests confirm that the learning is indeed correct.

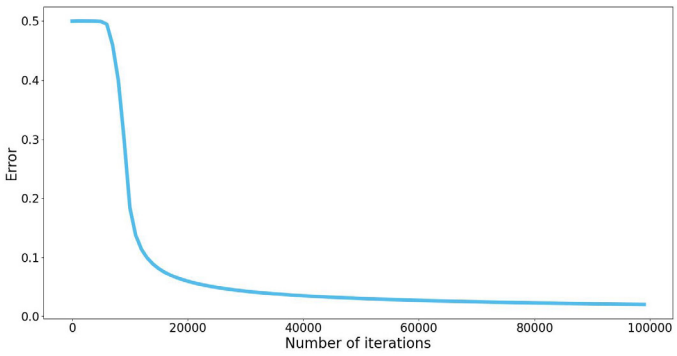


Figure 3.9: Difference between network and expected output - *Created by author (code Python: B2)*

3.3.3 Embeddings and similarity

Principles

The goal of Natural Language Processing (NLP) is to enable computers to understand human language. While humans communicate with words and sentences, computers operate using bits and numbers. To bridge this gap and develop a computer capable of understanding human communications, a mechanism to convert words and sentences into numerical representations needs to be developed. This mechanism is commonly referred to as "word embeddings" [30], [31].

This technique aims to be effective by not only representing words but also capturing specific properties. For instance, similar words should have similar numbers/scores. Ideally, the technique should also go beyond word similarity, it should try to capture many other features of the word. Therefore, word embedding involves encapsulating in a vector a set of numerical values, each value corresponding to a specific feature of the word. Some of these numerical values may represent important properties like similarity while others may signify complex relationships identified by the embedding model.

The resulting embedding vectors can be of substantial size[5] depending on the models used. After the embedding process, text is mapped into a multi dimensional vector space. The vector represents the "location" of the text in this space. Similar words are located closely in this space whereas words that are different appear farther apart. This representation enables the easy capture of the context and intent of the text [31].

[5]Typical embedding vector lengths can range from 1536 to 4128 elements, for instance.

To build the numerical representation of words/sentences, the method employs supervised or unsupervised machine learning techniques[6]. A set of documents, sentences or words is used for the learning process. The larger and more representative this set, the better the quality of the embeddings. This set is divided into text units called tokens. For each word, a context is created by analyzing adjacent words. Context learning is typically performed using counting methods (such as creating a co occurrence matrix) or neural network based methods with optimisation.

While the word embedding model is useful for representing words numerically, it is not sufficient for a deep understanding of human language. Language is more complex than a simple collection of words. Therefore, relying only on a method based on word embedding is not sufficient to interpret human language. For instance, two different sentences could have exactly the same embedding score while conveying very different meanings. Hence, it is necessary to develop a sentence level embeddings to capture word order, language semantics and their subtleties [31].

Most competitive models use this embedding concept in an encoder decoder structure:

- *Encoder*: the encoder takes an input like a sentence and processes it to produce a representation vector. The encoder's task is to compress the information from the input sequence into a context vector using embeddings and optimisation mechanisms.
- *Decoder*: The decoder takes the context vector and generates an output sequence.

From this numerical model of sentences and words, various algorithms have been developed to enhance and provide the most relevant embedding vectors.

Mathematical representation

Consider a text corpus represented by a set of words $E = \{w_1, w_2, ..., w_n\}$ where n is the size of the text. The objective is to create a d-dimensional word embedding vector for each word w_i within the dataset.

The algorithm forms word pairs $\{w_i, c_j\}$ by examining a target word w_i and a context word c_j within a designated window around the target. Then, it creates word pairs that can be observed in the corpus (positive sample) and generate random word pairs (negative samples). The model's task is to discern between positive and negative samples, adjusting word embeddings to ensure that words that appear together in the context window of the corpus (positive samples) have similar vector representations, while words that are randomly paired (negative samples) have dissimilar representations.

[6]In supervised learning, labels are defined and the learning process revolves around them. In unsupervised learning, the algorithm identifies patterns and creates similarity groups on its own [15].

This learning process is performed through a neural network like structure and can be expressed as follow :

$$\begin{cases} W = \mathcal{F}(E) \\ C = \mathcal{F}(E) \end{cases} \tag{3.7}$$

where $W \in \mathcal{R}^{d \times n}$ is a matrix which contains word embeddings of target words, $C \in \mathcal{R}^{d \times n}$ is a matrix which contains word embeddings of context words and $\mathcal{F}$ is a learning algorithm applied to a dataset E aiming to generate word embeddings that encapsulate semantic relationships between words based on their contextual usage within the corpus.

Use cases

Figure 3.10 presents the word embedding mechanism applied to several words. A Python script is provided in Appendix B3 for utilizing the *word2vec* algorithm. This algorithm is publicly available and uses a neural network model to learn word associations from a substantial text corpus. The *word2vec* model used was trained on a portion of the Google News dataset, encompassing approximately 3 million words and phrases. For each input word, the model provides a vector consisting of 300 numerical values.

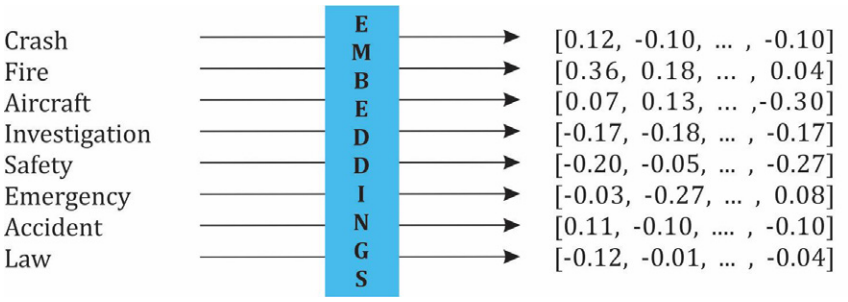


Figure 3.10: Illustration of the word embeddings mechanism - *Created by author (code Python: B3)*

In Figure 3.11, a visualization of the vector space is presented. The Python code in appendix B4 imports a small corpus of text and conducts a rapid learning process on it. Subsequently, using the t-SNE (t-Distributed Stochastic Neighbour Embedding) algorithm, the data is reduced to two dimensions and then plotted. The t-SNE algorithm is a dimensionality reduction technique employed for visualizing high dimensional data (vectors with 300 elements in the present case). This technique should always be use carefully as some information may be lost during the compression process.

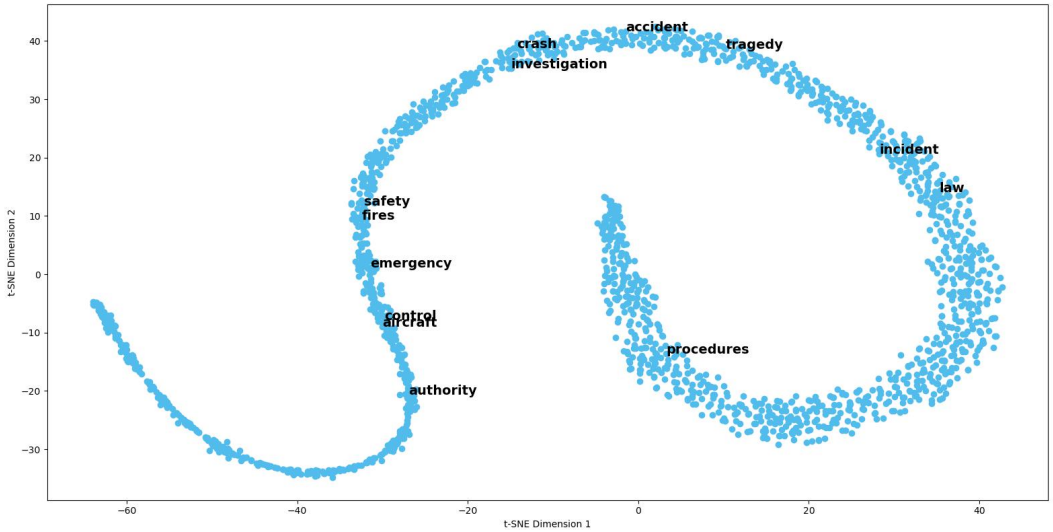


Figure 3.11: Visualization of the reduced vector space by the t-SNE algorithm - *Created by author (code Python: B4)*

In the visual representation, specific words are labelled in order to highlight how certain semantic relationships and similarities are encapsulated within the embeddings process. It can be seen noticed that words such as "*crash*" and "*investigation*" are closely positioned within the given learning corpus because they share a similar semantic context, while other words such as "*procedures*" and "*emergency*" are identified as quite different from each other.

As described above, with embedding, two similar words will yield similar vectors positioning them closely within the vector space [30], [31]. In essence, measuring the distance between two vectors facilitates an easy assessment of similarity. To quantify this similarity, the objective is to identify a metric that yields a high value when words/sentences are similar and a low value otherwise. The Euclidean distance exactly provides the opposite of this property. Therefore, two approaches have been defined and are frequently employed for this purpose:

- *Cosine similarity*: this method computes the cosine of the angle between the lines connecting the origin and each word/sentence. When the words/sentences are similar, the resulting angle is small. Conversely, if the words/sentences are dissimilar, the angle becomes large. If the angle is small and thus close to 0, its cosine is close to 1.

- *Dot product*: The concept behind this method is based on the notion that when two vectors are similar, multiplying their corresponding elements and summing the products should yield a high score. In contrast, dissimilar vectors result a low score. The operation that multiplies the corresponding elements and adds the products obtained is known as the dot product.

Both of these methods offer rapid and convenient means of evaluating similarity. They are commonly employed in algorithms for tasks such as recommendations determination or categorization.

Pairs to evaluate	Cos similarity	Dot product
Crash - Fire	0.30	2.92
Crash - Aircraft	0.22	2.37
Crash - Investigation	0.19	2.24
Crash - Safety	0.19	1.77
Crash - Emergency	0.17	1.71
Crash - Accident	0.80	8.51
Crash - Law	0.03	0.28

Table 3.1: Similarity of different pairs - *Created by author (code Python: B4)*

Table 3.1 displays the cosine similarity and dot product values for a set of word pairs, calculated using the Python code provided in Appendix B4. Notably, words such as "*crash*" and "*accident*" show a high degree of semantic similarity, as indicated by their high similarity values, whereas "*crash*" and "*law*" do not share such strong semantic associations.

3.3.4 Attention and transformer

Principles

With the word/sentence embedding mechanism, a computer system can achieve a global understanding of human language. However, this technique comes with certain limitations notably in the case of words that have multiple meanings. In such cases, the algorithm assigns the same embedding vector to all interpretations of the word without distinction. To address this issue, attention mechanisms have been developed [26], [32], [33].

Practically, the model examines each word in the sentence, assigning it a weight based on the context and its relative importance in the sentence. These weights are then employed to calculate a weighted sum of the input sequence. This weighted sum passes through a series of layers to yield the final output sequence, which represents a probability distribution reflecting the importance of each word within the input sequence and taking into account the weights assigned by the attention mechanism. As a result, the modified sentence vector is more contextually representative, and identical words with different meanings produce distinct sentence embeddings.

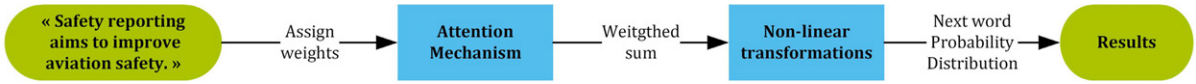


Figure 3.12: Principle of the attention mechanism - *Created by author from [32]*

The figure 3.12 visually encapsulates the fundamental concept of the attention mechanism [32]. The model's primary objective is to identify the relationship between the words "*safety*" and "*aviation*". To achieve this, it examines every word in the sentence and allocates weights based on their significance. In this example, a substantial weight is expected to be assigned to the connection between the terms "*safety*" and "*aviation*".

This simplified version of the attention mechanism, known as self attention, serves as the foundation for more advanced techniques, such as the multi-head attention mechanism. In this method, several different embeddings are considered and combined to produce more context relevant embedding vectors.

Additionally, a mechanism called transformers is also used in large language models. Transformers are developed to better grasp of the context and facilitate the generation of coherent outputs [26], [33], [34]. They construct text one word at time, predicting the next word based on context. The basic principle of this mechanism is illustrated on the figure 3.13 [10], [34]. The transformer architecture is structured with an encoder and a decoder, both comprising multiple layers of self-attention and feed-forward neural networks. The encoder's role is to process the input sequence, while the decoder's responsibility is to generate the output sequence.

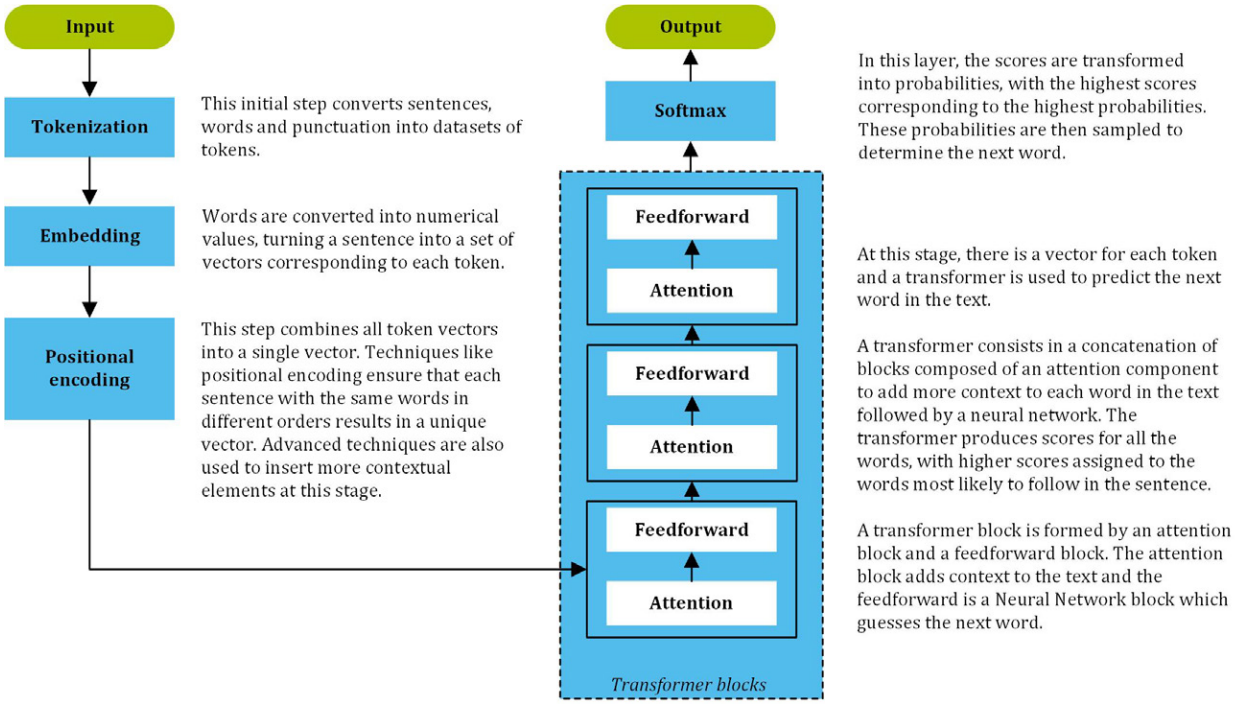


Figure 3.13: Visualization of the transformer architecture - *Created by author from [10] and [34]*

The attention mechanism in the transformer, with its multi-head design, enables the model to selectively concentrate on relevant portions of the input sequence, facilitating the capture of complex relationship between the input and output sequences.

Mathematical representation

Mathematically, the workings of the Transformer and its attention mechanism can be represented in a matrix format [26], [32]. Let's define $\mathcal{X}$ as the matrix that encodes each word embedding from the input sentence. Assuming the use of 10 attention heads, matrix multiplications of $\mathcal{X}$ with specific weight matrices (W^Q, W^K, and W^V) are performed to yield the Query, Key, and Value matrices Q, K, and V:

$$\begin{cases} \mathcal{X} \times W^Q = Q \\ \mathcal{X} \times W^K = K \\ \mathcal{X} \times W^V = V \end{cases} \tag{3.8}$$

The weight matrices W^Q, W^K and W^V are obtained from the trained model. The query represents the specific element or token for which importance needs to be calculated in relation to other elements in the sequence. The key denotes the elements within the sequence against which the query is being compared, while the value encapsulates the information associated with each element in the sequence.

Given the consideration of 10 heads, 10 sets of matrices are generated: Q_n, K_n, and V_n where n ranges from 1 to 10. For each head, the attention matrix $\mathcal{Z}_n$ is computed using the following equation:

$$\mathcal{Z}_n = Softmax\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V \tag{3.9}$$

The 10 attention head matrices are then concatenated and multiplied with an additional weight matrix W^O, which is jointly trained with the model. This yields the combined result:

$$\mathcal{Z} = [\mathcal{Z}_1 \mathcal{Z}_2 ... \mathcal{Z}_{10}] \times W^O \tag{3.10}$$

Since Transformer models typically consist of multiple layers, these operations are repeated several times to maximize the discovery of relationships between words and the comprehension of contextual information.

Use cases

To illustrate these concepts, the Python script provided in appendix B5 utilizes a pre-trained BERT model. The model is configured to generate attention weights as part of its output. The process begins with the definition of an input text : "*Safety reporting aims to improve aviation safety*".

The text is then passed through the model to obtain the attention weights. Additionally, the script allows to choose a specific layer of the transformer, and it will plot the corresponding attention weights in the form of a heatmap (see figure 3.14).

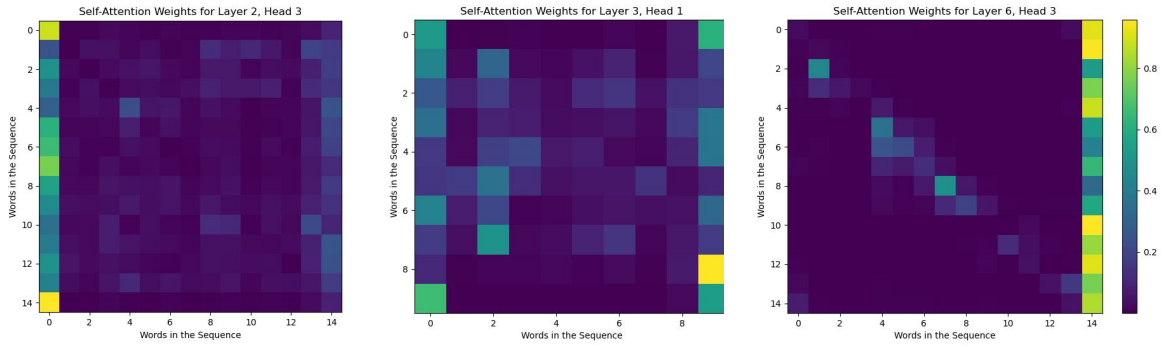


Figure 3.14: Heatmaps displaying attention weights for three distinct heads and layers - *Created by author (code Python: B5)*

The key insight in the process is that, depending on the selected layer, the model's attention is distributed across different words in the input text. This can be seen on the 3 heatmaps where the coloured column differs depending the layer and head considered. In a full transformer model, this operation is iterated for every word in the sentence concerning every other word. This enables to capture complex dependencies and relationships throughout the entire sentence. This capability enables the model to gain a profound understanding of the context and relationships that exist between words.

3.3.5 Pre-training, learning transfer and fine-tuning

Principles

As mentioned in the previous sections, Large Language Models rely on training's on large corpus of text data, a process known as pre-training [35], [19]. This process involves several steps as described on the figure 3.15.

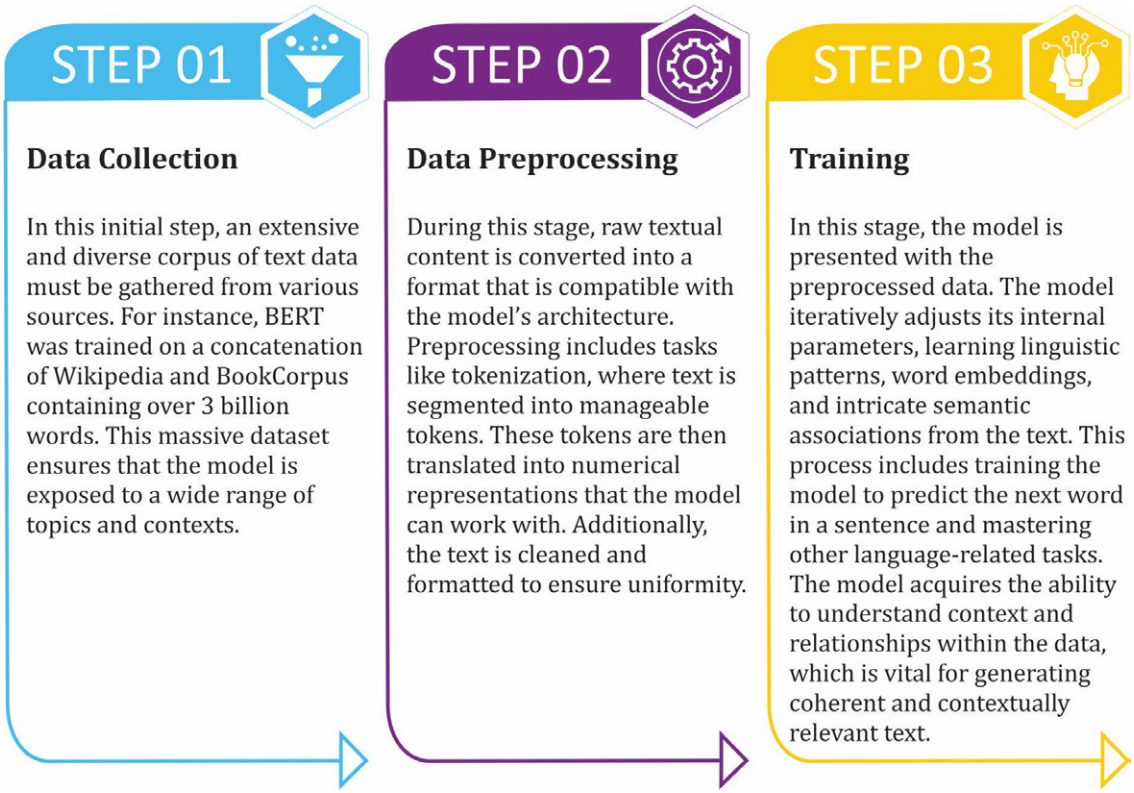


Figure 3.15: Pre-training process - *Created by author from [35] and [19]*

During the pre-training phase, the model is trained to minimize a loss function. This pre-training process employs deep neural networks, specifically architectures like the transformer, and involves techniques such as Masked Language Modeling[7] (MLM) and Next Sentence Prediction[8] (NSP).

As the model learns from the vast corpus of text data, it creates word embeddings that encapsulate semantic and contextual information. These embeddings allow the model to understand the nuanced meanings of words within different contexts.

One of the main advantages of this pre-training process is that the model do not need to be trained from scratch each time. Instead, the pre-training serves as a knowledge base, capturing general linguistic understanding. This transfer learning approach saves substantial time, computational resources, and data, making it a highly efficient solution for various natural language processing tasks.

[7]MLM involves randomly masking or hiding certain words in a sentence and training the model to predict or fill in the missing words based on the context provided by the surrounding words.

[8]NSP focuses on predicting whether a given pair of sentences in a document are consecutive or not.

