



**Politecnico  
di Torino**

POLITECNICO DI TORINO

Master Degree course in Digital Skills for Sustainable Societal Transitions

Master Degree Thesis

**Empowering Enterprises with  
Lightweight Large Language Models:  
Automated *Rule Card* Extraction from  
Grant Documents**

**+ CIM4.0**

**Supervisors**

Prof. Gianvito URGESE

Vittorio FRA

**Company Tutors**

Flavio CERATO

Edoardo TODDE

**Candidate**

Hesamedin ALEMIFAR

ACADEMIC YEAR 2024-2025

# Acknowledgements

*"I would like to express my sincere gratitude to my academic supervisors, Prof. Gianvito Urgese and Vittorio Fra, for their invaluable guidance, insights, and continuous support throughout this research.*

*I am also deeply grateful to my company tutors, Flavio Cerato and Edoardo Todde, for their mentorship, technical expertise, and practical insights, which greatly enriched my understanding of the subject. Their support has been crucial in bridging academic research with real-world applications.*

*Finally, I express my heartfelt gratitude to my parents and my sister for their unconditional love, unwavering support, and constant encouragement throughout my academic journey. This achievement would not have been possible without them"*

## Abstract

In the face of rapidly expanding unstructured data, organizations, especially small and medium-sized enterprises (SMEs), require automated solutions that not only offer accurate information extraction but also preserve data privacy. This thesis addresses such needs by introducing a lightweight, open-source Large Language Model (LLM) pipeline designed to extract structured "Rule Cards" from Italian grant and funding documents. By running locally with models like Llama 3.1, the system mitigates potential privacy risks associated with sharing data on external servers.

The proposed pipeline employs a modular approach encompassing PDF parsing, Optical Character Recognition (OCR), chunk-based text segmentation, and domain-specific prompt engineering. Compared to frontier models (e.g., GPT-4, Gemini Pro), these smaller open-source models demonstrated competitive performance, as measured by BERTScore, while retaining the advantages of reduced computational overhead and on-premise deployment. On the other hand, some limitations exist: occasionally, missing or incomplete information arose from overly long or imprecise instructions, and hallucination occurred when the model attempted to generate details that were absent from the source document. Despite these issues, the focused prompts and verification steps minimized the impact of errors, underscoring the pipeline's adaptability and potential in real-world settings.

By highlighting the viability of lightweight LLMs for specialized tasks, this thesis opens avenues for future research, such as fine-tuning multimodal models to enhance OCR for Italian texts and expanding the pipeline to handle additional data types. Ultimately, the findings demonstrate that domain-tuned, open-source LLMs can effectively extract structured information while maintaining privacy, offering a practical and scalable solution for SMEs and other organizations.

**Keywords:** Large Language Models (LLMs), Information Extraction, Open-Source Models, Italian Grant Documents, Domain-Specific Prompt Engineering

# Contents

<b>1</b>	<b>Introduction</b>	9
1.1	Motivation	9
1.2	Problem Statement	9
1.3	Importance of the research	10
1.4	Research objectives	10
1.5	Thesis structure	10
<b>2</b>	<b>Background and related works</b>	13
2.1	Introduction to Artificial Intelligence	13
2.1.1	Artificial Intelligence (AI)	13
2.1.2	Machine Learning (ML)	14
2.1.3	Deep Learning (DL)	14
2.1.4	Natural Language Processing (NLP)	15
2.1.5	Generative AI	15
2.2	Introduction to LLMs	16
2.2.1	Architecture of transformer model	16
2.2.2	Taxonomy of LLM tasks	17
2.2.3	Domain-specific applications of LLMs	18
2.3	Foundation model selection criteria	19
2.4	Strategies and techniques for customizing LLMs	23
2.4.1	Fine-tuning	23
2.4.2	Retrieval-Augmented Generation (RAG)	25
2.4.3	Prompt engineering	28
2.5	Tools for creating LLM applications	30
2.5.1	LLM-app platforms	30
2.5.2	Vector databases	31
2.5.3	LLM orchestration frameworks	32
2.6	Related work	33
2.6.1	Large Language Models for Generative Information Extraction: A Survey	33
2.6.2	Exploring Open Information Extraction for Portuguese Using Large Language Models	33
2.6.3	Exploring the Potential of Lightweight LLMs for Medication and Timeline Extraction	34