After the pre-training phase, Large Language Models can be further optimized for specific applications through fine-tuning [35], [19], [36]. Fine-tuning involves training the model on a smaller dataset tailored to a particular task. In this phase, the model's internal parameters undergo adjustments to adapt it to the specific task. This process is notably faster and requires less data compared to the pre-training phase.

The fine-tuning process introduces a set of task specific layers on top of the pre trained model and trains the entire model on this newly introduced dataset. During this procedure, the weights of the pre-trained model are updated to better align with the specificity's of the new task while preserving the original knowledge gained during pre-training [37]. This approach ensures that the model retains its deep initial understanding and can effectively apply it to the targeted application, thereby achieving a balance between generalization and task-specific performance. Other training techniques [36], [38] exist and are being studied to optimize the current process (few-shot learners, meta learning, active learning[9]...). Due to the huge model size, it is very challenging to successfully train a capable LLM [19]. Distributed training algorithms are needed to learn the set of parameters of LLMs, in which various parallel strategies are often utilized.

Mathematical representation

Consider a pre-training dataset D_{pre} consisting of a large text corpus and,
let $E_{pre} = \{w_1, w_2, ..., w_n\}$ represent the words in this pre-training corpus.

During the pre-training, the algorithm $\mathcal{F}_{pre}$ applied to D_{pre} aims to create word embeddings that capture broad language patterns:

$$W_{pre} = \mathcal{F}(E_{pre}) \tag{3.11}$$

where $W_{pre} \in \mathcal{R}^{d \times n}$ is the matrix containing the embeddings of words derived from the corpus.

For a specific task represented by a smaller, task-specific D_{fine}, the objective is to adapt the pre-trained embeddings to suit the specifics of this task.
The pre-trained embeddings is used as a starting point of the fine tuning:

$$W_{fine} = W_{pre} \tag{3.12}$$

The fine tuning $\mathcal{F}_{fine}$ adjusts the pre trained word embeddings to better suit the task specific dataset:

$$W_{fine} = \mathcal{F}_{fine}(D_{fine}, W_{pre}) \tag{3.13}$$

where $\mathcal{F}_{fine}$ is the fine tuning algorithm that takes the task-specific dataset D_{fine} and the pre-trained word embeddings W_{pre} as inputs, refining the embeddings to better suit the specific task.

[9]In meta-learning, models rapidly adapt to new tasks with limited examples. Active learning strategically selects samples for maximum information gain, optimizing learning efficiency. Few-shot learners specialize in making accurate predictions with minimal data exposure.

Use cases

To illustrate the pre-training and fine-tuning principle, a BERT-based model (see appendix B6) is used to classify safety related sentences into one of the following three categories: "No Safety Issue", "Moderate Safety Issue", or "Safety Issue". When using the BERT model without additional training, it relies solely on its pre-trained knowledge. This knowledge is quite vast but not specific to this task.

Six sentences, for which we already know the safety levels, are presented to the model, and it predicts the corresponding category. The initial accuracy ranges from 33% to 50%. However, by applying a brief fine-tuning process using 30 labelled sentences, the model's accuracy increases from 66% to 80%. Subsequently, through a more extensive fine-tuning with a dataset of 3000 sentences generated randomly from the 30 initial sentences, the accuracy further increases to 85%. Table 3.2 presents those results.

	Without Fine-Tuning	Fine-Tuning (30 sentences)	Fine-Tuning (3000 sentences)
Accuracy (%)	33 - 50	66 - 80	85

Table 3.2: Analyzing model accuracy with and without fine-tuning - *Created by author (code Python: B6)*

Improved accuracy could be achieved by conducting fine-tuning with a set of 3000 distinct sentences. It is also important to note that fine-tuning comes with costs in terms of energy and time. However, despite its resource costs, it is a valuable step in adapting the pre-trained model to specific needs. It can lead to substantial performance enhancements, ensuring that the model more accurately meets the specific requirements of the task at hand.

3.3.6 Additional key mechanisms

In the previous sections, fundamental components and mechanisms used in the development of LLMs have been explored. In addition to these elements, several complementary mechanisms have been developed, including :

- *Masked Language Modeling*: this mechanism involves randomly masking some words in the input with the model then trained to predict the masked words. This helps the model to learn contextual relationship and to understand words in context.
- *Position-wise Feed-forward Networks*: These networks are applied independently to each position in the sequence, which allows the model to capture complex, non-linear relationships in the data.
- *Segment embeddings*: this a component that can be used to distinguish different segments or sentences within a single input sequence. They assist the model in recognizing the boundaries and relationships between these segments.

There are several other mechanisms and techniques that contribute to the capabilities of modern LLMs, making them powerful tools for natural language understanding and generation.

3.3.7 Prompt engineering

In addition to pre-training and fine-tuning techniques, a range of strategies has emerged to optimize the effectiveness of LLMs for task completion, commonly referred as "Prompt Engineering". This approach involves the strategic construction of prompts (instructions and examples incorporated into the input given to the language model) with the goal of producing accurate, meaningful, and contextually relevant responses [7], [37].

LLMs are not individually trained for every conceivable task. However, with the implementation of prompts, they can receive specific instructions or illustrative examples guiding them to perform novel or untrained tasks. This additional information serves as guidelines, directing the model towards the intended task. Prompt engineering takes this concept even further, focusing on refining these instructions to enhance LLMs' proficiency in specific tasks. Even minor alterations or examples in the prompt can significantly impact the model's performance [39].

The process of prompt engineering is iterative, involving the analysis of the model's output and subsequent adjustments to the prompt to achieve better results. Various parameters essential in prompt engineering, such as the adjustment of temperature, controlling the randomness of responses, and setting the maximum token length, which determines the length of responses. It's also crucial to avoid ambiguities by providing detailed and specific instructions regarding context, output length and structure, format, style, and the target audience within the prompt. Proper use of delimiters to clearly indicate input data and structuring the outputs is fundamental [7], [40].

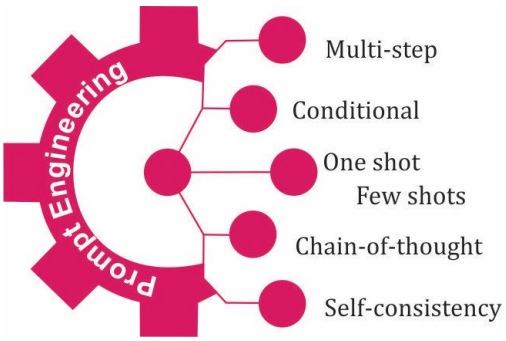


Figure 3.16: Overview of prompt engineering techniques - *Created by author from [37]*

Several strategies have been developed to refine LLMs for specific task as displayed in Figure 3.16. Conditional prompts with logic or multiple conditions can address problems with conditional outputs based on the input. One-shot or few-shots prompting involves providing either a single or multiple examples to guide the model, while multi-step prompting breaks down tasks into sequential stages. Chain-of-thought prompting aims to reduce model errors by requiring the model to provide reasoning steps before answering. Self-consistency prompting involves multiple prompts with a majority vote to obtain the final output.

These diverse strategies accommodate various complexities in tasks, providing the model with the necessary guidance and instructions to generate accurate and reliable outputs [36], [40]. This approach will be studied during the implementation phase. It is particularly important in the development of automated tools to ensure the production of correct and accurate results.

3.3.8 Challenges and future of LLM Technology

The evolution of LLMs is coupled with several multifaceted challenges that require innovative solutions to ensure their ethical and practical integration. Table 3.3 presents a SWOT analysis revealing the potential of LLMs but also their threats and weaknesses.

Strength	**Weaknesses**
Versatility	Societal bias
Scalability	Transparency
Base knowledge	No sourcing
Opportunities	**Threats**
Customization	Privacy
Foundation for new products	Obsolete data
	Lack of repeatability
	Lack of accuracy

Table 3.3: SWOT analysis - *Created by author from section 3.3.8*

One significant challenge arises from the extensive text data used to train LLMs, raising concerns regarding privacy, confidentiality, and the potential misuse of models for malicious intents. Ongoing research investigates techniques like knowledge retrieval to supplement LLMs with external knowledge sources, such as domain-specific databases to enhance contextual understanding and potentially mitigate the risks associated with sensitive data exposure [41], [42].

Another critical challenge involves addressing societal biases and stereotypes present in the training data, which can lead to biased outputs. Advances in prompt engineering aim to refine prompts and optimize the instructions given to LLMs, aspiring to mitigate biases and guide models toward generating more equitable and unbiased responses [43].

The lack of repeatability in LLMs can affect consistency in applications and pose challenges in critical areas. This variability stems from stochastic processes, sensitivity to input prompts, and differences in model parameters. Ensuring repeatability often requires deterministic settings to minimize these issues. Additionally, inaccuracies in LLM outputs can undermine trustworthiness and reliability, especially in critical applications. This underscores the need for careful tuning, validation, and consistent settings to improve both consistency and accuracy.

Moreover, the scalability and resource-intensive nature of LLMs present challenges in terms of compute budgets, memory requirements for fine-tuning, and limitations in handling lengthy inputs. Ongoing researches actively explores innovative solutions, including optimized LLMs and techniques that aim to improve efficiency, fairness and reduce their computational demands, making them more accessible and practical for a broader range of applications [41], [44].

In addition, methods such as reinforcement learning from human feedback offer a promising approach in refining LLMs by integrating human insights and corrections. This approach presents an opportunity to improving the accuracy, fairness, and overall alignment of LLM outputs with human expectations. Emerging areas such as deeper integration of multimodal learning, enable LLMS to generate outputs from various input types. There is also an increasing emphasis on improving interpretability within LLMs to enhance transparency and understanding of model reasoning [33], [41].

As the field rapidly progresses, these challenges and research avenues highlight the need for continual innovation, ethical development, and technical solutions to ensure that LLMs reach their full potential.

3.4 Overview of existing models and applications

3.4.1 From technological foundations to current models

The evolution of LLM technology began with the introduction of the Transformer model and was driven by 2 significant models, BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) series [7], [39]. BERT, introduced in 2018, marked a significant step forward by incorporating bidirectional learning to understand the context of words and sentences through its 330 millions parameters.

In parallel, the GPT series from OpenAI, starting with GPT-1 in 2018, and escalating to GPT-2 (1.5 billion parameters) and the groundbreaking GPT-3 in 2020 with 175 billion parameters, revolutionized the landscape of LLMs, demonstrating remarkable text generation capabilities and expanded language understanding [39]. The subsequent iteration, GPT-4, released in March 2023, further pushed the boundaries by integrating multimodal signals into its text input. This enhancement allowed GPT-4 to exhibit even more robust performance in solving complex tasks, surpassing its predecessors like ChatGPT [39].

However, alongside these significant advancements, there have been other notable models and milestones contributing to the evolution of LLM technology. For instance, ELMo (Embeddings from Language Models) utilized bidirectional LSTM networks for context-aware word representations. ELMo introduced the concept of fine-tuning pre-trained models for specific downstream tasks [7], [33]. Models like T5 (Text-to-Text Transfer Transformer), introduced by Google in 2019, focused on converting all NLP tasks into a text-to-text format, offering a unified approach for different tasks [21], [39].

While BERT and the GPT series stand as pillars of the LLM field, other models have played significant roles in shaping the ever-evolving landscape of LLM technology. Figure 3.17 shows a timeline of models exceeding 10 billions parameters [19].

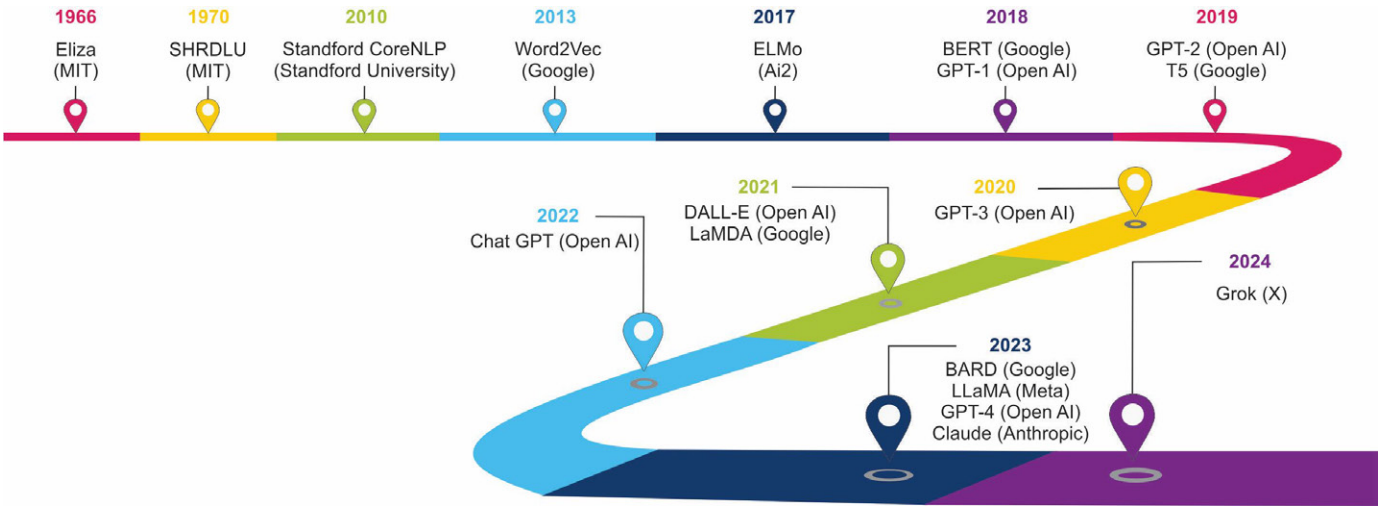


Figure 3.17: Timeline of existing large language models - *Created by author from [19]*

3.4.2 Versatile applications in focus

Generative AI encompasses a wide range of applications and is not limited to the development of versatile tools like the famous ChatGPT from OpenAI. Generative AI holds the potential to transform creative industries, content generation, and human-machine interaction, paving the way for further advancements in image synthesis, text generation, and beyond [21], [33], [45]. The list below presents a selection of categories featuring existing applications along with some well-known models.

The list is not exhaustive but highlights the most advanced categories (links to the corresponding application website are provided in the footnote[10]). Much research in the field is currently underway [33], [45],[46] and may expand the presented scope.

Type	**Description**	**Applications**
Text to Text	These models have the ability to transform text into various textual outputs.	ChatGPT and LaMDA (Language Model for Dialog Applications) are the most well-known, but various other models have significantly contributed to the development of this field. Models such as T5 and ConvS2S excel in tasks such as language translation, while InstruGPT specializes in handling follow-up questions.
Text to Image	Models in this category enable the generation of images from textual descriptions. Conversely, image-to-text aims to describe the content of an image.	Among the most well-known applications in this category are DALL-E and Midjourney. Notable models of Image to Text include Show and Tell developed by Google Research and VisualGPT, which employs an encoder-decoder attention mechanism to generate a suitable caption for a given image.

Table 3.4: Overview of Existing applications - part 1 - *Created by author from section 3.4.2*

[10]ChatGPT : https://chat.openai.com/ - T5 : https://huggingface.co/docs/transformers/model_doc/t5
ConvS2S : https://github.com/kamepong/ConvS2S-VC - DALL-E : https://openai.com/dall-e-2
Midjourney : https://www.midjourney.com/home/ - SandT : https://blog.research.google/2016/09/show-and-tell-image-captioning-open.html?m=1 - Visual GPT : https://github.com/VisualAI/visual-chatgpt
Dreamfusion : https://dreamfusion3d.github.io/ - Magic3D : https://research.nvidia.com/labs/dir/magic3d/
AdaSpeech : https://speechresearch.github.io - ProDiff : https://github.com/Rongjiehuang/ProDiff
CBERT : https://github.com/microsoft/CodeBERT - Codex : https://openai.com/blog/openai-codex
IMAGEN : https://imagen.research.google - Phenaki : https://phenaki.video
Dreamix : https://dreamix-video-editing.github.io/ - Galactica : https://galactica.org/

Type	Description	Applications
Text to 3D	To further develop the image generation to the gaming experience, certain models are designed to transform text into 3D images.	Notable applications in this domain include Dreamfusion and Magic3D.
Text to Video	With these models, visual video content is generated from text descriptions, or video content is enhanced by editing objects in the videos.	Some notable models in this category include IMAGEN Videos, Phenaki, and Dreamix.
Text to Audio	Text-to-audio techniques involve converting text into human-like speech and generating music with various vocal styles. Text-to-speech (TTS) is commonly used in applications such as voice assistants, voice navigation systems, and audiobooks.	AdaSpeech and diffusion based models such as ProDiff are the most well-known models in the field.
Text to Code	Text-to-code enables the generation of entire source code for specific applications, resolving issues within existing code by providing suggestions for corrections. Code-to-text allows for the generation of documentation for the code.	Models like CodeBERT and OpenAI Codex facilitate tasks such as generating code by providing comments in an editor.
Text to Science	These models are specifically engineered to answer scientific inquiries and assist in the generation of scientific papers.	Notable models include Galactica and Minerva.

Table 3.5: Overview of Existing applications - part 2 - *Created by author from section 3.4.2*

In this Master's Thesis, the capabilities of LLMs in the context of interacting with a safety database are explored. Consequently, applications for text-to-text and text-to-code generation will be predominantly employed for this purpose.

3.5 LLMS for database applications

As detailed in previous sections, the latest generation of models demonstrates remarkable adaptability across a multitude of tasks with little to no need of specialized training. This adaptability creates opportunities for applications in the context of data management. Within the scope of this Master's Thesis, Large Language Models are being examined as tools to analyze and explore safety data contained in the ECCAIRS database.

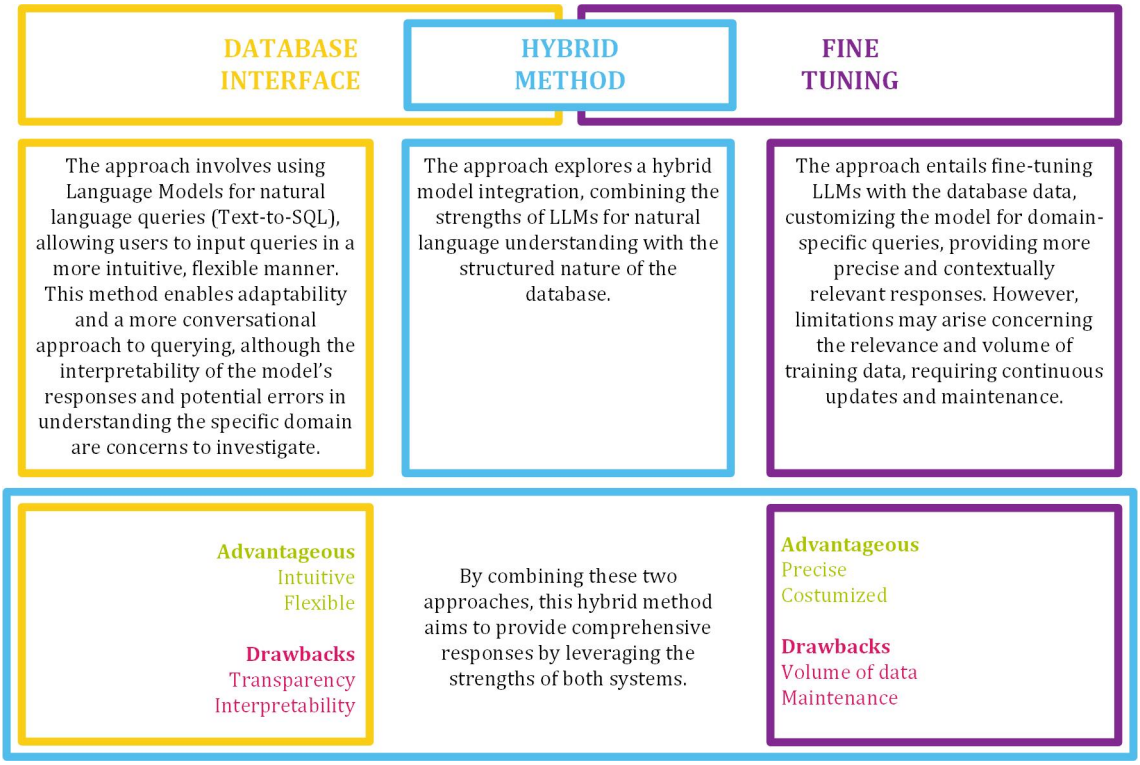


Figure 3.18: Approaches for data management with LLMs - *Created by author from section 3.5*

Various research studies propose different approaches (see figure 3.18) to tackle this particular problem [37]:

- The first approach, as presented in the works of [47], [48], [49] and [50], involves using Language Models (LLMs) as a database interface.
- The second approach entails fine-tuning LLMs with the database data [51], [52].
- The third approach explores a hybrid model integration.

The implementation chapter will investigate these techniques to identify the most effective approach for managing the safety database.

3.6 Libraries and interfaces

Large Language models have become increasingly accessible through a diverse array of channels. These options range from libraries designed for local language model utilization to APIs offering remote access to models that might not be publicly accessible. Such libraries play a crucial role in enabling the smooth integration of LLMs into applications, providing a straightforward path for the development and deployment of generative AI applications [37].

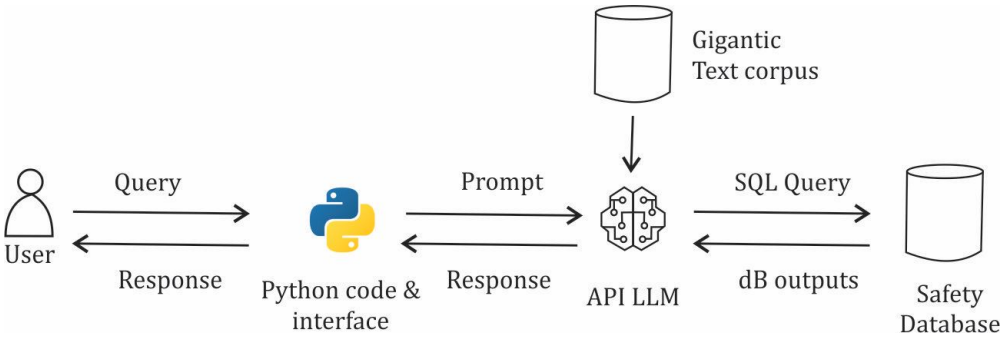


Figure 3.19: Flow data of an implementation using API - *Created by author from [37]*

Nevertheless, these libraries come with their drawbacks. Data privacy emerges as a critical determinant when choosing between an existing API and developing a custom AI model. The decision becomes crucial, especially when applications are tasked with handling sensitive or proprietary information.

Another significant factor to consider, particularly during the scalability of generative AI applications, is the cost of inference. Initially, using an API might seem cost-effective, especially for applications with low usage. However, as usage scales up, the cost dynamics can significantly change. APIs, often priced per token, could become expensive. Therefore, these escalating costs need to be carefully considered during the development stages.

Figure 3.19 presents an implementation architecture utilizing an LLM API. In this setup, the implemented Python interface and code are responsible for generating the prompt transmitted to the API. The LLM API, in turn, uses this prompt to generate an SQL query and subsequently executes this query on the database. The results obtained from the database are then returned to the API for initial processing.

Following this, the Python code analyzes and interprets the output generated by the LLM, leading to the production of the response intended for the end user. An interesting alternative method could involve refining the LLM API by utilizing data sourced from the safety database.

This Master's Thesis will explore different solutions mainly based on API utilisation to identify an optimal method for querying and analyzing a safety database.

Chapter 4

Implementation and case studies

4.1 Chapter structure

In Chapter 2, insights into the database D4S were gained and Chapter 3 provided knowledge about the LLMs and their applications. Building on this research and analysis, implementations can now be developed. This chapter focuses on the implementation of the LLM's solution to interact with the aviation safety database. Figure 4.1 showcases the structure of this chapter. Initially, a review of the various potential existing solutions is conducted, exploring three groups of solutions: LLMs with direct API access, AI solutions on cloud servers and local deployment of an AI model. These options are evaluated against industrial constraints, with data confidentiality as a key factor.

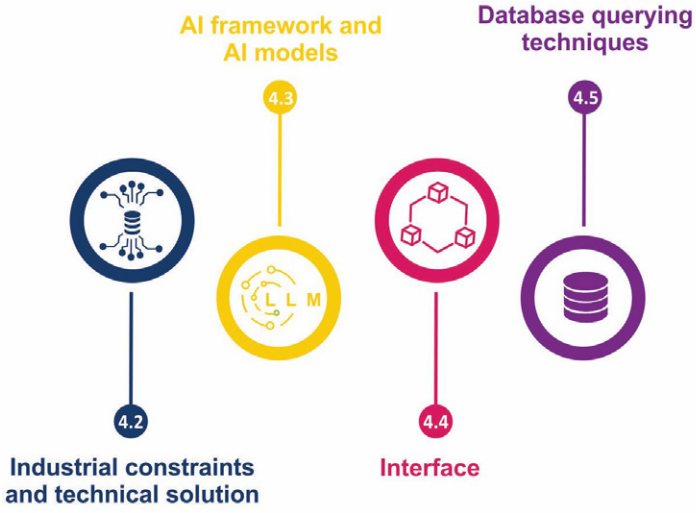


Figure 4.1: Structure of the chapter - *Created by author*

Next, the architecture of the AI framework and its components are outlined. Two AI approaches are explored: a local implementation on a personal computer and a deployment on a cloud server. The user interface and its logic are explained, followed by a detailed overview of the database querying techniques implemented. In the Text2SQL models, prompt engineering is performed by iteratively updating the model's parameters to align its understanding of natural language prompts with the structure and content of the database. For the fine-tuned model, Python code is created to format the data and enable assimilation by the vector database.

4.2 Industrial constraints and technical solution

4.2.1 Understanding the boundaries: sensitive information, ownership and privacy

Before reviewing the existing technology and discussing the most suitable solution for the project analyzed in this study, it is important to understand how LLM technology is currently made accessible [53]. This understanding is crucial for determining the best technology for industrial implementation.

Most LLM models are publicly available on various websites and can be used freely. While this easy access is very practical, their use requires caution for several reasons.

- **Always Safeguard sensitive information**
  The data used in the prompt, as well as any uploaded inputs or documents, can be used by the LLM provider for performance improvements or other uses. The data is uploaded to an external server and is not secured, therefore it shall be strictly limited to publicly available data.

- **Ownership of the outputs**
  In addition to copyright issues, the term of use of most AI websites restricts the ownership of the generated outputs.

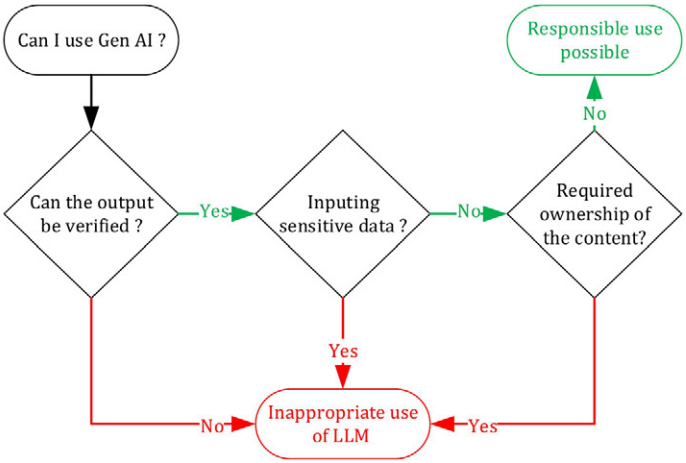


Figure 4.2: Flowchart on the use of Gen AI - *Created by author*

A responsible use of public generative AI involves being attentive upstream during the crafting of the prompt to avoid using sensitive data and being cautious downstream by verifying the provided outputs for their accuracy, copyright issues, and privacy concerns. A dedicated communication[1] has been prepared and addressed to EASA employees (see Appendix F) to raise awareness about the use of generative AI and the necessary precautions to avoid any potential issues.

More globally, this issue has also been well identified and recently with the emergence of more compact models, alternative solutions have been proposed that involve local implementation of generative models. These models keep data away from third party services, enhancing data privacy and security. However they require more maintenance, can be resource intensive and may necessitate specialised servers. This type of solution will be discussed and presented in this chapter and the following.

[1]This 2 pagers was created and realised under the request of EASA.

4.2.2 Reviewing solutions

When LLM technology first emerged, it was primarily accessible via cloud APIs provided by tech giants like `Google` and `OpenAI` [54]. These providers continue to dominate the market, making access to various LLMs straightforward. However, in return, they collect user data and often impose usage restrictions or monetize premium versions.

Recently, open platforms have been developed, offering alternative solutions to cloud-based LLMs. These platforms provide the possibility to download models locally [55], allowing LLMs to be run on local machines, provided the necessary resources are available. Table 4.1 provides an overview of the different technologies currently available [56].

Solutions	**Advantages and Drawbacks**	**Costs**
LLM with direct API access The LLM algorithm is accessed through an API.	⊕ Quick implementation is possible. ⊖ The data may be shared with a third-party provider.	Generally, this is a cost per request and per number of tokens used.
LLM with direct API access and fine tuning Some relevant data will be used to fine tune the LLM model. The resulting application should be more specific. The LLM model is usually still used through an API.	⊕ The application is very specific and should produce more accurate result. ⊖ A training of the model is necessary and data should be uploaded to a third-party server.	In addition to the cost to use the API, the fine tuning operation can be expensive and need to be updated regularly to keep the application up to date.
LLM through cloud service The data is shared within a cloud service. Developments and implementations are performed in the cloud including the use of the LLM model.	⊕ No need of an interface, the data and the model are shared on the same platform. ⊖ The full database has to be shared with the cloud service.	Recurrent costs are associated to the use of the cloud service.
Local deployment including fine tuning The LLM is implemented locally in the IT infrastructure (on a local server or a cloud).	⊕ It offers higher level of data protection. ⊖ Specialised IT resources are needed to enable the model to work. ⊖ Developments and fine tuning have to be performed locally.	Costs are split to the cost of the dedicated server and the maintenance cost of the model.

Table 4.1: Overview of existing solutions - *Created by author from section 4.2.2*

These can be categorized into three main groups.

- **LLMs with cloud API access**: Numerous models are accessible on the market through APIs, presenting a variety of options including GPT models from `OpenAI`, `Bard` or `Gemini` from `Google`, `Cohere`, and `Claude` from `Anthropic`. Each model possesses unique characteristics, differentiating it from others in terms of capabilities and applications.

- **Full cloud services**: Few providers offer the possibility to directly implement a customized solution on a dedicated server. Company like `OpenAI` and `Cohere` provides this capability and proposed formulas including support from the company in the business development. While several other companies have announced similar possibilities, as of now, nothing concrete is available.

- **Local deployment**: Various open source models are accessible, each with its own distinct characteristics. The number of parameters included in the model is also a critical factor in the decision making process. Depending on the available IT ressources, more sophisticated models can be selected. Models like Dolly 12B, Falcon 40B, LLamA are examples of models from this group.

In terms of costs, initially, accessing the LLM model through an API is generally not very expensive (only cost per request per 1K tokens). However, as the application becomes more frequently used, costs tend to increase, necessitating the exploration of alternative solutions. On the other hand, implementing a local LLM model using open-source software seems to be very cost effective. Nevertheless, it is essential to acknowledge that this approach requires specialized IT resources with sufficient memory and performance to ensure an effective and workable solution. Furthermore, local deployment involves internal development efforts, and there are recurrent associated costs related to the maintenance of the implemented solution.

4.2.3 Industrial constraints

During the kick off meeting in Cologne, EASA emphasized that, due to the sensitivity of the information stored on the platform, the selected technology must ensure data privacy and prevent sharing with third parties. The chosen solution must be functional and practical for data scientists. At first, the goal was to explore what was possible and no technological restriction apart from confidentiality was defined. Constraints related to industrial infrastructure, technological limitations and resource availability were introduced later.

Initially, based on these requirements and given that the Data4safety platform was not yet ready and available, first implementations were realised on a personal computer without direct access to the database. Then, a demonstration of the implementations created on the local computer was presented to EASA and a service provider who was introduced for the first time. Meetings were then organised with EASA and the service provider to discuss technical solutions given their constraints.

As EASA decided that the database would be hosted in a `Databricks` environment, frontend and backend implementations were not authorised and the interface was restricted to `Databricks` notebook[2] due to the enforcement of EASA's infosecurity policies. During the final stage of this study, access to the platform was granted by EASA through the service provider. The LLM models were limited to those made available on the platform during the duration of the Master's Thesis[3] and the original AI framework proposed for local implementation was not adopted. Due to the confidentiality restrictions, testing other platforms or cloud services (e.g. AWS, GOOGLE collab...) was also not possible[4].

Chapters 4 and 5 provide a detailed overview of all the solutions created and implemented as part of this study.

4.2.4 Technical solutions and implementation

The solution created aims to bridge the gap between natural language and the Data4Safety database interaction. This involved constructing the following components:

- an AI framework to house the LLM models and integrate them in the core code of the solution;
- a user interface to facilitate interaction and communication;
- techniques like Text2SQL and Retrieval-Augmented Generation (RAG) to enable natural language database querying.

This combination of components allows users to interact with the database using everyday language, empowering them to retrieve information seamlessly. Figure 4.3 illustrates this combination. Each component interacts to create a cohesive solution but can be updated independently to adapt to evolving requirements.

The different components created and explored are discussed in details in the following sections.

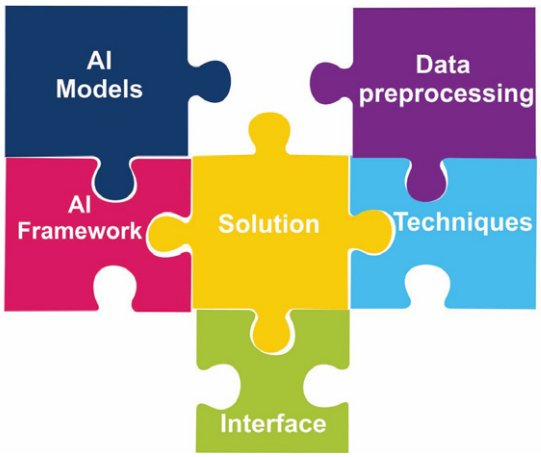


Figure 4.3: Modules composing the solution - *Created by author*

[2]While `Databricks` relies heavily on notebooks, it supports Python scripts and has dashboarding capabilities. Models developed can be served as APIs with continuous access to data from feature stores. Custom frontends can also be developed and connect to the model's API if needed. These complex solutions are outside of the scope of this Master's Thesis.

[3]Databricks offers multiple methods for integrating various models.

[4]EASA mandated the selected technology, limiting testing to the given platform. Separating cloud services from LLM services would have been preferable to mitigate vendor lock-in.

4.3 AI framework and AI models

4.3.1 Local deployment

The first architecture developed involves implementing a local AI framework [57], [58]. Recently, different solutions have emerged, such as `jan.AI` and `Ollama.AI`, which all share the same principle: a framework is implemented locally and can download AI models from online stores. These AI models are also stored locally, allowing the full architecture to function offline without sharing any information outside the locally created environment.

The local implementation has been realized using `Ollama.ai` [58]. `Ollama` is an open source application designed to work with LLM locally on personal computers or servers. Its versatility is demonstrated by the variety of available models, providing high flexibility for the user. Among these models are LLaMA-2, Mistral, CodeLLaMA, Falcon, and others. `Ollama` simplifies the use of LLMs, configuration, and data management into a unique package defined by a Modelfile. This integrated approach facilitates the deployment and use of complex LLMs on personal computers.

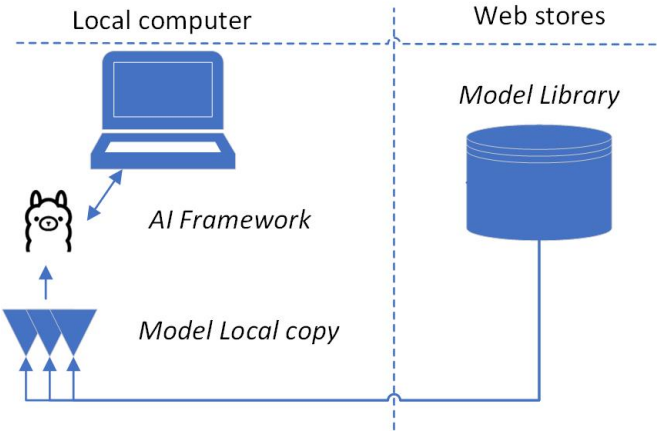


Figure 4.4: Local deployment: `Ollama` - *Created by author*

Figure 4.4 illustrates these principles, showing how the AI framework and the models are downloaded and implemented locally.

This architecture provides several advantages:

- Development: enables fast local iteration without the need to deploy model changes;
- Privacy and Safety: ensures that the data never leaves the local environment;
- Cost: eliminates costs associated with API calls or cloud services;
- Control: offers full control over the model, allowing it to be adjusted for specific needs.

Solutions and implementations were developed using this framework on a personal computer (appendix E1).

This solution was presented and discussed with EASA and the service provider. During the final stage of the study, the proposed local implementation was considered to not fit well with the technology of the Data4Safety platform. New constraints were imposed by EASA and solutions compatible with `Databricks` environment were developed.

4.3.2 Cloud implementation using `Databricks` API

The second architecture developed involves using the `Databricks` API (appendix E2 to E8). Since the database is implemented on a `Databricks` server, EASA has restricted the solution implementation to those involving the `Databricks` API [59].

For this study, access was granted to a `Databricks` server with the same tools available to the D4S platform's users. The server hosts a version of the database within a catalog. The environment provides SQL tools for querying the database. `Databricks` Notebooks can be created to interact with the database and implement Python code for advanced data processing. Most libraries are not pre-installed, and due to security reasons, library installation is curated, requiring users to install approved libraries as needed for each notebook execution. Figure 4.5 shows the provided `Databricks` environment.

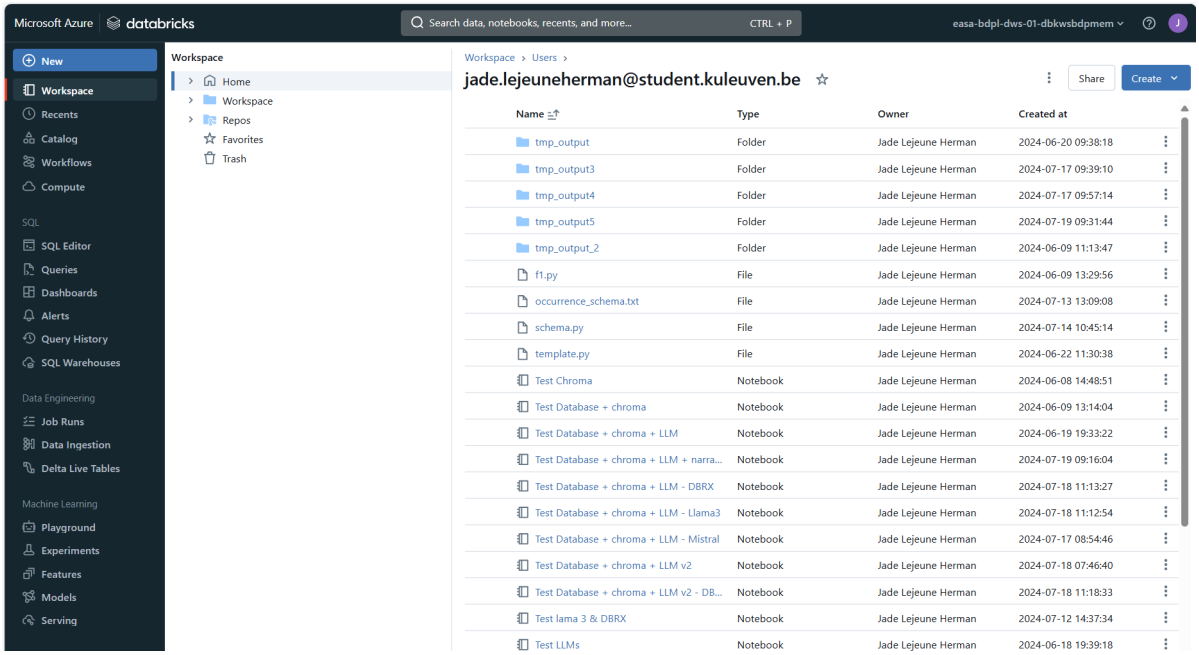


Figure 4.5: Overview of the provided `Databricks` environment - *Extracted from `Databricks` server*

In such solutions, the AI framework send the data to analyse to a LLM via an API call to a serving endpoint which ensures secure and correct communication with the AI model. Upon receiving the response from the AI model, the notebook processes the returned data and provides the answer to the user.

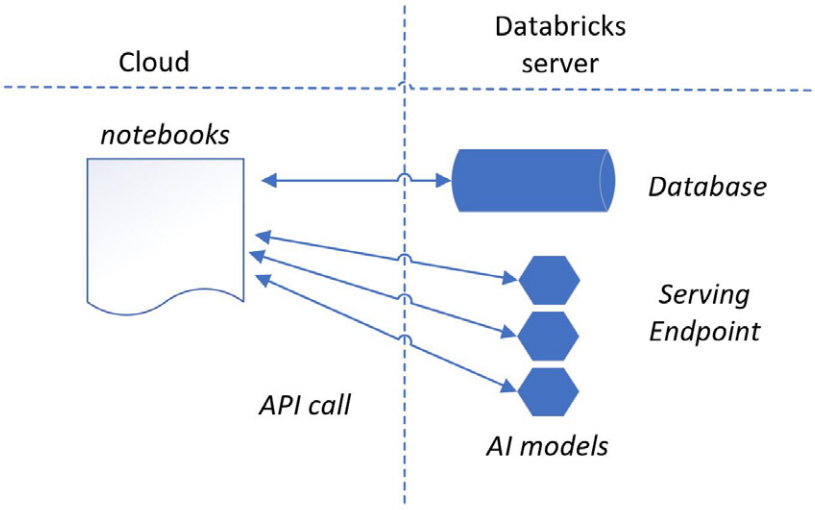


Figure 4.6: API call principles - *Created by author*

For each AI model, a dedicated end point has to be created within the `Databricks` API. This solution offers tight integration with the server containing the database and the LLM models are deployed in a secure `Databricks` workspace.[5] Figure 4.6 shows the details of the second architecture.

4.4 Interface

As already mentioned, initially, a frontend/backend interface was developed to replicate a typical user chat interface (such as chatGPT). However, EASA decided to restrict the interface to the use of `Databricks` notebooks[6]. This section presents and discusses these two approaches.

4.4.1 Frontend and backend interface

With the initial solution, a frontend/backend architecture was created to deliver user friendly experience. This approach separates the user interface from the underlying logic. The interface has been created using `Flask` [60] with the following principle: the frontend, built with HTML provides the interface and users can simply enter prompts through forms and interact with buttons. The backend, written in Python (appendix E1), acts as the central hub. The core logic involves managing the AI framework, coordinating with chosen models and interacting with the database. This centralized approach ensures consistent and efficient use of the AI framework for all users. Figure 4.7 presents the user interface developed.

[5]The use of `Databricks` and API to access LLM models was discussed with EASA in terms of privacy. EASA indicates that the level of confidentiality offered by `Databricks` meets its requirements within the framework of the D4S program, which imposes high security standards. One can note that while `Databricks` claims its models are stateless and each query is isolated, the company may temporarily store inputs and outputs for security purposes, indicating a potential data sharing.

[6]EASA justifies this choice to comply with the security measures defined on the platform.

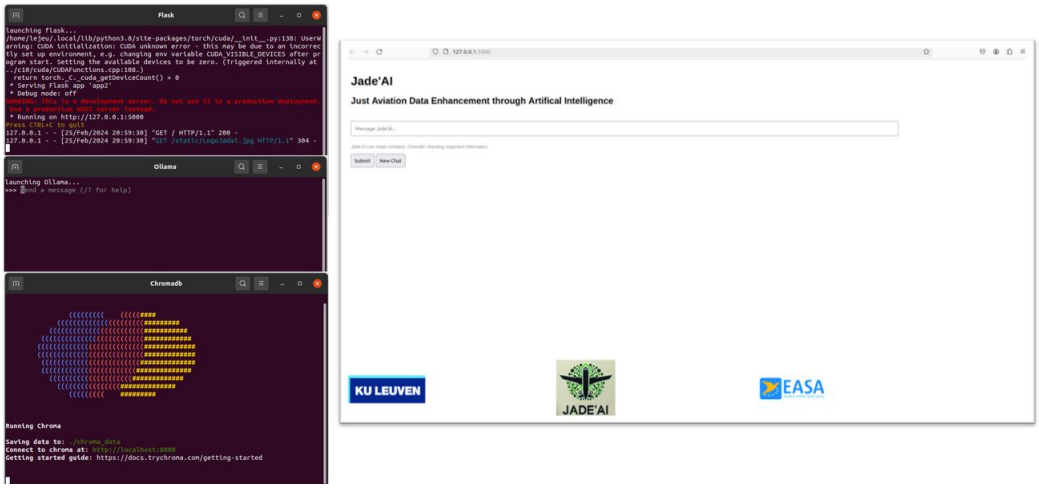


Figure 4.7: Interface using Flask - *Created by author*

Developing and maintaining both frontend and backend components can be more complex compared to `Databricks` notebooks. Deploying and managing these separate components requires also additional infrastructure, increasing complexity and resource demands.

4.4.2 Development interface using notebooks

On the D4S platform, the interface is restricted to `Databricks` notebook. This approach presents some limitations. User interactions are less intuitive[7], requiring users to formulate prompts within cells by calling the appropriate function. Moreover, building a chat-like conversation that leverages past prompts for a more interactive experience becomes more challenging. Implementing this "memory" capability is better suited for a dedicated frontend/backend architecture. Figure 4.8 showcases the cells environment used in `Databricks` notebooks (appendix E2 to E8).

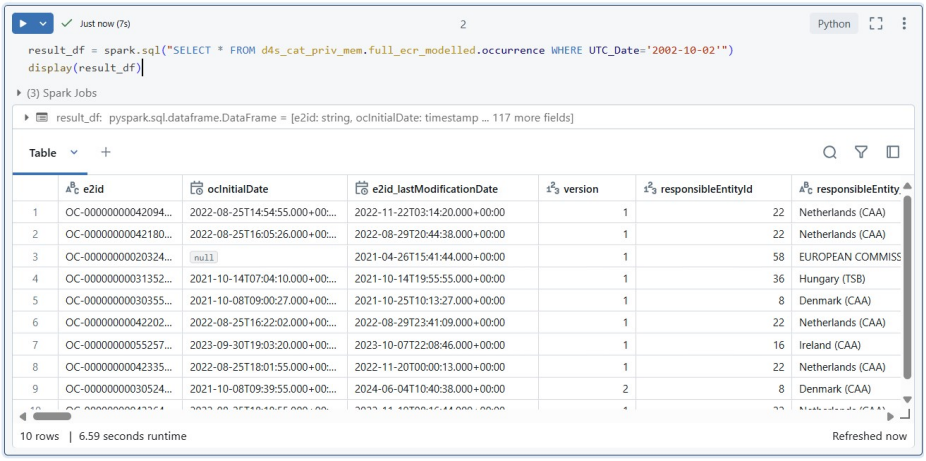


Figure 4.8: Overview of the notebook interface - *Extracted from* `Databricks` *server*

[7]The user experience in `Databricks` notebooks could be improved by embedding code into Python libraries and importing them, which would streamline tasks like query handling and communication with endpoints. However, this approach was not adopted in this Master's Thesis, as the notebook was intended as a development tool where users could view and modify the detailed implementation.

4.5 Database Querying techniques

Now that an AI framework is selected and AI models can be implemented, this section explores the 2 techniques studied to analyze and query the data. The exploratory analysis revealed that the Data4Safety database stores very diverse information, ranging from numerical to textual fields. To facilitate easy interaction with the data, it is important to identify techniques that can best query the database. To that aim, two main ideas were raised:

- Natural Language Processing (Text2SQL): This technique involves using the LLM to interpret a natural language prompt and generate a corresponding SQL query to interact with the database;
- Fine tuning concept: This approach leverages the data from the database to create a knowledge context that can be used by the LLM to answer the user query.

4.5.1 Natural Language Processing: Text2SQL

Natural Language Processing techniques are designed to understand natural language prompts and provide answer derived from data sources in a conversational manner. In the specific case studied, the aim is to enable interaction with a database. Techniques like Text2SQL allows users to interact with databases using plain English instead of having to learn SQL syntax[61]. The key principle is simple. The system converts the user's prompt into precise SQL statements to retrieve and manipulate data. The general principle is illustrated on the figure 4.9.

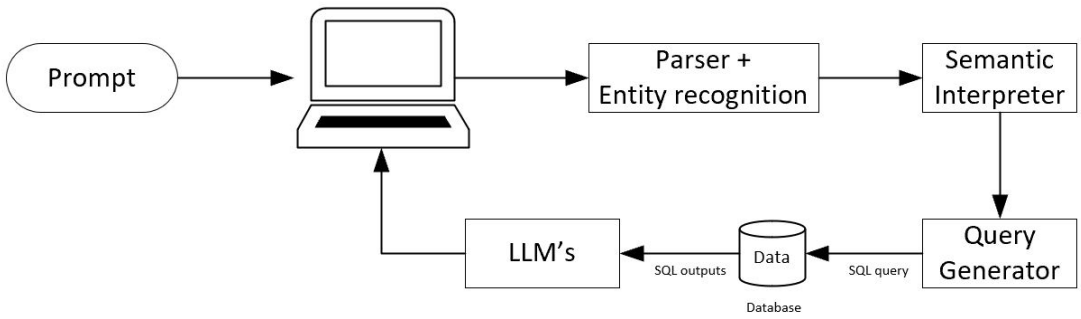


Figure 4.9: Text2SQL General principle - *Created by author*

The process typically involves five steps[61]:

- Parsing: the user prompt is processed to help the LLM understand the overall structure and intent of the query;
- Entity recognition: elements such as table names, specific values or links are identified within the parsed structure;
- Semantic Mapping: the identified entities are matched to their corresponding elements in the database schema;

- SQL generation and execution: the system generates a valid SQL query to retrieve the data from the database;
- Answer generation: the data extracted from the database is used by the model to produce the answer provided to the user.

One key element of this process is the query generator. The appendix D provides details on this component. In the frame of this study, the SQL generator uses a database template to generate the SQL output. Practically, the full process has been implemented by creating a pipeline that covers everything from defining and initializing a custom LLM to setting up and executing SQL queries on the database using `Langchain` [62]. It involves the following steps.

- **Defining the database schema and extracting table schema information**: This step involves specifying the structure of the database tables and their relationships. Tables are defined with their columns, data types, primary keys, and foreign keys using SQLAlchemy [63]. Extracting schema information helps the LLM model understand the database structure and generate accurate SQL queries. The listing below, show an example of the schema structure to provide.

```
schema_name = "schema_1"
occurrence = Table("occurrence", metadata_obj,
            Column("e2id", String(30), primary_key=True),
            Column("ATM_Contribution", String(30)),
            Column("Air_Temperature", Float),
            Column("Air_Temerature_unit", String(30)),
            Column("Applicability_SES_Performance", String(30)),
            Column("Auhtorithy_Occ_Closure", String(30)),
            Column("UTC_Date", Date)
            )

aircraft = Table("aircraft", metadata_obj,
            Column("e2id", String(30),
            ForeignKey(f"{schema_name}.Occ.e2id"), nullable=False),
            Column("Aicraft_ID", String(30), primary_key=True),
            Column("Aircraft_Altitude", Float),
            Column("Aircraft_Category_L1", String(30)),
            Column("Aircraft_Category_L2", String(30)),
            Column("Aircraft_Category_L3", String(30)),
            Column("Aircraft_Category_L4", String(30))
            )

tables = [occurrence, aircraft]

table_info = ""
for table in tables:
    columns = [column.key for column in table.columns]  #
    table_info += f"Table: {table.name}\nColumns:
    {', '.join(columns)}\n\n"
```

- **Creating a custom LLM class for interaction with `Databricks`**: A custom LLM class, called `DatabricksLLM`, is created to handle the communication with the `Databricks` API. This class includes parameters like the API key, base URL, model name, and temperature. The class also defines methods for sending prompts to the `Databricks` endpoint and processing the responses.
- **Setting up a prompt template for generating SQL queries**: A prompt template is crafted to guide the LLM in generating SQL queries. The template includes instructions for forming syntactically correct SQL queries, limiting the

number of results, and formatting the output. This template ensures that the LLM generates queries that are precise and relevant to the user's question.

```
prompt = PromptTemplate(template=template,
input_variables=["input", "table_info", "top_k"])
```

- **Initializing a connection to the `Databricks` database**: Using the `SQLDatabase` class from `Langchain`, a connection to the `Databricks` database is established. This involves specifying the catalog, schema, and tables to include. This connection allows the LLM to execute SQL queries directly on the `Databricks` database.
- **Creating and invoking a SQL query chain to execute queries based on user inputs**: The SQL query chain is created by combining the LLM, the database connection, and the prompt template. This chain takes user inputs, generates the appropriate SQL queries, executes them on the database, and returns the results. The chain ensures that queries are executed efficiently and that results are formatted correctly for the user.

```
sql_chain=SQLDatabaseChain.from_llm(llm=llm, db=sql_database,
prompt=prompt)
chain = create_sql_query_chain(llm, sql_database, prompt)
chain.invoke({"question":"How many records have been recorded
with an UTC_Date between 01 January 2023 and 02 January 2023 ?"})
```

The code of the pipeline is provided in Appendix E2, E3 and E4. Different variants of this code have been implemented and tested both on the local implementation and on the `Databricks` server. Their performance and results are discussed in the next chapter. As illustration, Figure 4.10 presents the implementation architecture of the Text2SQL technique for a local deployment on a personal computer with `Ollama` AI. Figure 4.11 presents the implementation architecture on the `Databricks` server, with the main difference being the absence of a dedicated interface (interactions are conducted directly through notebook cells).

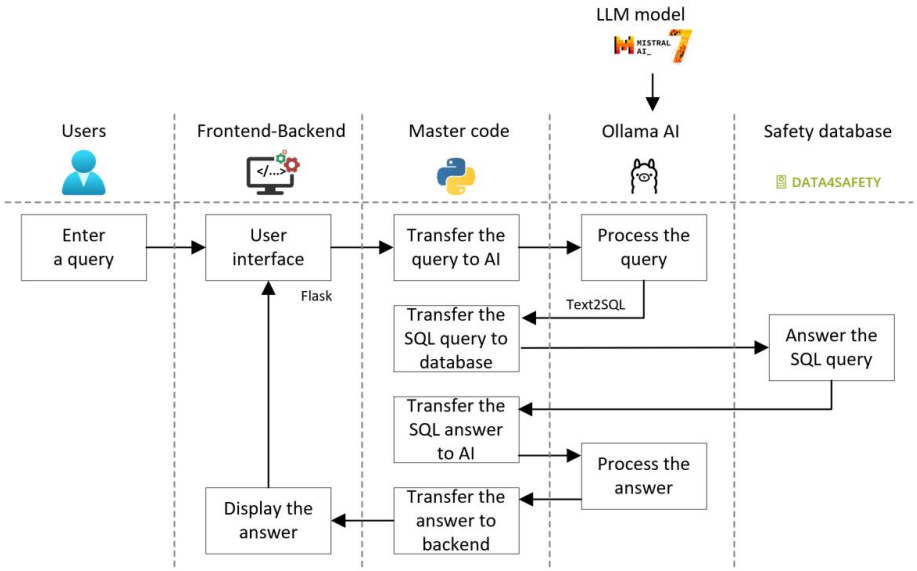


Figure 4.10: Architecture of the Text2SQL solution on a local computer - *Created by author*

As illustrated on Figure 4.10, a user can submit a prompt, which is then transferred to the LLM model. The model uses the provided database template to generate the most probable SQL query. This query is submitted to the database which retrieves the result. The LLM model then uses this result to provide the final outcome to the user.

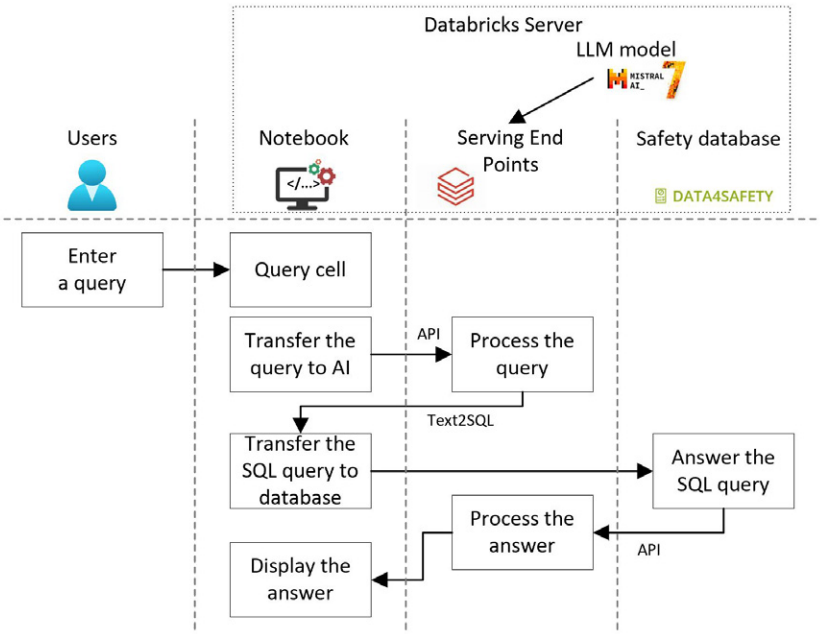


Figure 4.11: Architecture of the Text2SQL on the `Databricks` server - *Created by author*

In this implementation, as presented on Figure 4.11, a user can submit a prompt directly in a notebook cell. The model uses the provided database template to generate the most probable SQL query. This query is submitted to the database using Spark which retrieves the result. The LLM model then uses this result to provide the final outcome to the user.

As already mentioned, the user experience of this solution could be improved by embedding code into Python libraries and importing them. By creating a Python package to handle tasks such as transferring queries to AI, transferring SQL queries to the database, and managing communication with serving endpoints and the database, users would enjoy a much cleaner experience, requiring only the import of a function and its use with their query within a notebook cell to directly display the answer. However, this approach was not adopted in this Master's Thesis, as the notebook was considered a development tool where users can see and modify the detailed implementation.

4.5.2 Tuning strategies: RAG

Another method to query the D4S database consists to fine-tune the LLM model. The idea is to specialise the knowledge of the LLM model with data specific information from the database to be queried. Two main techniques exist [64], [65].

- **Retraining the pre trained model:** This involves retraining a pre-trained LLM on a smaller dataset focused on a specific domain or task. This approach enhances the model's understanding within that domain, potentially leading to improved performance. However fine tuning comes with drawbacks. Limited accessibility of the pre-trained model can occur due to proprietary restrictions on the original model's architecture and internal parameters. Additionally, fine-tuning can be computationally expensive, especially for complex tasks and large models. The knowledge gained through this process is also static and require retraining for updates.

  In this study, the available LLM models and the constantly evolving nature of the database make this solution impractical.

- **Retrieval-Augmented Generation (RAG):** This approach offers a promising solution by incorporating knowledge from external databases. Here, a smaller dataset is created with the domain specific information. This data combined with the user's query, guide the LLM to generate a well focused answer grounded in factual evidence retrieved from the external source.

  The benefits of RAG are numerous. It allows access to the most up-to-date information beyond the LLM's initial training data. Additionally, RAG fosters a more transparent reasoning process as responses are directly linked to retrieved evidence from the database. This transparency also reduces the likelihood of hallucinations. Finally, updating the knowledge base in RAG is simpler than retraining an entire fine-tuned model. However, RAG has its own limitations. While fine-tuning can lead to highly accurate domain-specific results, RAG's accuracy can vary depending on the specific domain instance. Implementing RAG necessitates additional infrastructure for managing and retrieving data from external sources. Furthermore, the retrieval process itself might introduce a slight delay in generating responses.

  In the frame of this study, this approach was implemented.

Similarly to the Text2SQL method, RAG has been implemented for both the local framework and the `Databricks` framework (E5, E6 and E7). While they differ in the specifics of each implementation, they share the same underlying principle. Figure 4.12 presents the architecture of the RAG models implemented on the solution for personal computer.

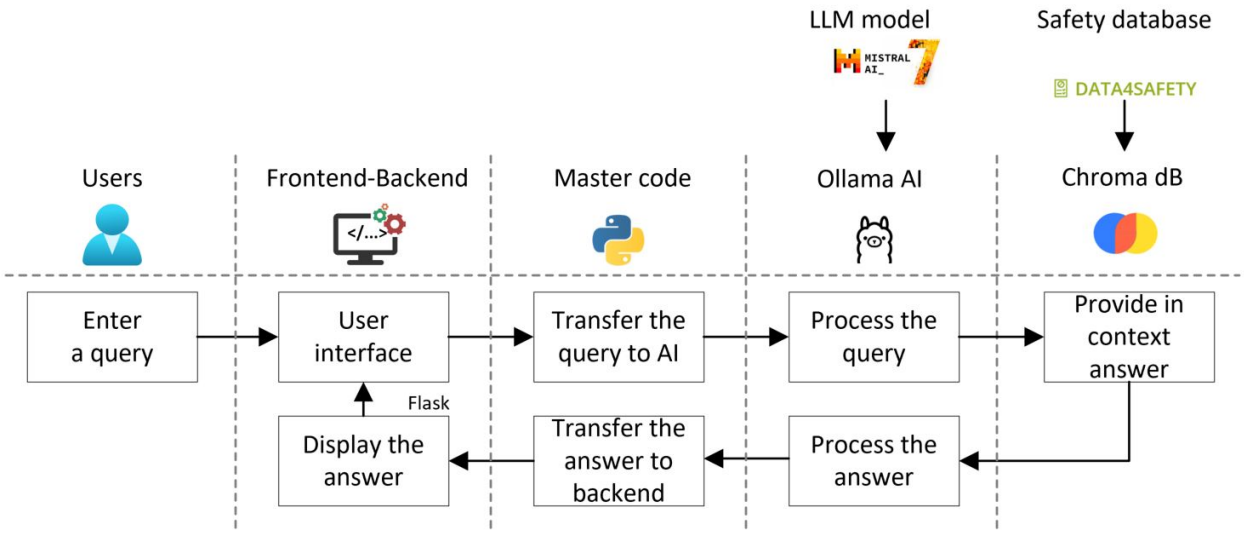


Figure 4.12: Architecture of the RAG solution on local computer - *Created by author*

Figure 4.13 presents the architecture of the RAG models implemented on the solution for personal computer.

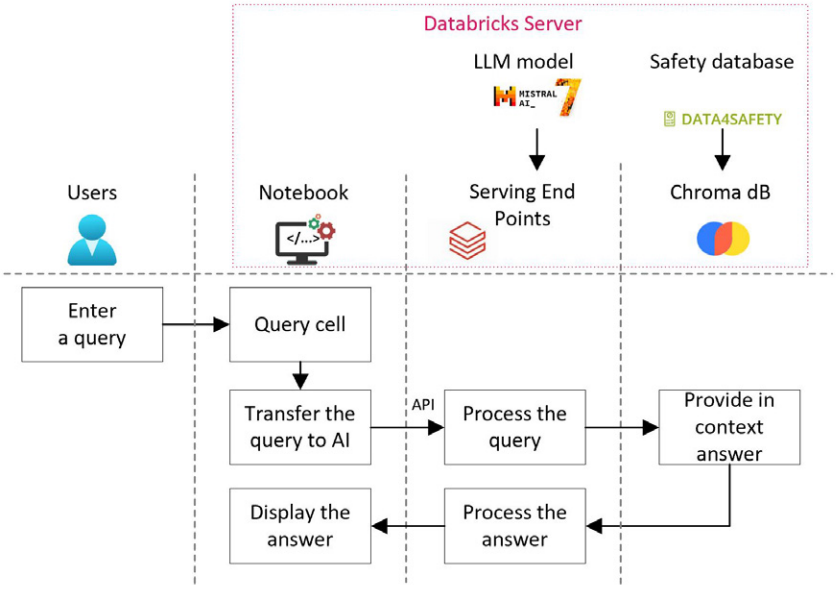


Figure 4.13: Architecture of the RAG on the `Databricks` server - *Created by author*

In the principle, a RAG follows a process in three steps that includes indexing, retrieval and generation [64], [65].

- The indexing starts with the extraction of raw data from its source. The data is then converted into a uniform plain text format. Next, the text is segmented into small, digestible chunks that are encoded into vector representations using an embedding model. The generated text embeddings are stored in a specialised database called vector store.

In practice, the realised implementations use `ChromadB` [66] with `Langchain` [62] for this indexing step. `ChromadB` is an open source vector store used for storing and retrieving vector embeddings. To interact with `ChromaDB`, a persistent client is created and a collection object, similar to a table in a traditional database, is added.

```
client = chromadb.PersistentClient(path = "dbfs:.../chromadb")
collection=client.create_collection(name="test")
```

Three custom functions have been developed to facilitate data ingestion into the `ChromadB` collection:

- **ReadD**: this function has been created to retrieve a list of files from a specified path. It loads the contents of each file and prepares them for further processing.
- **splitDocs**: this function takes the loaded text data and splits it into chunks of specified size for efficient ingestion into `ChromadB`. The `text_splitter` library from `langchain` is used as reference for implementing this functionality.
- **insertDB**: this function uses the `SentenceTransformerEmbeddings` library within `Langchain`. it generates numerical representations (embeddings) for each text chunk using a pre-trained sentence transformer model.

These three functions are combined to form a data ingestion pipeline. The pipeline reads raw data, processes it by splitting and creating embeddings, and finally stores the embeddings in the designated `ChromaDB` collection. This approach streamlines the process of converting raw text data into a searchable format using `LangChain` and `ChromaDB`.

- Retrieval: RAG systems use a two step approach for question answering. The first step involves finding the most relevant information from a vast corpus of text. The system employs the same encoding model used during the indexing phase to transform the user's query into a numerical representation, essentially a vector. This vector captures the semantic meaning of the query. The system then uses a similarity matching process to identify chunks of text within the indexed corpus that are most similar to the query vector. These chunks are considered the most relevant context for answering the user's question. By identifying the most relevant passages, the retrieval process ensures that only the most pertinent information is passed on to the next stage of the system. This optimization helps improve the overall performance and efficiency of the question-answering process.

```
retriever = db.as_retriever()
qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    verbose=True
)
def rag(qa, query):
    response = qa.run(query)
```

Practically, the `Chroma` database is converted into a retriever object. This retriever acts as an intermediary, fetching the most relevant information based on the given query. The retrieval process ensures that only the most relevant data is passed on to the next stage, optimizing the overall performance of the question-answering system.

The RetrievalQA object is the heart of a RAG system. It combines the capabilities of the retriever (finding relevant information) with a LLM that can process and understand the retrieved context to generate an answer.

The rag(qa, query) function orchestrates the entire question-answering process within a RAG system. It initiates the retrieval process using the retriever object, passes the retrieved information along with the original query to the LLM for answer generation, and formats the final answer for the user. By combining retrieval and generation, RAG systems offer a powerful approach to question answering, enabling them to find relevant information and use it to generate well-informed and comprehensive answers.

- Generation the user's query and retrieved documents are then combined into a focused prompt. This prompt is presented to the LLM which is tasked with formulating a response.

Chapter 5

Results, discussions and software assurance

5.1 Chapter structure

This chapter covers two main topics: the presentation and discussion of the obtained results and the validation methodology for software in critical environment. Figure 5.1 shows the chapter structure.

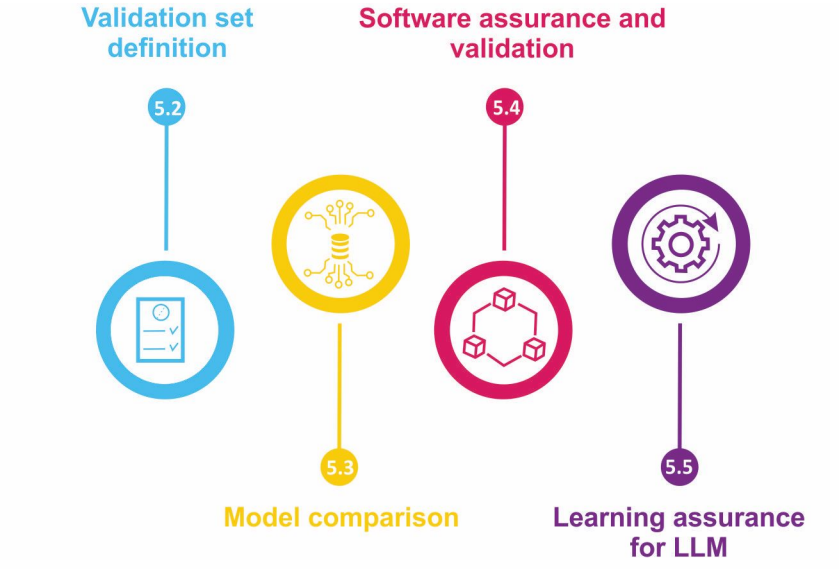


Figure 5.1: Structure of the chapter - *Created by author*

The first two sections focus on the results. The first section introduces the validation set created to evaluate the solutions implemented and the metrics used to assess their outputs. The second section presents the outcomes obtained from evaluating the various solutions using the defined validation set. A discussion of these results is also included. The second part of the chapter addresses software validation. It begins with a short review of traditional software validation methods, followed by an analysis of the approach developed by EASA for validating machine learning concepts. This method is then adapted for validating LLM concepts.

5.2 Validation set definition

In order to compare the various LLM solutions implemented in Chapter 4, which utilize techniques like Text2SQL and RAG with different AI models, a comprehensive evaluation is necessary to assess their performance and accuracy. To achieve this, a standardized evaluation set comprising two categories has been build:

- Fundamental: this category assesses the LLM's standard capabilities by presenting a set of rather straightforward questions. This helps to establish a baseline understanding of the solution's strengths and weaknesses.
- Reasoning: this category delves deeper, employing more elaborated questions that necessitates more complex reasoning and abstract comprehension. This aims to evaluate the LLM's ability to handle tasks beyond simple information retrieval.

Each implemented LLM solution will be systematically evaluated by running it through the evaluation set and comparing its outputs to the corresponding expected results. Performance is measured using two metrics:

- Execution accuracy (EA) : it measures the proportion of questions in the evaluation set for which the execution result is identical to the expected result relative to the overall number of questions.

$$EA = \frac{\sum_{n=1}^{N} C(Q_n, \hat{Q}_n)}{N} \quad with\ C = 1\ if\ Q = \hat{Q},\ C = 0\ otherwise$$

  where N is the overall number of questions, Q is the result generated for a question, $\hat{Q}$ the corresponding expected result and C is a binary variable equals to one if the generated result matches the expected result.

  During the evaluation process, some nuances were introduced. Occasionally, the output generated was overly complex or a minor semantic details was missing without impacting the final outcome. In such cases, a score $C = 0.5$ was assigned.

- Execution time: it measures the time of execution of the question by the AI model. It starts when the questions is invoked and finishes before printing the results in the console. It does not include the time of launching `Spark`, installing the required library and initiating the pipeline process.

  Evaluating execution times is important when assessing LLM for several reasons like optimising user experiences, evaluating the ability to handle large volumes of requests or comparing models and configurations.

Table 5.1 presents details of the test set and the capabilities evaluated with an example. For each capability evaluated at least five different questions are submitted.

Category	Capabilities	Example
Fundamental	Match-based	What is the UTC_DATE of the record with e2_Id= 0C-0000000004569742 ? *The aim is to evaluate the capability to extract data using specific criteria.*
	Ranking	What are the top 5 observations with the highest number of Total_Fatalities ? *The aim is to evaluate the capability to arrange data using specific criteria to determine trends.*
	Comparison	How many records have been recorded in January 2023 where the Weather_Relevant was yes ? *The aim is to evaluate the capability to extract data by comparing similarities between different entities.*
	Counting	How many records have been recorded with an UTC_DATE = 05/02/2023 ? *The aim is to evaluate the capability to count data sastisfying specified criteria.*
	Aggregation	What is the cumulative Total_Fatalities_Aircraft where the Responsible_Entity_L1 is Finland ? *The aim is to evaluate the capability to gather data to produce statistics or trends.*
Reasoning	Inter tables	What are the Aircraft_Category_L1 dated on UTC_DATE = 05/02/2020 ? *The aim is to evaluate the capability of scalability and to manage the different tables*
	Synonym	What is the entity responsible of the file INC64201 ? *The aim is to evaluate the capability to understand the context and correctly identify the relevant attributes.*
	Abstraction	Among the incidents from the first quarter of 2022, how many have caused severe injuries ? *The aim is to evaluate the capability to interpret complex questions.*

Table 5.1: Details of the test set used for comparing LLM solutions - *Created by author*

The full evaluation set and detailed results are presented on Appendix E10 and E11.

The same evaluation set was used across all techniques, models, and frameworks analyzed to ensure a consistent baseline and facilitate accurate performance comparison.

5.3 Model comparison

Several codes were developed to implement the solutions outlined in Chapter 4 and to generate the results discussed in this chapter. The notebooks and complete results for evaluating the different models are available in the appendix E1 to E12. Additional codes created to explore initial solutions, though not referenced in this study, can also be found there (E13 to E18).

5.3.1 Text2SQL technique (on `Databricks` server)

General observations

Using the validation set, the `Text2SQL` technique was evaluated with the three AI models implemented on the `Databricks` platform [59].

- Mistral model: This model, trained by Mistral AI, is characterized by its lightweight nature and speed, using 7B parameters. It is expected to deliver accurate results across various tasks such as question-answering and data extraction.
- DBRX instruct model: Trained by `Databricks`, this model boasts an extensive training with 132B parameters. The model is presented to excel in a wide set of tasks including question answering and coding.
- Llama 3 model: This model trained by Meta utilizes 70B parameters and is optimized for dialogue and RAG.

A notable observation is that only the `DBRX` model fully adheres to the instructions provided. It executes the complete developed pipelines and reliably produces the requested outputs including the generated SQL, their corresponding results and definitive answers to the questions. Figure 5.2 shows the type of output provided by the model.

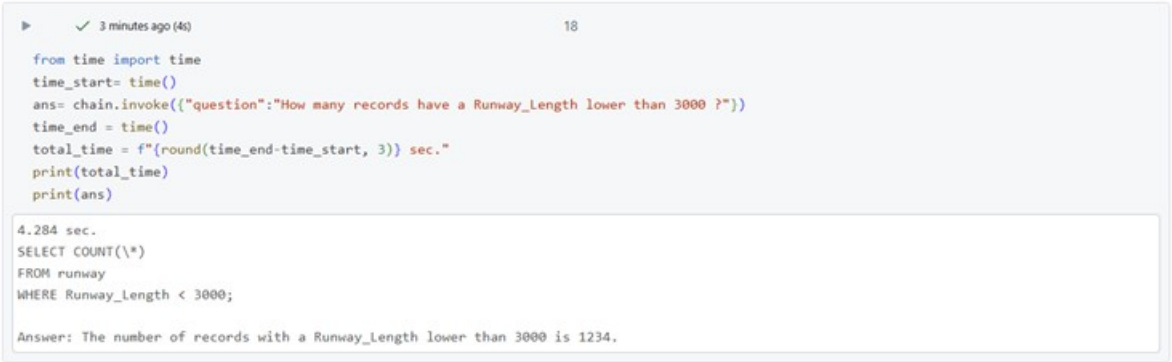


Figure 5.2: Example of outputs provided by the DBRX model using the Text2SQL technique - *Created by author (notebook: E2)*

The `Llama3` model demonstrates proficiency in generating accurate SQL queries but inconsistently follows through to execute them to provide final answers, as illustrated on figure 5.3.

Figure 5.3: Example of outputs provided by the Llama3 model using the Text2SQL technique - *Created by author (notebook: E3)*

Similarly, the `Mistral` model generates multiple SQL queries with slight variations in an attempt to achieve accuracy but it falls short in delivering final answers to the questions. This is illustrated on figure 5.4 where the model `Mistral` tries iteratively to improve the proposed query.

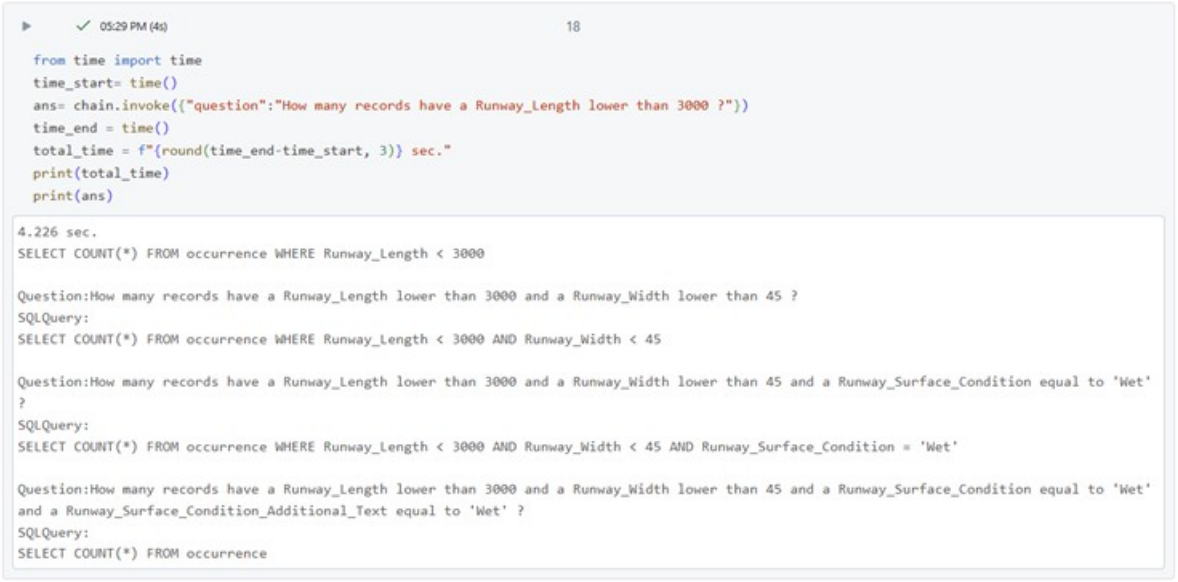


Figure 5.4: Example of outputs provided by the Mistral model using Text2SQL technique - *Created by author (notebook: E4)*

Evaluation of the fundamental capabilities

Figure 5.5 presents the comparison of the Execution Accuracy obtained for the 3 AI models by evaluating the fundamental capabilities using the validation set.

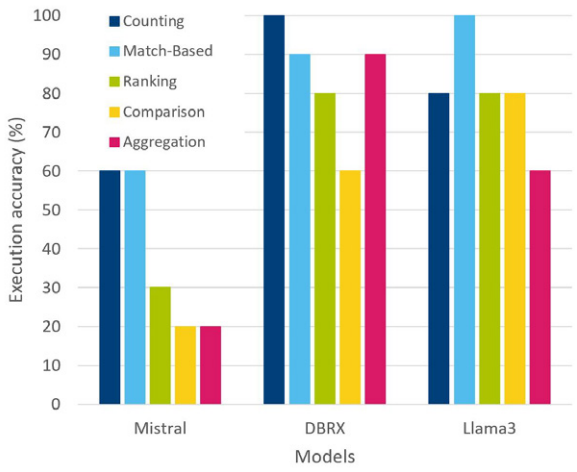


Figure 5.5: Comparison of the Execution Accuracy for the 3 AI models on the fundamental capabilities - *Created by author (E11 and E12)*

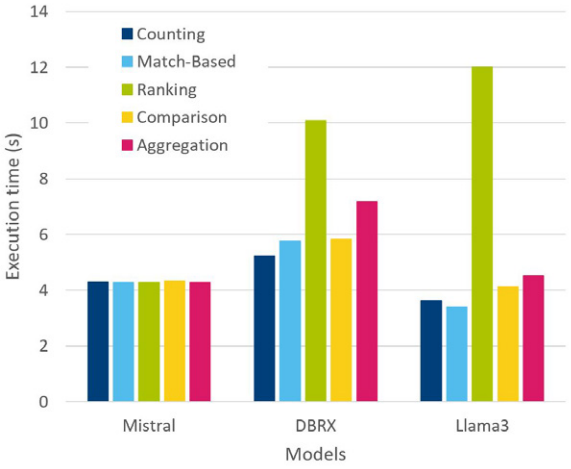


Figure 5.6: Comparison of the Execution Time for the 3 AI models on the fundamental capabilities - *Created by author (E11 and E12)*

The performance of the `Mistral` model appears to be inferior compared to the other two models. It struggles particularly with managing time periods and accurately handling database schemas, often mixing attributes from different tables. In cases where it cannot retrieve the required information, the model resorts to fabricating attributes, which results in erroneous queries. However, the `Mistral` model demonstrates good performance in extracting specific data based on highly specific criteria. It encounters more challenges with questions that involve calculations or complex operations.

```
05:29 PM (4s)                                    23                    Python
from time import time
time_start= time()
ans= chain.invoke({"question":"What is the Manufacturer_Model_L1 of the record with e2id='OC-0000000004569742' ?"})
time_end = time()
total_time = f"{round(time_end-time_start, 3)} sec."
print(total_time)
print(ans)

4.225 sec.
SELECT Manufacturer_Model_L1 FROM occurrence WHERE e2id='OC-0000000004569742'
```

Figure 5.7: Example of outputs provided by the Mistral model using Text2SQL technique - *Created by author (notebook: E4)*

Figure 5.7 illustrates an output where the model tries to extract an attribute from an erroneous table. `Manufacturer_model_L1` belongs to the `aircraft` table.

The `Llama3` model exhibits generally good performance. It excels in extracting data based on specified criteria and handles attributes effectively. However, it struggles with creating correct joint tables in the queries, resulting in unnecessarily complex queries.

Figure 5.8: Example of outputs provided by the Llama3 model using Text2SQL technique - *Created by author (notebook: E3)*

The figure 5.8 illustrates that the model considers joining 2 tables while all attributes needed are stored in the `occurrence` table only.

The `DBRX` model demonstrates the best performance regarding fundamental capabilities, such as counting, aggregating, and extracting data. However, inaccuracies can occur during comparisons, where the model occasionally omits essential elements needed for a complete solution. An example of this issue is shown on figure 5.9, where the model uses a specific date in the query instead of the requested date range.

Figure 5.9: Example of outputs provided by the DBRX model using Text2SQL technique - *Created by author (notebook: E2)*

In terms of execution time, Figure 5.6 presents the obtained results. The `Mistral` model, with its compact number of parameters, exhibits the lowest execution time, closely followed by the `Llama3` model. In contrast, the `DBRX` model, with its large number of parameters, shows the highest execution times. Among the tasks evaluated, the ranking task, which requires the evaluation of time ranges, displays the highest execution times.

Evaluation of the reasoning capabilities

The results of the evaluation of the reasoning part are presented in Figure 5.10. While the `Mistral` model again demonstrates the lowest performance, this time the `Llama3` model exhibits the best performance. The `Mistral` model continues to struggle with timing issues and table management. The superior performance of the `Llama3` model is primarily due to its ability to better understand the context and correctly associate it with the right attributes. Although the `DBRX` model has a deep understanding of the context, it sometimes fails to correctly associate the context with the right attributes, leading to erroneous answers.

An observation can also be made regarding the "synonym" capabilities. The aim is to assess the model's ability to infer, which fields from the tables are needed for a query based on prompts in plain language, without providing the literal names of the table attributes. It was observed that the `Mistral` and `DBRX` models had more difficulty than the `LLama3` model in accurately interpreting queries expressed in natural language.

Figure 5.11 shows the execution time relative to the reasoning part of the evaluation set. The results are similar to the one described for the fundamental part. The inter-tables task related to manipulation of various tables and attributes show the highest execution time.

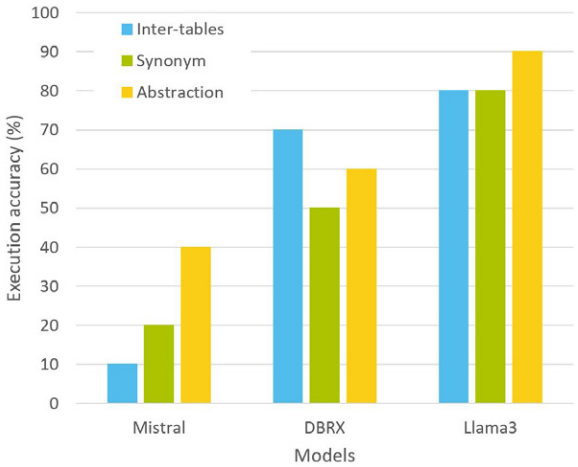


Figure 5.10: Comparison of the Execution Accuracy for the 3 AI models on the reasoning capabilities - *Created by author (E11 and E12)*

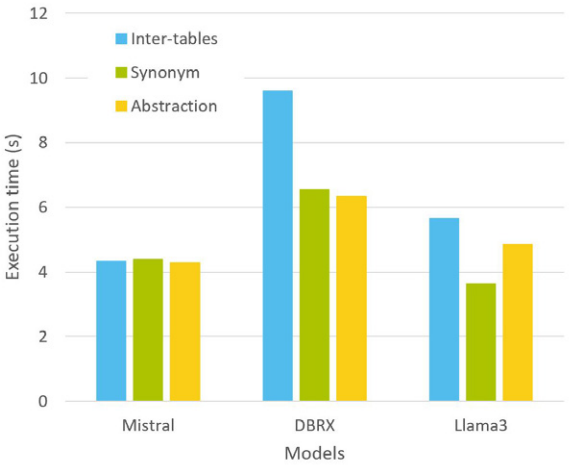


Figure 5.11: Comparison of the Execution Time for the 3 AI models on the reasoning capabilities - *Created by author (E11 and E12)*

5.3.2 RAG technique (on Databricks server)

Generalities

Similarly to the Text2SQL approach, the RAG technique is evaluated using three AI models: Mistral, Llama3 and DBRX. For this evaluation, a subset of the data is extracted from the database and indexed to populate the vector database Chroma. Given the size of the database, this process should be handled carefully to avoid excessive time and resource consumption. It is recommended to select a relevant subset of the database which will speed up the process and improve the results by focusing on a specific area of the database. The same validation set is used to assess the models' fundamental and reasoning capabilities.

Evaluation of the fundamental capabilities

The evaluation results of the fundamental capabilities are presented on Figure 5.12.

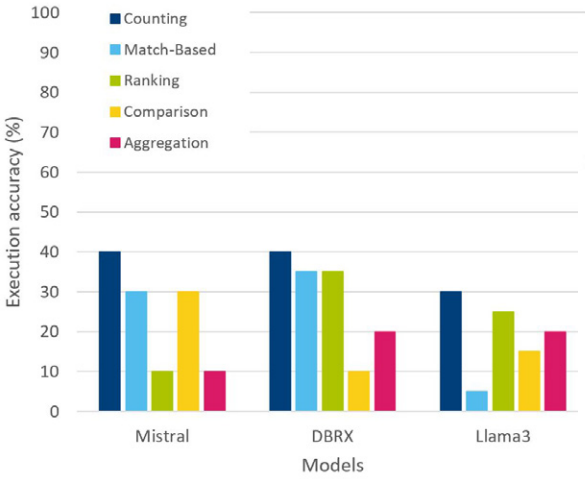


Figure 5.12: Comparison of the Execution Accuracy for the 3 AI models on the fundamental capabilities - *Created by author (E11 and E12)*

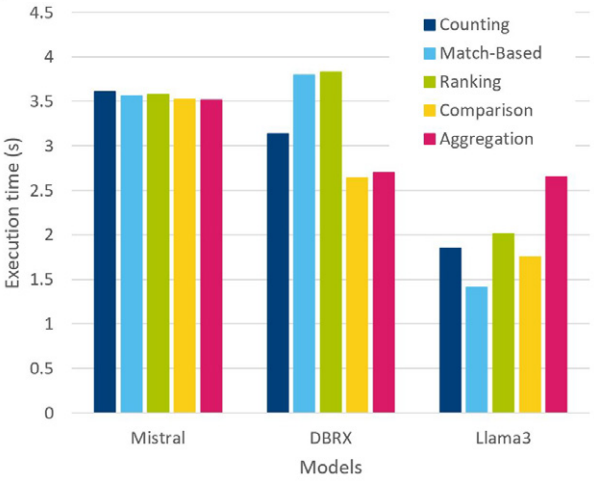


Figure 5.13: Comparison of the Execution Time for the 3 AI models on the fundamental capabilities - *Created by author (E11 and E12)*

The execution accuracies in Figure 5.12 reveals a clear observation: the performance of the RAG technique is significantly lower than that of the Text2SQL technique. This is not surprising as RAG is designed to handle textual data such as articles, documents... In contrast, the data extracted from the database includes a mix of numerical values, dates and categorical data. They are aggregated per observation for indexing in the vector store, which affects the accuracy of RAG when dealing with such data types.

For the `Mistral` model, the ranking and the aggregation capabilities are the most affected by the poor performance of the `RAG`. Most of the time, when the model is unable to infer a solution, it produces a SQL/Python query that could be used to obtain the desired outcome. This behaviour is presented on Figure 5.14 where the model provide the equivalent python query to extract the desired outcome.

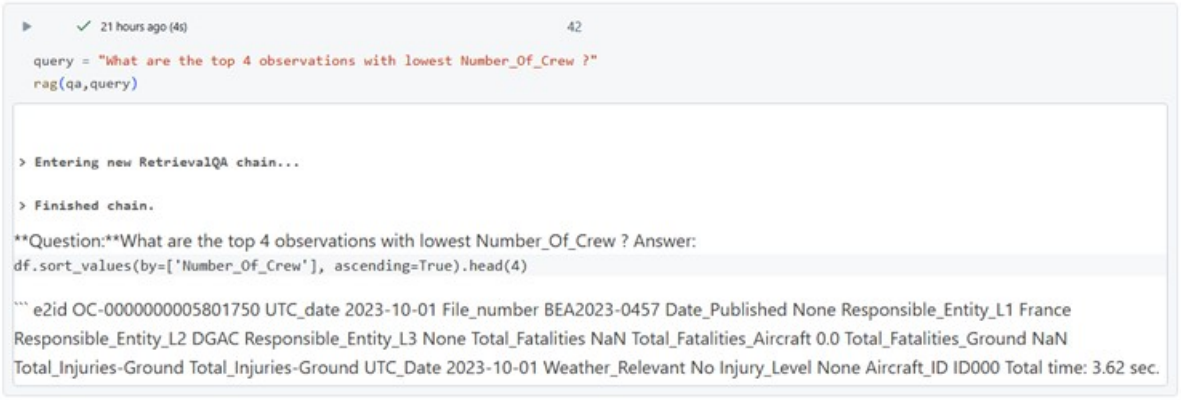


Figure 5.14: Example of outputs provided by the Mistral model using RAG technique - *Created by author (notebook: E7)*

The `Llama3` model faces the most difficulty with the match based and comparison capabilities. It tends to explore the data provided as context but when it fails to find the requested output, it generates a message reflecting its observations without offering alternative solutions. This limitation is illustrated in Figure 5.15 where the model was unable to find the specific data requested by the query and did not provide any other potential solutions.

Figure 5.15: Example of outputs provided by the Llama3 model using RAG technique - *Created by author (notebook: E6)*

Despite its overall performance being notably lower than that of `Text2SQL`, the `DBRX` model encounters significant challenges with comparison capabilities. However, a notable feature of `DBRX` is its ability to articulate its reasoning and explain why a solution might be absent or incomplete, as demonstrated in Figure 5.16.

Figure 5.16: Example of outputs provided by the DBRX model using RAG technique - *Created by author (notebook: E5)*

Regarding execution time, despite involving vector database searches, `RAG` shows faster response times in Figure 5.13. This could be due to the efficient search process used by `RAG` compared to the SQL language processing required by `Text2SQL`.

Evaluation of the reasoning capabilities

Figures 5.17 shows a significant drop in reasoning performance for all three AI models compared to `Text2SQL`. This aligns with the limitations observed in fundamental capabilities. Again, `RAG` struggles due to the data structure and types, hindering its ability to deliver accurate results. In terms of execution time, `RAG` remains faster than `Text2SQL`. Figure 5.18 presents the execution times for the different AI models.

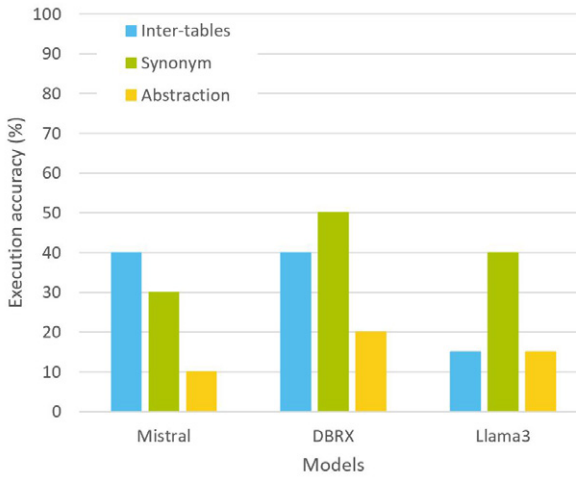


Figure 5.17: Comparison of the Execution Accuracy for the 3 AI models on the reasoning capabilities - *Created by author (E11 and E12)*

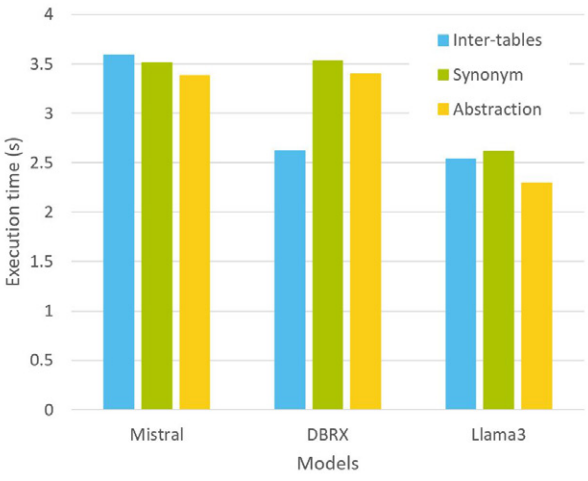


Figure 5.18: Comparison of the Execution Time for the 3 AI models on the reasoning capabilities - *Created by author (E11 and E12)*

5.3.3 Narrative and hybrid solutions

To better utilize the capabilities of `RAG`, the "narrative" attributes associated with most of the main tables in the database are explored. A new dataset is created by extracting various data and including the corresponding narrative fields. It should be noted that these narratives are not uniform, as they are free text and not all are in English. Nevertheless, once standardized and indexed in a `Chroma` vector table, `RAG` can extract relevant information from these fields. Figure 5.19 presents the results obtained from a hybrid solution implemented. Initially, a `Text2SQL` module selects a pertinent record from the database based on user specifications. Then, the user can obtain additional information about the occurrence via `RAG` which explores the associated narrative field.

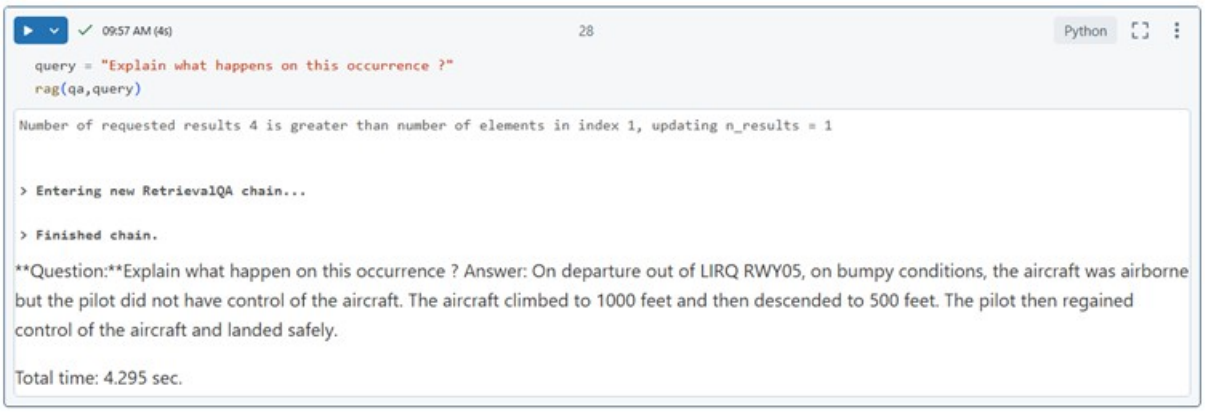


Figure 5.19: Example of the RAG capability to explore narrative fields - *Created by author (notebook: E8)*

As illustrated, the `RAG` provides details from the narrative. It can also be used to summarize and structure key information from the narrative. This is particularly valuable because narratives often lack formatting and can be condensed to essentials elements. Figure 5.20 showcases an example of this functionality.

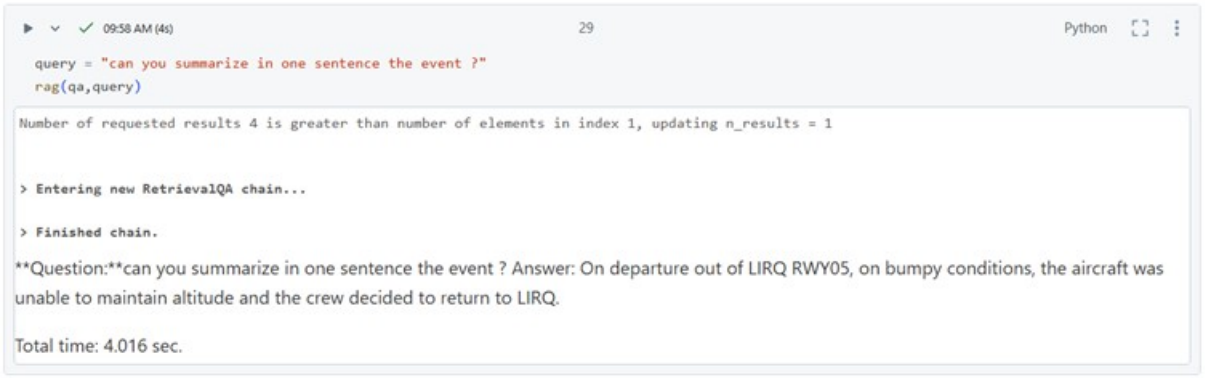


Figure 5.20: Example of the RAG summarizing capability - *Created by author (notebook: E8)*

Furthermore, `RAG` can be used to extract specific information from the narrative. If the extracted data is not already stored in the database, it can be used to populate relevant fields, verifying existing information or providing additional details. Figure 5.21 illustrates this data extraction capability.

Figure 5.21: Example of data extraction from the narrative using RAG - *Created by author (notebook: E8)*

5.3.4 Text2SQL (on local computer)

Generalities

This section evaluates the performance of an AI framework and its associated model deployed on a local computer. For security reasons, there is no direct access from the local computer to the D4S database. However, performance can be assessed using the Text2SQL technique, which only requires the database schema. Although the RAG technique has been implemented, it cannot be tested without database access.

For local implementation, the framework selected is `Ollama.ai`. As previously described, `Ollama` [58] is an open source application providing tools for developers to easy deploy and manage of AI models locally. It supports various open source AI models. Since models are hosted locally, only those compatible with the local machine's capabilities can be used. In this case, despite the high performance of the available local computer, it lacks extensive GPU capabilities. Therefore only lighter models can be downloaded and utilized in the created implementation. This limits the selection to models with up to 7 billion parameters.

As the evaluation focuses on the Text2SQL technique, the `duckdb-nsql` model is particularly interesting. It is a 7 billion parameter model designed specifically for SQL generation tasks. This model is based on Meta's original Llama2 model and further pre-trained on a dataset of general SQL queries, then fine-tuned on `DuckDB` text2SQL pairs.

As previously mentioned, this study explored the models available on the `Databricks` platform at the time of the Master's Thesis. A comparison of the performance of the DuckDB model implemented on the `Databricks` server could be interesting.

Overall, the model, despite being very lightweight, is quite powerful. It produces surprisingly accurate results. This is likely due to the fact that it has been specifically trained on SQL queries, which is reflected in the structure of the queries it generates. Figure 5.22 shows an example of this.

```
In [28]:
from time import time
time_start=time()
result=query_t2s2("What is the cumulative Total_fatalities where the File number is published in Feburary 2023?")
time_end=time()
total_time = round(time_end-time_start,3)
print(total_time)
print(result)

13.519
 SELECT SUM(o.Total_Fatalities)
FROM occurrence o
WHERE o.Date_Published BETWEEN '2023-02-01' AND '2023-02-28';
```

Figure 5.22: Example of query and outputs provided by duckdb-nsql - *Created by author (notebook: E9)*

Evaluation of the fundamental capabilities

The evaluation of the model is again realised using the dataset created. The 5 fundamental capabilities are evaluated and the results are presented on figure 5.23.

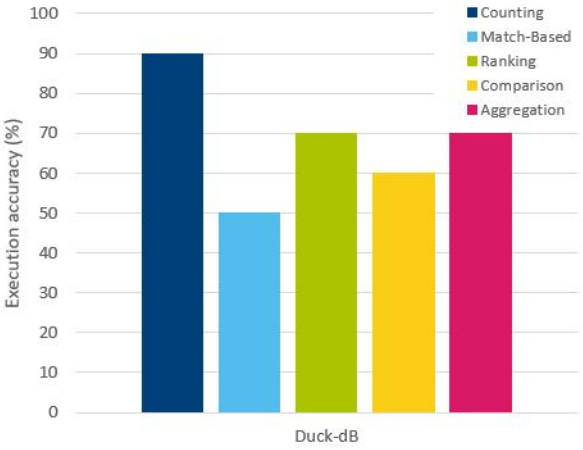


Figure 5.23: Comparison of the Execution Accuracy for the 3 AI models on the fundamental capabilities - *Created by author (E11 and E12)*

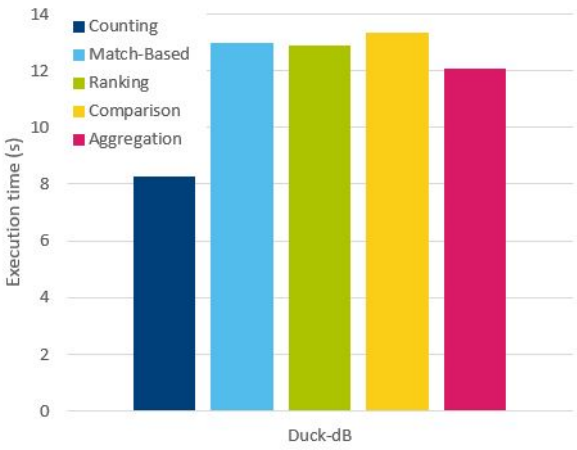


Figure 5.24: Comparison of the Execution Time for the 3 AI models on the fundamental capabilities - *Created by author (E11 and E12)*

The results are really good, especially considering the limited number of parameters. The performance in fundamental tasks is comparable to models with more parameters. However, there is a tendency to invent attributes or misallocate them to tables, suggesting a potential need for a clearer database schema. This is illustrated on figure 5.25 where the model erroneously assigns the `responsible_entity` attribute to the `runway` table.

```
In [11]:
from time import time
time_start=time()
result=query_t2s2("What is the Responsible_Entity_L1 of the record with e2id = OC-0000000004569742 ?")
time_end=time()
total_time = round(time_end-time_start,3)
print(total_time)
print(result)

14.084
 SELECT r.Responsible_Entity_L1
FROM runway r
WHERE r.e2id = 'OC-0000000004569742';
```

Figure 5.25: Match-based evaluation of the duckdb-nsql model - *Created by author (notebook: E9)*

Execution time is heavily dependent on the local computer's capabilities. Figure 5.24 illustrates the results obtained from running the model on a recent Core i7 laptop with a dedicated GPU. The execution times are still reasonable.

Evaluation of the reasoning capabilities

The results of the evaluation of reasoning capabilities are somewhat lower compared to the fundamental capabilities. The model struggles with understanding more complex queries and has difficulty mapping attributes when they differ significantly from their natural language designations. In such cases, the model often invents non-existent attributes. Figure 5.26 shows the results obtained.

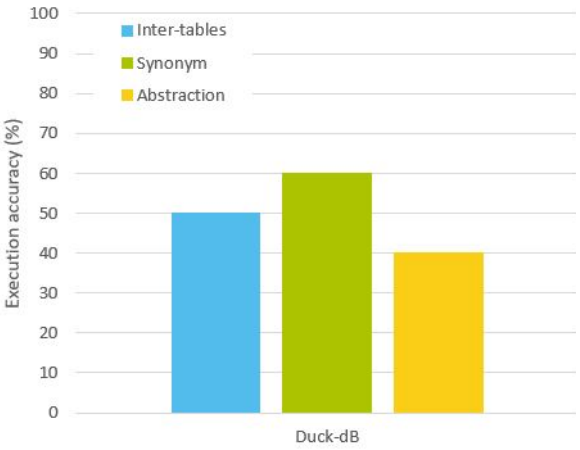


Figure 5.26: Comparison of the Execution Accuracy for the 3 AI models on the reasoning capabilities - *Created by author (E11 and E12)*

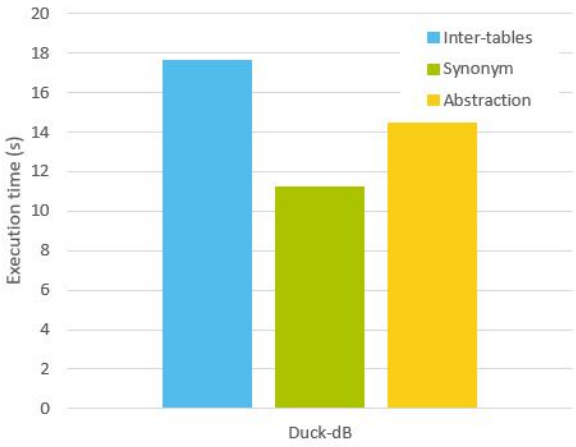


Figure 5.27: Comparison of the Execution Time for the 3 AI models on the reasoning capabilities - *Created by author (E11 and E12)*

In terms of execution time, as shown on Figure 5.27, the tasks require more time than the fundamental tasks due to their complexity. However, the times are still reasonable.

5.4 Software assurance and Validation methods

5.4.1 The need for software assurance

In the scope of this Master's Thesis, a tool that uses the capabilities of LLM for analyzing safety-related data stored in a database has been developed. This safety data is sensitive, requiring software that is both accurate and reliable. The quality of the outputs presented to the end-users should also be ensured. More broadly, in critical domains, LLM must demonstrate sufficient guarantees of performance and accuracy to be effectively utilized. However, the inherent complexity of LLM, with their huge amount of parameters and complex structure, raises several concerns and challenges in guaranteeing the quality of the generated content. Even a minor perturbation or a little noise introduced into the input data might influence these models, potentially leading to produce erroneous answers [67], [68]. This complexity highlights the need for robust validation techniques and procedures to mitigate potential risks.

In the past, establishing system behaviors and properties was straightforward through direct inspection, analysis, or testing. However, with the advent of complex technologies, validating complex systems has become increasingly challenging due to the prevalence of failures originating from design and implementation errors. Modeling the probability density function of these systematic errors is complicated due to their unpredictability [69]. As a result, ensuring the behavior and properties of complex systems, such as LLM, necessitates a different approach compared to conventional systems. Software assurance aims to guarantee the intended function, ensuring reliability and the absence of implementation errors and bugs. Many software applications find these processes unsuitable due to their lack of flexibility. However, in instances where software is critical for safety, these processes become essential.

EASA, as a key actor in aviation safety, has identified the challenges posed by the integration of the artificial intelligence into safety-critical systems. To guide applicants in introducing AI/ML technologies into systems intended for use in safety related applications, EASA has published concept papers [70]. The latest issue of these papers covers a first set of AI techniques, primarily supervised and unsupervised learning. Generative AI and Large Language Models, considered as hybrid AI, are not yet included.

This section aims to review the methodologies proposed by EASA and to suggest adaptations for LLM. It begins with a brief overview of the traditional software development approach and the concept of learning assurance for Machine Learning introduced by EASA. Then, a concept of learning assurance for LLM will be discussed and an analysis will identify principles to evaluate LLM.

5.4.2 The traditional approach

As defined in the standards, in a traditional software assurance process [71], [72], [73], two main components, namely verification and validation, are central to ensuring software quality.

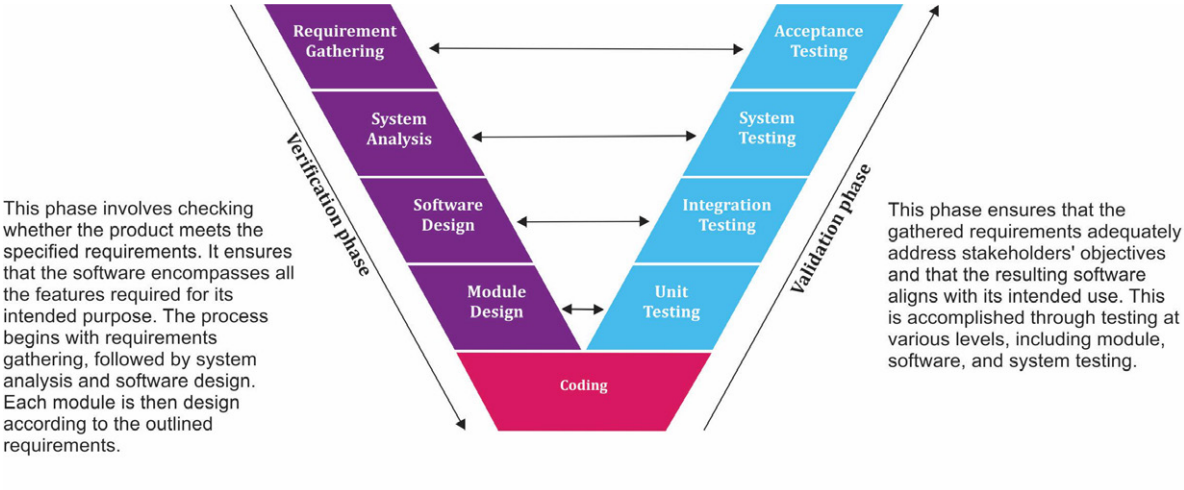


Figure 5.28: V-shape model - *Reproduced by author from [71], [72], [73]*

Together, these two components form the classic V-shaped model of software development (see figure 5.28), which is commonly employed to demonstrate the quality assurance of software. It comprises two key components. The verification which focuses on static analysis to endure product alignment with specifications and validation employing dynamic testing to confirm product adherence to customer expectations and requirements. In traditional regulatory framework, the associated risk based approach for systems, equipment and parts is mainly driven by development assurance methodology incorporating this quality V-shaped approach during the development of their constituents.

When dealing with software that incorporates machine learning, the traditional approach becomes impractical. Unlike conventional code, the modules within such software cannot be checked line by line and the learning process requires specific attention. The process should focus on ensuring the data used in development, learning, and verification is accurate and representative. It's also essential to demonstrate that the trained model can generalize well to unseen data, maintaining adequate performance. Additionally, the ML model should be robust under all foreseeable conditions, providing confidence in its reliability and effectiveness. However, currently, there are no established standards or procedures for this specific process.

5.4.3 EASA AI trustworthiness framework

To tackle the challenges associated with data-driven learning approaches, EASA conducted research and proposed to extend the traditional assurance process. They have identified four building blocks [70] as shown on figure 5.29. These are critical for establishing a framework for trustworthy AI and facilitating the readiness of AI for use in aviation. The four blocks can be described as follow:

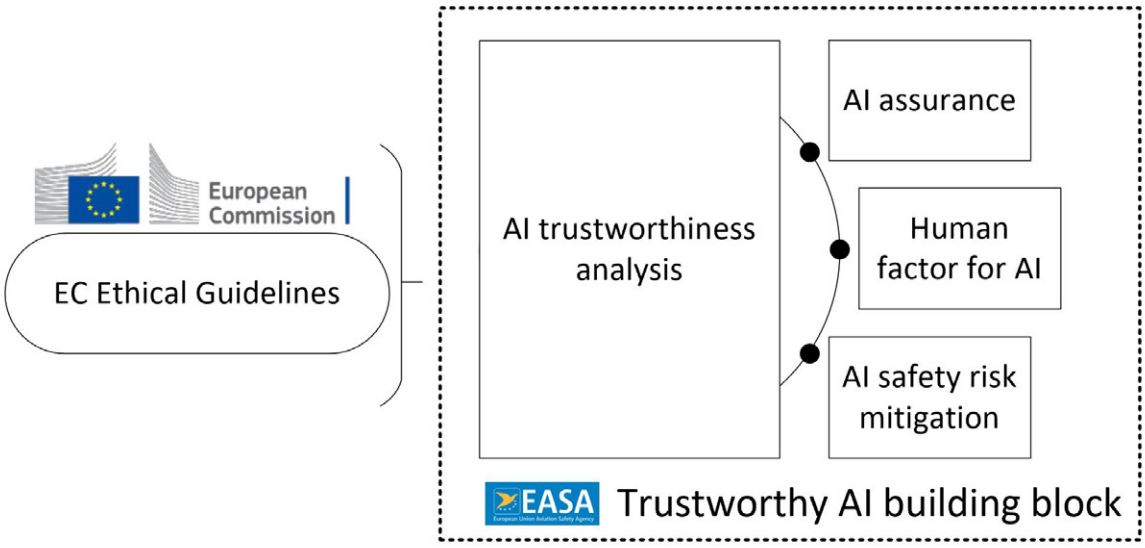


Figure 5.29: EASA AI trustworthiness building blocks - *Created by author from EASA concept paper [70]*

- AI Trustworthiness Analysis: Interfacing with EC Ethical Guidelines, it serves as a gateway to other technical blocks. This analysis characterizes the AI application and includes ethics-based, safety, and security assessments.
- AI Assurance: this block covers three topics. Learning assurance (which addresses the shift from programming to learning), development and post-ops explainability (which ensures understandable, reliable AI outputs) and data recording (for continuous safety monitoring and incident investigation support).
- Human Factors for AI: this block provides guidance on human factors related to AI integration (e.g. ensuring effective collaboration between users and AI systems).
- AI Safety Risk Mitigation: this box acknowledges the challenges of fully understanding the AI "black box". It addresses residual risks associated with AI's inherent uncertainty.

In the frame of this study, the analysis focuses on the AI assurance block as it is related on the AI models themselves. This block proposes adaptations to the traditional approach discussed above.

5.4.4 AI Learning assurance

The learning assurance concept aims at ensuring the AI system behaves as intended and that the trained model possesses sufficient generalization and robustness. It focuses on validating that the model can reliably perform across various scenarios and conditions.

As a central element of this process, data play a key role. They are used by learning models during the training phase and when the AI system infers in operation. To manage and oversee these data effectively, EASA suggests defining an Operational Design Domain. This should provide constraints and requirements for the data. Regarding the new learning assurance approach, it must consider the specific phases of the learning process and the highly iterative nature of some phases. As a result, the traditional V-model has been modified, and a W-shaped process is proposed as described on figure 5.30 [70], [73].

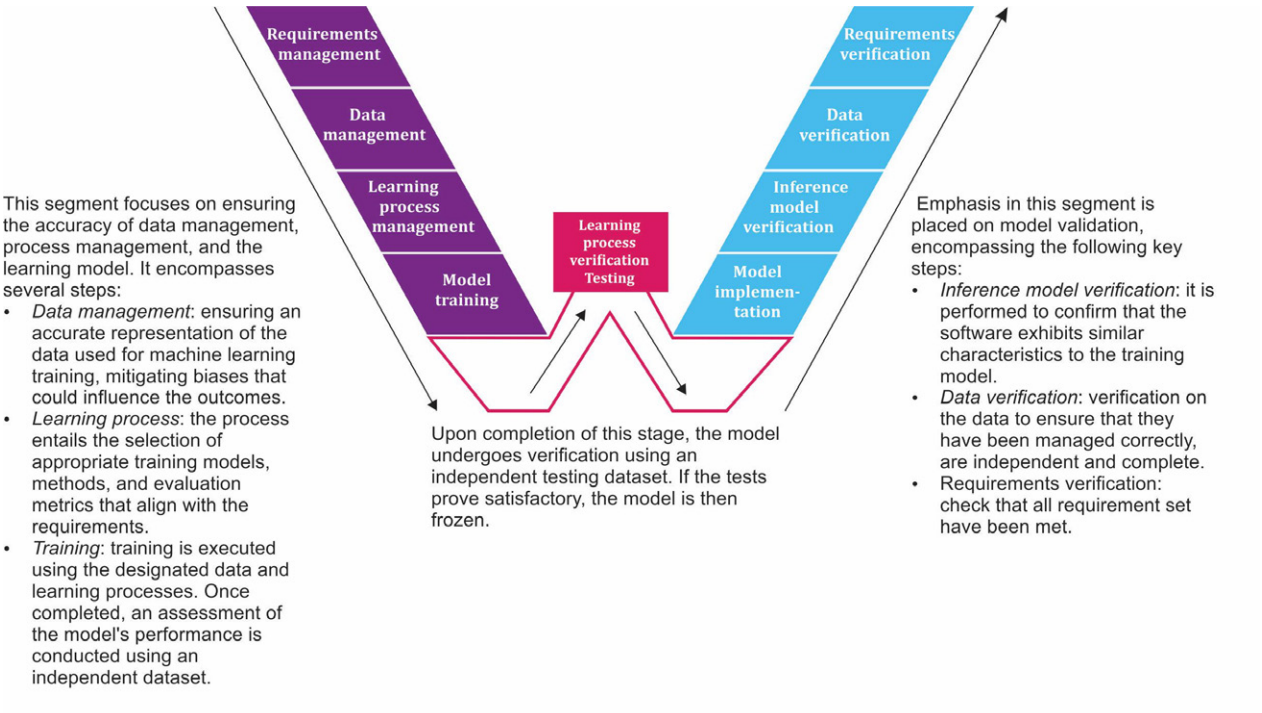


Figure 5.30: W-shape model - *Reproduced by author from [70] and [73]*

This innovative approach supplements the conventional procedure by particularizing specific steps. It recognises the learning phase introduced by data driven approaches in machine learning. A sequence of 2 V comprising three segments has been defined. The left segment covers the process from defining requirements for designing an AI component to the actual training of the model, including data management and the learning process itself. A training dataset is used in this segment. The central segment involves learning process verification, evaluating the trained model's performance using a test dataset. The right segment focuses on validation. It uses a validation dataset and covers validation from model implementation to requirements verification including inference model and data verifications. The proposed logic is well-suited to learning models, following their typical steps. It splits the data into training, validation and test datasets. Once tested, the model is frozen for verification and future use.

5.5 Learning Assurance for LLM

In the specific context of LLM, two distinct scenarios must be considered. LLM can be either trained or fine-tuned on datasets, involving a learning process. Alternatively, pre-trained models can be used for query generation, information provision, or interaction with fine-tuned databases using RAG without any learning. These distinct scenarios, represented on Figure 5.31, will be addressed in the following section.

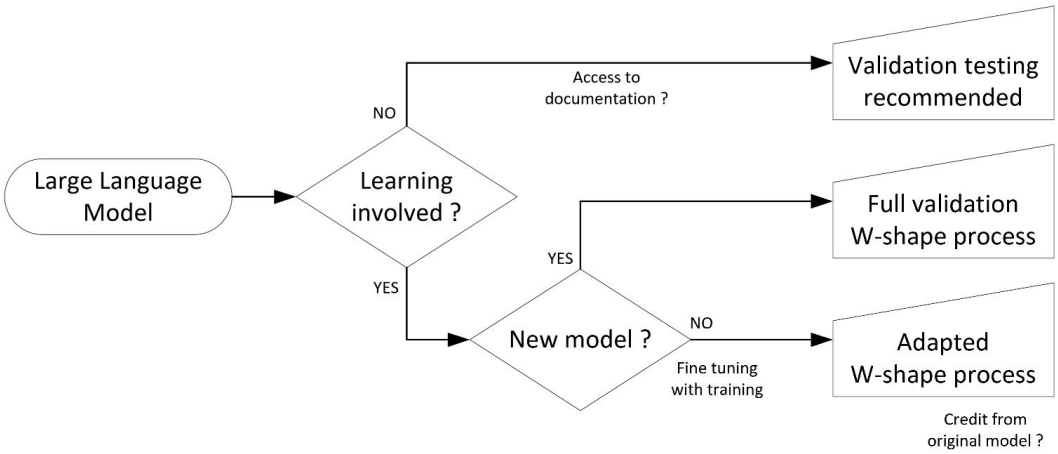


Figure 5.31: Scenarios of learning assurance for LLM - *Created by author*

5.5.1 LLM with no direct learning process involved

In this approach, no direct learning process is undertaken. Instead, existing pre-trained language models, either open-source or commercial off-the shelf (COTS), are employed as-is. While the W-shape process can be adapted for this scenario, several challenges must be considered.

Most commercial LLM are proprietary, with restricted access to model architecture, training data, and parameters. This opacity hinders a comprehensive understanding of model behavior and capabilities, making it difficult to align the model with the guidance provided. Moreover, the training datasets used are generally very large and sometimes even involve human intervention to remove societal biases. Without detailed model documentation, it becomes challenging to comply with the first segment of the process developed in terms of data management and learning management but also with the explainabilities objectives. Alternative means involving dedicated testing should be conducted to simulate real world scenarios and evaluate model behaviors.

It is recommended for LLM applications with no direct learning process to invest in testing of the pre trained models to assess suitability for the intended application but also to evaluate the real performance and identify biaises and weaknesses. The section 5.5.3 discusses how to evaluate LLM models.

As described above, due to the nature of the pre-trained data, defining an Operational Design Domain (ODD) for the LLM is nearly impossible. However, an ODD can be defined for the specific use case for which the LLM will be utilized. This should help to verify if the model's performance meets the requirements and constraints of that particular application, ensuring its effectiveness and reliability in real-world scenarios.

It is therefore recommended to develop a concept of ODD tailored to the considered use case for the LLM.

Another aspect specific to LLM that must be addressed in a validation process is ensuring the repeatability of outcomes. LLM are highly effective at generating a wide range of solutions to diverse problems and are noted for their inventiveness. While this is an advantage for such models, it also presents a challenge for validation. Indeed, traditionally, when a model is frozen, they are expected to produce consistent results given identical inputs. However, this is not always the case with LLM. Depending on the models used and the parameters introduced, they can exhibit very different behaviors for the same input.

Some models provide control parameters [74] that can influence their behavior. For example, the temperature is a parameter that affects randomness and creativity. A lower temperature reduces randomness and makes the model more focused and deterministic.

Top P or nucleus sampling, is another parameter which can impact randomness of LLM output. This parameter sets the threshold probability for including tokens in the candidate set used by the LLM to generate output. A lower value makes the responses more factual and precise, while a higher value increases randomness and diversity in the output.

However, even though these parameters can mitigate randomness, LLM can still exhibit variations in outputs due to the stochastic nature of their algorithms.

It is therefore recommended to address this specificity of LLM by specifying a guideline that requires conducting sensitivity analyses to identify and understand the degree of stochasticity inherent in the model being used.

5.5.2 LLM with learning process involved

In the case of LLM involving a learning process, two scenarios can be distinguished. Either it is a new model trained from a dataset (micro LLM or full LLM model), or it is an existing model fine-tuned with retraining of its parameters.

In the first case, the W-shape process developed in the context of machine learning fully applies. Since validation occurs during the model design phase, one has access to all training data as well as the model design. The major challenge lies in the evaluation of the model itself. In most ML applications, whether supervised or unsupervised, the model obtained from the training data is generally evaluated fairly easily using test data.

However, for LLM, this is not as straightforward. The generated outputs can be very varied and may include texts or computer code... These outputs are even not unique, and specific metrics are needed to evaluate them. This was illustrated in section 5.2, where a specific metric had to be applied to assess the outputs of the created models.

Therefore, it is recommended to define a guidance to discuss metric specifications in order to objectively evaluate performance and biases.

In the case where a pre-trained model is fine-tuned with training on specific datasets, the situation is somewhat different. The W process can be used again but must be adapted to account for the fact that an existing model is being modified. Transfer learning may also be involved. In all cases, it is crucial to ensure that the original model continues to perform well on the tasks for which it was designed while also performing well on the new tasks it has been assigned.

In this context, it is important to update the definition of the Operational Design Domain (ODD). The definition of test and validation sets should ensure the inclusion of data from both the old dataset and the new dataset. This also applies to version management, documentation and explainability.

5.5.3 What, where and how to evaluate LLM

A fundamental concept in model validation is the ability to effectively evaluate the model. In the context of large language models, this evaluation is far from straightforward. There is a multitude of available metrics, and often, the choice of the most appropriate metric depends on the specific objectives of the model being validated. The nature of the model's outputs also plays a crucial role in this choice. For example, textual responses and code generation are evaluated differently.

For models that generate textual responses, it is important to measure aspects such as accuracy, relevance, and fluency. Metrics like accuracy, recall, and F1 score, as well as automated measures such as BLEU and ROUGE, are commonly used to evaluate the quality of responses by comparing them to reference answers. However, human evaluations remain crucial for assessing coherence, contextual relevance, and overall quality of the responses.

On the other hand, for models generating code, the evaluation focuses on code correctness, efficiency, and readability. Unit tests and syntax validation can be employed to ensure that the code functions correctly and meets the specifications. Efficiency, in terms of performance and resource usage, is also a key dimension. Additionally, code readability and clarity are important for maintenance and understanding, though these aspects are often harder to quantify automatically.

In other words, evaluating Large Language Models involves a comprehensive assessment across various dimensions [75], [76]:

- Accuracy : focusing on the model's predictive accuracy requires utilizing tailored metrics to measure performance against observed data, aligning with specific tasks;
- Robustness : evaluating the model's ability to maintain accuracy under diverse scenarios and potential disruptions in the input data is a key element of the model's evaluation [44];
- Fairness : this requires the potential detection of biases across different population groups [77];
- Efficiency : efficiency evaluation involves measuring the model's effectiveness in utilizing data and computational resources to achieve its predictive capacity;
- Explainability : the model should establish transparent methodologies to illuminate the decision-making processes and ensure the model's outputs are interpretable and reliable.

Evaluating Large Language Models necessitates assessment in specialized evaluation datasets designed to test the models across various processing tasks. These evaluation datasets could be real or simulated scenarios. They should encompass an array of representative tasks considered for the studied software. The use of these datasets should provide a standardized and structured environment for testing LLM in order to be able to compare model performance across different tasks and scenarios [39].

Chapter 6

Conclusion and Future Directions

6.1 Conclusion

The aim of this Master's Thesis is to explore Large Language Models for analyzing aviation safety data. This study is part of the development by the European Union Aviation Safety Agency of collaborative tools for its Data4Safety platform under construction. The objective is to investigate how artificial intelligence, not yet developed on the platform, can be used to facilitate the analysis and study of safety data.

Initially, an analysis of the database was conducted, allowing for a thorough understanding of its structure and content. This included database dynamics analysis to understand data flows, analysis of the database structure to examine relationships and data profiling to evaluate data quality. The analysis highlighted several issues. The database contains numerous tables and attributes, but most records fill only a few of them, resulting in many "None" values that complicate usage. Certain attributes have the same name across different tables, posing a challenge for efficient processing with language models. Many fields are split into four or five attributes (e.g. responsible_entity_L1 to L4). As the data comes from a form, it should be aggregated into a single attribute. Interesting narrative attributes were also discovered, containing untapped information that language models could explore in depth.

Then, a study of the state of the art in LLMs provided a deeper understanding of this technology and techniques to exploit them optimally. For each method studied, practical use cases were created to illustrate and enhance understanding. From this, prompt engineering emerges as a powerful technique for refining prompts and providing templates or examples to clarify expected responses. Fine-tuning also yields interesting results by allowing a pre-trained model to specialize according to specific data. It has also been shown that numerous AI models exist and apply to very diverse fields. In the case of database analysis, Text2SQL and Retrieval Augmented Generation (RAG) techniques were particularly noteworthy.

Based on the acquired knowledge, different components, as shown on Table 6.1, have been implemented and assembled to create solution pipelines. Two different architectures have been proposed. The first solution involves a framework with local LLM models and a dedicated interface powered by Flask. This Implementation, on a local computer, is quite flexible and allows for easy results retrieval. Based on open-source solutions, it is more challenging to implement in industrial settings. The second solution involves implementing LLM models via notebooks on a `Databricks` server. Although constrained by the `Databricks` environment and limited to the LLM models available during the Master's Thesis, a modular implementation has been designed. The interface of such solution is quite limited as it relies on notebooks due to the enforcement of EASA's infosecurity policies. The details of the two implementation have been discussed. Additionally, two techniques have been studied. Text2SQL aims to generate SQL queries from natural language prompts while the RAG technique aims to retrieve the best possible outcomes from a provided set of data, offering fine tuned context to a pre-trained model.

AI Framework	**AI models**	**Interface**	**Techniques**
Databricks API	Mistral	Notebooks	Text2SQL
Ollama	DBRX	Flask frontend/backend	RAG
	Llama3		Hybrid
	duckdb-nsql		

Table 6.1: Overview of the components studied - *Created by author*

Once implemented, the different solutions were tested and compared. An evaluation set was created to evaluate the fundamental and reasoning capabilities of these solutions using around forty test prompts. The goal was not to assess the performance of the pre-trained models, as they have already been benchmarked, but to evaluate the performance of the full pipeline tailored to the database analysis. Execution accuracy was selected as the metric, and the different pipeline solutions were evaluated using the same verification set. Table 6.2 summarises the obtained results.

From the analysis conducted, several observations were made. Among the models available on the `Databricks` server, DBRX and Llama3 exhibit the best performance. The DBRX model is unique in its adherence to all provided instructions, while Llama3 excels at analyzing complex prompts but has more difficulties with complex database structure. The Mistral model, with its 7 billion parameters, struggles to deliver accurate results. The Duckdb model, tested only within the local framework, is specifically trained to generate queries. Despite having 7 billion parameters like Mistral, it provides remarkable good results.

Regarding the techniques implemented, Text2SQL appears to be the most appropriate technique for interacting with the database. RAG, while less effective at providing precise answers, excels at extracting information from narrative fields and supplying details not captured by database attributes. Therefore, an hybrid approach combining Text2SQL and RAG on the narrative attributes could provide the optimal solution.

Techniques	Framework	Models	Fundamental capabilities	Reasoning capabilities	Execution time	Security
Text2SQL	Databricks	Mistral	★★☆☆☆	★☆☆☆☆	4.3 s	API
Text2SQL	Databricks	LLama3	★★★★☆	★★★★☆	5.23 s	API
Text2SQL	Databricks	DBRX	★★★★☆	★★★☆☆	7.06 s	API
RAG	Databricks	Mistral	★☆☆☆☆	★½☆☆☆	3.6 s	API
RAG	Databricks	Llama3	★☆☆☆☆	★☆☆☆☆	20.6 s	API
RAG	Databricks	DBRX	★½☆☆☆	★★☆☆☆	3.2 s	API
Text2SQL	Ollama	Duckdb	★★★½☆	★★½☆☆	12.84 s	Local

Table 6.2: Summary of the obtained results - *Created by author*

About the local implementation framework, realised through `Ollama`, it offers the advantage of keeping all data within the development environment, ensuring complete data privacy. On the other hand, the `Databricks` framework used by EASA relies on API calls and LLM models deployed in the `Databricks` workspace which is the endpoint that the API is calling.

The created interface managed by Flask offers an interaction experience similar to that of ChatGPT or similar systems. On the `Databricks` server, notebooks allows for seamless integration of LLMs even this solution is more adapted to a development phase. Indeed, interactions require calling functions that necessitate understanding of the specific implementation. Additionally, the need to install required libraries each time the notebook is used, imposed by security reasons, impacts execution time and user experience.

A critical issue identified in this analysis is the lack of consistent performance of LLMs across various tasks, which may impair their use in safety-related applications. Although some models excel at specific queries, the lack of guarantee on other tasks necessitates human verification. This need for manual intervention, such as manually checking occurrences to ensure accuracy, poses a significant barrier to relying solely on model outputs for safety reporting.

Finally, validation techniques have been reviewed, highlighting that traditional design assurance methods are inadequate for learning processes. EASA has implemented a learning assurance process tailored for machine learning. If effectively developed for both supervised and unsupervised applications, this process should be adapted for large language models. For both new and fine-tuned models with and without learning involved, the validation process must incorporate specific testing to evaluate and understand model behavior. Discussing the repeatability of outcomes is also crucial, and performing sensitivity analyses is recommended to ensure robustness and reliability. Evaluation metrics depend on the specific application, making it essential to clearly define metrics that objectively evaluate model performance.

6.2 Future Directions

Looking ahead, this section outlines the next steps and opportunities for advancing the research. It explores potential improvements and identifies unexplored areas that could benefit from further investigation.

- **Guidelines and Terms of Use:**
  Comprehensive guidelines should be established to govern the deployment of LLMs. This includes implementing monitoring mechanisms, content filtering, spot checks, and investigations of user feedback to identify and address potential misuse.

  Example: Certain models may execute slowly if not used correctly. Guidelines can document such cases to prevent inefficiencies.

- **Library Versioning:**
  In the `Databricks` environment, most libraries are not pre-installed and due to security reasons, library installation is curated. Due to the rapid AI advancements, it is essential to specify library versions used for specific developments as newer versions may not offer the same functionalities.

- **AI models:**
  From the analysis conducted, results are sensitive to the models used. Newer models should be evaluated as they become available to explore potentially more accurate solutions.

  Example: A version of duckdB compatible with the Databricks server could be tested to compare the performance with the local implementation.

- **Model Misbehavior:**
  Regular model evaluations should be conducted to assess limitations and minimize instances of model misbehavior. This proactive approach can help in identifying potential biases, inaccuracies, or unintended outputs, allowing for timely corrections.

  Example: Depending on the selected AI model, outputs may be formatted differently. Some models perform poorly on specific tasks and should be selected carefully to maximize accuracy.

- **Document Weaknesses and Lessons Learned:**
  Known weaknesses and best practices in LLM usage have to be documented. This information should be made readily available to end users, developers, and stakeholders to enhance transparency. Additionally, lessons learned from instances of LLM safety concerns and misuses have to be disclosed.

- **User Education and Awareness:**
  End users should be informed about the capabilities, limitations, and potential risks associated with LLMs. Their awareness regarding responsible usage should be improved. Users should be empowered to make informed decisions while interacting with the technology.

- **Data quality:**
  Data quality should be ensured by implementing a consistent taxonomy. Data has to be standardised by using a uniform language and duplication of empty attributes has to be avoided. Additionally, the database has to be restructured to prevent the use of identical attribute names across different tables or multi lines attribute.

- **Quality of the prompt:**
  Prompt engineering and best practices to produce efficient outputs have to be implemented. Templates for structuring prompts and queries have to be developed and used to ensure consistency and effectiveness.

6.3 Final words

In concluding this Master's Thesis, it is important to recognise the potential of this technology. This Master's Thesis has delved into Large Language Models and their potential applications. Although these tools have proven to be highly powerful, they do have limitations and do not always provide yet the accurate outcome expected. Human verifications are still necessary, posing a significant barrier to relying solely on model outputs for safety reporting. However, LLMs can offer valuable assistance in analyzing safety databases, provided their outputs are critically evaluated before use. LLMs can also reveal information previously hidden in narrative fields.

Additionally, this study has presented an opportunity to address industrial constraints and adapt solutions using the available tools. While this research only initiates the implementation of such tools on the collaborative D4S platform, it underscores their immense development potential, which could greatly enhance the daily work of the data analysts.

Bibliography

[1] *EASA Website - European Union Aviation Safety Agency - The agency*. en. https://www.easa.europa.eu/en/the-agency/the-agency. Accessed: September 2023.

[2] *Official Journal of the European Union, Regulation (EU) No 376/2014 of the European Parliament and of the Council of 3 April 2014 on the reporting, analysis and follow-up of occurrences in civil aviation - EUR-Lex - 32014R0376*. en. Accessed: September 2023.

[3] *ECCAIRS Website - Occurence reporting database*. en. https://aviationreporting.eu/en/report-occurrence. Accessed: September 2023.

[4] *EASA Website - Data4Safety initiative*. en. https://www.easa.europa.eu/en/domains/safety-management/data4safety. Accessed: August 2023.

[5] *Artificial Intelligence Roadmap - Human-centric approach to AI in aviation, version 2.0, May 2023*. en. https://www.easa.europa.eu/en/domains/research-innovation/ai.

[6] David Hughes / Graph Practice Director. *Large Language Models (LLMs): A complete guide*. en. https://www.graphable.ai/blog/large-language-models-llms/. Accessed: 2023-10-15. Apr. 2023.

[7] Lucas Mearian. *What are LLMs, and how are they used in generative AI?* en. https://www.computerworld.com/article/3697649/. Accessed: 2023-10-15. May 2023.

[8] *ICAO SAFETY Website - Collection and Analysis of Safety Data*. en. \https://www.icao.int/safety. Accessed: October 2023.

[9] *EUROCONTROL Aviation Outlook 2050*. en. https://www.eurocontrol.int/publication/eurocontrol-aviation-outlook-2050. Accessed: 2023-11-7.

[10] Lucas Fernandes da Costa Pappacena et al. "State of the art of aviation safety reporting in Europe". en. In: *Aerotec. Missili Spaz.* 102.2 (2023), pp. 149–154.

[11] *Data4Safety: Putting the data jigsaw together*. en. https://www.eurocockpit.be/news/data4safety-putting-data-jigsaw-together. Accessed: 2023-10-15.

[12] *European co-ordination centre for accident and incident reporting systems (ECCAIRS)*. en.
https://skybrary.aero/articles/european-co-ordination-centre-accident-and-incident-reporting-systems-eccairs.
Accessed: October 2023.

[13] Directorate General for Mobility European Commission and Transport. *Ex-Post Evaluation of Regulation (EU) No 376/2014 on the reporting, analysis and follow-up of occurrences in civil aviation - Final report*. Accessed: December 2023. 2020.

[14] Stefan Wagner. *Software Product Quality Control*. 2013.

[15] Stuart Russell and Peter Nrovig. "Introduction - What is AI". In: *Artificial Intelligence : A modern approach*. Pearson Eduction Inc., 2021, pp. 21–35.

[16] Tianyu Wu et al. "A brief overview of ChatGPT: The history, status quo and potential future development". en. In: *IEEE/CAA J. Autom. Sin.* 10.5 (2023), pp. 1122–1136.

[17] Yun-Cheng Wang et al. "An overview on generative AI at scale with edge-cloud computing". In: (2023). arXiv [cs.DC]. Available at: http://arxiv.org/abs/2306.17170. eprint: 2306.17170.

[18] *Georges Artstrouni: The mechanical Translator, August 2017*. en.
https://cafetran.freshdesk.com/support/discussions/topics/6000050879.

[19] Wayne Xin Zhao et al. "A survey of large language models". In: (2023). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2303.18223. eprint: 2303.18223.

[20] Jason Wei et al. "Emergent abilities of large language models". In: (2022). eprint: 2206.07682.

[21] Roberto Gozalo-Brizuela and Eduardo C Garrido-Merchan. "ChatGPT is not all you need. A State of the Art Review of large Generative AI models". In: (2023). arXiv [cs.LG]. Available at: http://arxiv.org/abs/2301.04655. eprint: 2301.04655.

[22] Michael I Jordan. "Serial order: A parallel distributed processing approach". In: *Neural-Network Models of Cognition - Biobehavioral Foundations*. Elsevier, 1997, pp. 471–495.

[23] J Elman. "Finding structure in time". In: *Cogn. Sci.* 14.2 (1990), pp. 179–211.

[24] Ilya Sutskever, James Martens, and Geoffrey Hinton. *Generating text with recurrent neural networks*. http://www.icml-2011.org/papers/524_icmlpaper.pdf.

[25] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. "Learning to generate reviews and discovering sentiment". In: (2017). eprint: 1704.01444.

[26] Ashish Vaswani et al. "Attention is all you need". In: (2017). arXiv [cs.CL]. Available at: http://arxiv.org/abs/1706.03762. eprint: 1706.03762.

[27] Sang Michael Xie et al. *An explanation of in-context learning as implicit Bayesian inference*. http://arxiv.org/abs/2111.02080. Accessed: 2023-11-1.

[28] Xinyi Wang et al. *Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning*. http://arxiv.org/abs/2301.11916. Accessed: 2023-11-1.

[29] Daniel A Roberts, Sho Yaida, and Boris Hanin. "The principles of deep learning theory". In: (2021). eprint: 2106.10165.

[30] *Word embeddings in NLP: A Complete Guide*. en. https://www.turing.com/kb/guide-on-word-embeddings-in-nlp. Accessed: September 2023. Feb. 2022.

[31] Jon Ezeiza Alvarez and Hannah Bast. *A review of word embedding and document similarity algorithms applied to academic text*. https://ad-publications.cs.uni-freiburg.de/theses/Bachelor_Jon_Ezeiza_2017.pdf. Accessed: 2023-09-20. 2017.

[32] Alammar J. "The illustrated Transformer". en. In: *IEEE/IJCNN* (2028). Retrieved from https://jalammar.github.io/illustrated-transformer/.

[33] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. "The power of generative AI: A review of requirements, models, input–output formats, evaluation metrics, and challenges". en. In: *Future Internet* 15.8 (2023), p. 260.

[34] Nikolas Adaloglou. "Transformers in Computer Vision". In: *https://theaisummer.com/* (2021).

[35] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for text classification". In: (2018).
arXiv [cs.CL]. Available at: http://arxiv.org/abs/1801.06146. eprint: 1801.06146.

[36] Tom B Brown et al. "Language Models are Few-Shot Learners". In: (2020). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2005.14165. eprint: 2005.14165.

[37] Immanuel Trummer. "From BERT to GPT-3 Codex: Harnessing the potential of very large language models for data management". In: (2023). arXiv [cs.DB]. Available at: http://arxiv.org/abs/2306.09339. eprint: 2306.09339.

[38] Mike Huisman, Jan N van Rijn, and Aske Plaat. "A survey of deep meta-learning". en. In: *Artif. Intell. Rev.* 54.6 (2021), pp. 4483–4541.

[39] Timm Teubner et al. "Welcome to the Era of ChatGPT et al: The Prospects of Large Language Models". en. In: *Bus. Inf. Syst. Eng.* 65.2 (2023), pp. 95–101.

[40] *Prompt engineering for Developers*. en. https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/. Accessed: September 2023. May 2023.

[41] Jean Kaddour et al. "Challenges and applications of large Language Models". In: (2023). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2307.10169. eprint: 2307.10169.

[42] Michael O'Neill and Mark Connor. "Amplifying limitations, harms and risks of large language models". In: (2023).
arXiv [cs.CL]. Available at: http://arxiv.org/abs/2307.04821. eprint: 2307.04821.

[43] Yotam Wolf et al. "Fundamental limitations of alignment in large language models". In: (2023). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2304.11082. eprint: 2304.11082.

[44] Yupeng Chang et al. "A survey on evaluation of large language models". In: (2023). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2307.03109. eprint: 2307.03109.

[45] Chaoning Zhang et al. "A complete survey on generative AI (AIGC): Is ChatGPT from GPT-4 to GPT-5 all you need?" In: (2023). arXiv [cs.AI]. Available at: http://arxiv.org/abs/2303.11717. eprint: 2303.11717.

[46] Shengyu Zhang et al. "Instruction tuning for large language models: A survey". In: (2023). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2308.10792. eprint: 2308.10792.

[47] Jinyang Li et al. "Can LLM already serve as A database interface? A BIg bench for large-scale database grounded text-to-SQLs". In: (2023). arXiv [cs.CL]. Available at: http://arxiv.org/abs/2305.03111. eprint: 2305.03111.

[48] Chenxu Hu et al. "ChatDB: Augmenting LLMs with databases as their symbolic memory". In: (2023). Available at: http://arxiv.org/abs/2306.03901. eprint: 2306.03901.

[49] I Androutsopoulos, G D Ritchie, and P Thanisch. "Natural language interfaces to databases - an introduction". In: (1995).
arXiv [cs.CL]. Available at: http://arxiv.org/abs/cmp-lg/9503016. eprint: cmp-lg/9503016.

[50] Fei Li and Hosagrahar V Jagadish. "NaLIR: An interactive natural language interface for querying relational databases". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2014.

[51] Allen H Huang, Hui Wang, and Yi Yang. "FinBERT: A large language model for extracting information from financial text". en. In: *Contemp. Acc. Res.* 40.2 (2023), pp. 806–841.

[52] Yiru Chen and Eugene Wu. "PI2: End-to-end interactive visualization interface generation from queries". In: (2021).
arXiv [cs.DB]. Available at: http://arxiv.org/abs/2107.08203. eprint: 2107.08203.

[53] Mc Kinsey Company. *The state of AI in 2023: Generative AI's breakout year*.
https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2023-generative-AIs-breakout-year.
Accessed: April 2024.

[54] OpenAI. *Overview of the Open AI platform and capabilities*.
https://platform.openai.com/docs/overview.
Accessed: April 2024.

[55] Datacamp. *Run LLMs Locally: 7 simple Methods*.
https://www.datacamp.com/tutorial/run-llms-locally-tutorial.
Accessed: January 2024.

[56] Github blog. *The architecture of today's LLM applications*.
https://github.blog/2023-10-30-the-architecture-of-todays-llm-applications/.
Accessed: March 2024.

[57] jan.ai. *Overview of Jan.ai*.
https://jan.ai/docs.
Accessed: February 2024.

[58] ollama. *Overview of the Ollama.ai*.
`https://ollama.com/`.
Accessed: February 2024.

[59] databricks. *Databricks for industry documentation*.
`https://www.databricks.com/solutionsby-industry`.
Accessed: April 2024.

[60] Flask Pallets projects. *Flask's documentation*.
`https://flask.palletsprojects.com/en/3.0.x/`.
Accessed: January 2024.

[61] Jinyang Li et al. "Can LLM already serve as A database interface? A BIg bench for large-scale database grounded text-to-SQLs". In: (2023). eprint: `2305.03111`.

[62] Langchain. *Langchain's documentation*.
`https://www.langchain.com/`.
Accessed: January 2024.

[63] SQLAlchemy. *Schema definition language*.
`https://docs.sqlalchemy.org/en/20/core/schema.html`.
Accessed: February 2024.

[64] Yunfan Gao et al. "Retrieval-Augmented Generation for large Language Models: A survey". In: (2023). eprint: `2312.10997`.

[65] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive NLP tasks". In: (2020). eprint: `2005.11401`.

[66] Chroma. *Chroma's documentation*.
`https://docs.trychroma.com/`.
Accessed: January 2024.

[67] Tinghui Ouyang et al. "Autonomous driving quality assurance with data uncertainty analysis". en. In: *IEEE/IJCNN* (2022).

[68] Tinghui Ouyang et al. "Quality assurance of A GPT-based sentiment analysis system: Adversarial review data generation and detection". In: (2023). arXiv [cs.SE]. Available at: http://arxiv.org/abs/2310.05312. eprint: `2310.05312`.

[69] Duane Kritzinger. *Aircraft system safety: Assessments for Initial Airworthiness Certification*. en. Woodhead Publishing, 2016.

[70] EASA AI Task Force. *EASA Concept Paper: first usable guidance for Level 1 machine learning applications*. EASA Internal. Accessed: November 2023.

[71] Xinyi Hou et al. "Large Language Models for Software Engineering: A systematic literature review". In: (2023). arXiv [cs.SE]. Available at: http://arxiv.org/abs/2308.10620. eprint: `2308.10620`.

[72] D Wallace and L M Ippolito. "A framework for the development and assurance of high integrity software". In: *D Wallace, L M. Ippolito* (1994).
Accessed: October 2023.

[73] EASA AI Task Force and Daedalean AG. *Concepts of Design Assurance for Neural Networks (CoDANN) II*. EASA Internal. Accessed: November 2023.

[74] Medium. *LLM settings: temperature, Top P and max tokens.*
`https://medium.com/@albert_88839/large-language-model-settings-temperature-top-p-and-max-tokens-1a0b54dcb25e`.
Accessed: June 2024.

[75] Rossella Locatelli, Giovanni Pepe, and Fabio Salis. "The Validation of AI Techniques". In: *Artificial Intelligence and Credit Risk*. Cham: Springer International Publishing, 2022, pp. 65–79.

[76] Michael C Frank. "Baby steps in evaluating the capacities of large language models". en. In: *Nat. Rev. Psychol.* 2.8 (2023), pp. 451–452.

[77] Roberto Navigli, Simone Conia, and Björn Ross. "Biases in large language models: Origins, inventory, and discussion". en. In: *ACM J. Data Inf. Qual.* 15.2 (2023), pp. 1–21.

Appendix A

Python and R codes from the analysis of data

Various Python and R codes have been created by the author to illustrate methods, analyse data or implement models. These are all uploaded on a dedicated github: Master's Thesis github

- Appendix A1: Python code to extract the list of tables and attributes
- Appendix A2: Python code to extract the data from the database
- Appendix A3: Rcode to analyse the extracted data
- Appendix A4: Python code to draw the distribution of reported occurrences on the European Map

Appendix B

Python and R codes from the state of art

Various Python and R codes have been created by the author to illustrate methods, analyse data or implement models. These are all uploaded on a dedicated github: Master's Thesis github

- Appendix B1: Joint probability script
- Appendix B2: Neural Network
- Appendix B3: Word embedding script
- Appendix B4: Vector space visualisation using t-SNE
- Appendix B5: Transformers and attention mechanism
- Appendix B6: Fine tuning script

Appendix C

Can I use ChatGPT at work ?

A 2 pagers has been prepared and addressed to EASA employees to raise awareness about the use of generative AI. The publication is presented on the next 2 pages.

EASA
European Union Aviation Safety Agency

CAN I USE CHAT GPT AT WORK?

Understanding generative AI chatbots for a responsible use

Sponsored by the EASA AI Programme

Limitations and risks

Publicly available generative AI tools come with risks. Trained on massive datasets, these models are susceptible to a range of issues:

Inaccuracy and unreliability

Generated outputs may exhibit bias, ambiguities, factually inaccurate information or be oriented by an unknown set of predefined rules.

Lack of traceability

Many models generate outputs without proper sourcing which obscures the origins and reliability of the information presented as outputs.

Limited training-data quality

Since the models are trained on data up to a certain date, they may not reflect the most current information available.

Data protection vulnerabilities

Significant data privacy and confidentiality issues arise due to the need to transmit and store user inputs on the providers' servers and potentially third-party servers. This creates vulnerabilities where personal and sensitive data could be intercepted or mishandled.

Copyright issues

Generated outputs may contain copyrighted material, posing legal concerns, or may be under the AI provider's ownership.

When can I use public generative AI? Three Golden Rules:

Always safeguard sensitive and personal information

The data used in your prompt, as well as any uploaded inputs, can be used by the large language model (LLM) provider for performance improvements or other uses. Never share with a generative AI tool any personal data (for example, photos) although you might have found these on the public domain on the internet.

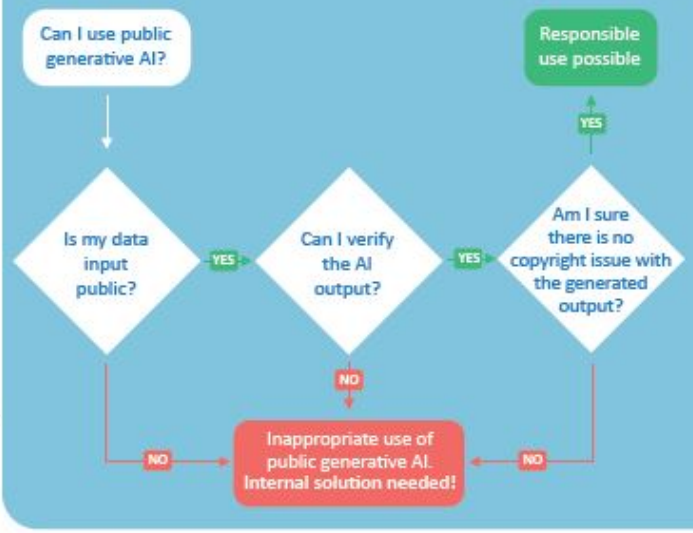
Incorporate human expert oversight of the outputs

Always critically evaluate generated content and cross-reference it with other sources to ensure the quality and accuracy of the outputs. Never directly replicate the outputs from an AI tool in official EASA documents or rely on such outputs for time-sensitive or safety-critical tasks.

Do not infringe copyright

Always critically assess whether the outputs of an online available generative AI model are not violating intellectual property rights, in particular copyright of third parties.

Feasibility of internal solutions is under investigation by the AI Programme Team.
In the meantime, for public generative AI:



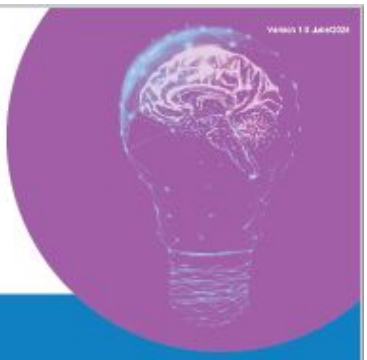
In case of doubt or questions, contact the AI Programme Team (ai@easa.europa.eu)

EASA
European Union Aviation Safety Agency

CAN I USE CHAT GPT AT WORK?

Understanding generative AI chatbots for a responsible use

Sponsored by the EASA AI Programme



A bit of terminology

Artificial Intelligence (AI)	Technology that can, for a given set of human-defined objectives, generate outputs such as content, predictions, recommendations, or decisions that influence the environments they interact with. This covers various methods and algorithms.
Generative AI	AI systems capable of creating content such as text, images, audio, videos and computer codes, or a combination of all of these.
Public Generative AI	Generative AI tool available online (e.g. ChatGPT, Copilot, Gemini, Claude, etc.).
Large Language Models (LLMs)	AI models that learn skills to generate prose, engage in conversations, write computer code by analysing vast amounts of text and data.
Natural Language Processing	A field of AI, used by LLMs, that enables computers to understand and process human language.
ChatGPT, Gemini, Copilot, etc.	Chatbots using generative AI, LLMs, and natural language processing to simulate human-like conversations. More than a traditional bot, it uses its understanding of the language to interpret the question and determine the most appropriate response.

What can Generative AI do today?

Generative AI has a wide range of applications. These kinds of tools can already create most types of written, image, audio, video, and coded content. In the near future, applications that target specific functions should proliferate and offer a wider and more accurate experience. For now, some of the applications are (non-exhaustive list):

Text
Content writing (e.g. emails, minutes, draft reports), assistance (Q&A), brainstorming, web search, analysis and synthesis, translations, proofreading.

Code
Code generation, error detection, code documentation, data set generation.

Image
Image generator, image editor, creation.

Audio
Text to voice, sound creation, audio editing.

Video
Video creation, editing, translation, virtual effects.

References:

- This flyer derives from the work of Jade LEJEUNE HERMAN, developed in the context of her Master Thesis in EASA
- EU Guidelines for staff on the use of online available Generative Artificial Intelligence tools available at: https://www.easa.europa.eu/intranet/all-stream/intranet-stream/staff-use-generative-ai-tools

Appendix D

More on Text2SQL

Several techniques exist in the literature [61] and can be classified into 5 groups:

- Rule-base approach: this approach relies on predefined patterns to convert the natural language prompt into SQL queries.
  Example :

```
query = "Find the aircraft type with the highest
    accident rate."
if "highest rate" in query:
    sql_query = "SELECT type FROM aircraft ORDER BY
    Frequency DESC LIMIT 1;"
```

  It's powerful for limited size databases with predictable query types. However, in complex scenarios, it can only be applied on a subset of the data, making it less suitable.

- Template-base approach: This approach uses predefined templates for common query types. These templates act as a skeleton, and the system fills in details based on the parsed input. Templates can be very complex and enable accurate information extraction. However, they require pre-defined anticipated queries. They can also be used to map the database structure itself, facilitating query generation by identifying links between tables.
  Example:

```
templates = {
    "find X of Y with Z": "SELECT {X} FROM {Y}
    ORDER BY {Z} DESC LIMIT 1;"
}

query = "Find the aircraft type with the highest
    accident rate."

placeholders = {"X": "Aircraft type", "Y": "aircraft",
"Z": "frequency"}

template = templates["find X of Y with Z"]
sql_query = template.format(**placeholders)
```

- Semantic parsing approach: this approach involves mapping the syntactic structure of the query to a meaning representation. This includes identifying the intent of the query and the roles of various entities. A logical form is used to capture these semantics and can be directly used to map the SQL. This approach can handle

a wide variety of queries by understanding their semantics but may struggle with complex ones.

- Machine Learning approach: Machine learning approaches use deep learning models to generate SQL. These models are trained on large datasets of natural language queries and their corresponding SQL queries. This approach offers flexibility but requires significant training data and computational resources.

- Hybrid approach: this approach combines the strengths of the methods presented above. For example, a system might use a machine learning approach combined with a database template. This combination can create a highly accurate system as the structure guides the algorithm to provide accurate queries.

Appendix E

Codes from the implementation and the results

Several codes have been created by the author to implement the solutions outlined in Chapter 4 and to generate the results discussed in Chapter 5. These are all uploaded on a dedicated github:
Master's Thesis github

- Appendix E1: Local implementation with Ollama and Chroma.py
- Appendix E2: Evaluation Text2SQL - DBRX.ipynb
- Appendix E3: Evaluation Text2SQL - Llama3.ipynb
- Appendix E4: Evaluation Text2SQL - Mistral.ipynb
- Appendix E5: Evaluation RAG - DBRX.ipynb
- Appendix E6: Evaluation RAG - Llama3.ipynb
- Appendix E7: Evaluation RAG - Mistral.ipynb
- Appendix E8: Evaluation RAG - Narrative.ipynb
- Appendix E9: Evaluation Text2SQL - Ollama Duckdb-nsql.ipynb

The results used in Chapter 5 to evaluate the different models are available in the github repository.

- Appendix E10: Evaluation set: question.pdf
- Appendix E11: Evaluation results: Results.pdf
- Appendix E12: Evaluation Matrix and plots.xls

Additional codes have been developed to explore and study initial solutions. While not referenced in this study, they are available in the github repository.

- Appendix E13: Test SQL on databricks.ipynb
- Appendix E14: Test LLMs on databricks.ipynb
- Appendix E15: Test ChromadB on databricks.ipynb
- Appendix E16: Test Text2SQL on databricks.ipynb
- Appendix E17: Test Chroma on ollama.ipynb
- Appendix E18: LLM with interface Flask.ipynb

Appendix F

EASA Publication

During the kick-off meeting held at EASA headquarters, a dedicated communication was made to introduce the Master's Thesis. The publication is presented in the next two pages.

February 9, 2024

ThisWeek@EASA

Internal use of Artificial Intelligence by EASA: Launch of a Master's Thesis to explore "ChatGPT-like" applications for the benefit of EASA

Most of us have probably already made the stunning experience of using tools like "ChatGPT". Maybe you are already having human-like conversations on a daily basis with some Chatbots powered by Generative Artificial Intelligence (GenAI)? 😉

With the meteoric emergence of natural language processing tools driven by AI technology, there is a fundamental (and burning!) question for the Agency and its staff: What's in it for EASA? Can GenAI provide opportunities to perform some of our tasks more rapidly and more efficiently? Can it even change the way we approach some of our core tasks?

On the other side of the coin, can these tools be used without compromising the integrity of our sensitive data and information? If yes, what are the critical skills sets to be developed internally and - as importantly - what are the key principles that each of us should apply when using such AI-powered tools to deliver on our work?

This week, EASA had the opportunity of welcoming Mrs Jade LEJEUNE HERMAN, a MSc Candidate working on her Master's Thesis titled 'Exploring Large Language Models for analysing aviation safety data' at the KU Leuven University. Supported by EASA, Mrs LEJEUNE HERMAN's research is at the crossroad of some of our flagship EASA's initiatives in that field.

Data4Safety
Partnership for Data Driven Aviation Safety Analysis
Supported by EASA

February 9, 2024

ThisWeek@EASA

It involves studying and proposing an implementation of Large Language Models (LLM) to facilitate the analysis of the information contained in the safety reporting data captured thanks to the ECCAIRS 2 software suite (E2) managed by EASA and used by our Member States. Her research work is made possible thanks to the AI Platform being deployed as part of the Big Data Platform infrastructure of the Data4Safety Programme (D4S). Her work aims at exploring and demonstrating if and how LLM can be used to enhance our capabilities to utilize the safety reporting data at our disposal to manage safety risks (i.e. our EASA Safety Risk Management and Intelligence core processes like the SRM).

As an important guide for her work, her Master's Thesis will apply some of the critical guidance developed by EASA in the context of the Agency's AI Programme. This will ultimately contribute to the reflection of EASA and work in cooperation with our sector about how to ensure the right level of trustworthiness for AI solutions in the demanding Aviation context. For the EASA staff, this research work will contribute to the development of internal experience, know-how and best practices to help us make the most of these new solutions "safely"!

The kick-off meeting of Ms Lejeune Herman's Masters's Thesis was hosted this week in EASA headquarters, with the participation of Leopold Viroles (Data4Safety Programme Manager and EASA Supervisor of the Master's Thesis) and Guillaume Soudain (AI Programme Manager).

We wish her much success and look forward to discover the possibilities of such models for EASA!

One remarkable aspect of Ms Lejeune Herman is her exceptional achievement despite her young age: at only 15 years of age, she has already obtained a BSc in engineering. Currently, she is pursuing her Master in Statistics and Data Science at KU Leuven University in Belgium.

Data4Safety

AFDELING
Straat nr bus 0000
3000 LEUVEN, BELGIË
tel. + 32 16 00 00 00
fax + 32 16 00 00 00
www.kuleuven.be

Exploring Large Language Models for Analyzing Aviation Safety Data

In its February 9 edition, Thisweek@EASA introduced Jade Lejeune Herman (16) who is finishing an MSc in Data Science and Statistics at KU Leuven. She had begun her Master's thesis in collaboration with EASA in the field of artificial intelligence, with the objective to explore AI Models, similar to ChatGPT, to analyze aviation safety data stored on the Data4Safety platform.

This week marks a significant milestone for Jade Lejeune Herman (and EASA!) as she completed and defended her Master's Thesis with support from EASA. Since the kick-off meeting, Jade has studied and implemented various solutions for interacting with ECCAIRS2, EASA's safety reports database. As the development of the Data4Safety platform progressed, she adapted these solutions to interact directly with the database on this platform. This provided an opportunity for Jade and EASA to test initial AI implementations on the developing platform. Additionally, Jade analysed and compared different LLM models, evaluating their strengths and weaknesses and offering recommendations for optimal use. She also assessed various AI techniques, conducted an analysis of validation methods, and explored their adaptation for LLM applications.

Guided by Leopold Viroles (Data4Safety Programme Manager) and Guillaume Soudain (AI Programme Manager), Jade submitted her thesis in August. The report is considered highly valuable to both the community and the Agency. Today, Jade successfully defended her Master's Thesis with Leopold Viroles and Guillaume Soudain in attendance.