<b>3</b>	<b>Methodology</b>	35
3.1	Case study . . . . .	35
3.2	Choice of LLM . . . . .	36
3.3	Model customization . . . . .	38
3.4	Pipeline definition . . . . .	39
<b>4</b>	<b>Results</b>	43
4.1	Benchmark . . . . .	43
4.2	Case study outcome . . . . .	43
4.3	Prompts . . . . .	44
4.3.1	Zero-shot learning . . . . .	44
4.3.2	Few-shot learning . . . . .	46
4.3.3	Instruction-based learning . . . . .	46
4.4	Evaluation approach . . . . .	50
4.4.1	Performance metrics . . . . .	50
4.4.2	Evaluation . . . . .	52
4.5	Error analysis . . . . .	54
4.5.1	Metrics visualization . . . . .	55
<b>5</b>	<b>Discussion</b>	69
5.1	Analysis of results . . . . .	69
5.2	Challenges encountered . . . . .	71
5.3	Implications for future work . . . . .	72
5.4	Meet the Research Objectives . . . . .	73
<b>6</b>	<b>Conclusion</b>	75
	<b>Bibliography</b>	77
<b>A</b>	<b>Project Repository</b>	81

# List of Figures

2.1	Relationship between AI, ML, DL, NLP, and LLM. . . . .	15
2.2	Architecture of transformer models [36]. . . . .	17
3.1	The pipeline . . . . .	40
3.2	The user interface of the application . . . . .	41
4.1	The PDF file example . . . . .	44
4.2	The extracted "Rule Card" . . . . .	45
4.3	Zero-shot prompt for JSON extraction . . . . .	45
4.4	Few-shot prompt for OCR improvement . . . . .	46
4.5	Few-shot prompt for filtering texts . . . . .	47
4.6	Instruction-based prompt for JSON extraction . . . . .	48
4.7	Instruction-based prompt for verification of the extracted information . . . . .	49
4.8	Types of metric scorers [25] . . . . .	51
4.9	Document 1 metrics. . . . .	55
4.10	Document-1 "titolo" ground truth . . . . .	56
4.11	Document-1 "titolo" extracted data . . . . .	56
4.12	Document 2 metrics. . . . .	56
4.13	"Interventi-ammissibili" ground truth of document 2 . . . . .	57
4.14	"Interventi-ammissibili" extracted data of document 2 . . . . .	57
4.15	Document 3 metrics. . . . .	58
4.16	"Risorse-finanziarie" ground truth of document 3 . . . . .	58
4.17	"Risorse-finanziarie" extracted data of document 3 . . . . .	58
4.18	Document 4 metrics. . . . .	59
4.19	"Scadenza" ground truth of document 4 . . . . .	59
4.20	"Scadenza" extracted data of document 4 . . . . .	59
4.21	Document 5 metrics. . . . .	60
4.22	"Risorse-finanziarie" ground truth of document 5 . . . . .	60
4.23	"Risorse-finanziarie" extracted data of document 5 . . . . .	61
4.24	Document 6 metrics. . . . .	61
4.25	"Candidati-idonei" ground truth of document 6 . . . . .	62
4.26	"Candidati-idonei" extracted data of document 6 . . . . .	62
4.27	"Risorse-finanziarie" ground truth of document 6 . . . . .	62
4.28	"Risorse-finanziarie" extracted data of document 6 . . . . .	62

4.29	Document 7 metrics. . . . .	63
4.30	"Titolo" ground truth of document 7 . . . . .	63
4.31	"Titolo" extracted data of document 7 . . . . .	64
4.32	Document 8 metrics. . . . .	64
4.33	"Scadenza" ground truth of document 8 . . . . .	65
4.34	"Scadenza" extracted data of document 8 . . . . .	65
4.35	Document 9 metrics. . . . .	65
4.36	"Titolo" ground truth of document 9 . . . . .	66
4.37	"Titolo" extracted data of document 9 . . . . .	66
4.38	Document 10 metrics. . . . .	67
4.39	"Titolo" ground truth of document 10 . . . . .	67
4.40	"Titolo" extracted data of document 10 . . . . .	68
4.41	"Candidati-idonei" ground truth of document 10 . . . . .	68
4.42	"Candidati-idonei" extracted data of document 10 . . . . .	68

# Acronyms

**AI** Artificial Intelligence

**ML** Machine Learning

**DL** Deep Learning

**NLP** Natural Language Processing

**LLM** Large Language Model

**VLM** Vision Language Model

**MLLM** Multi-modal LLM

**RAG** Retrieval-Augmented Generation

**CRAG** Corrective RAG

**ICL** In-Context Learning

**CoT** Chain-of-Thoughts

**ToT** Tree-of-Thoughts

**IE** Information Extraction



# Glossary

**API** Application Programming Interface is a software interface that enables communication and interaction between different programs.

**Context window** The model's capacity to process text at once is usually determined by the number of tokens it can handle.

**Hallucination** Hallucination occurs when Large Language Models (LLMs) generate responses that are misleading, factually incorrect, or entirely fabricated.

**Inference** Inference refers to the process of generating text using a generative LLM.

**JSON** JavaScript Object Notation is an open-standard format used for file and data exchange. It employs human-readable text to represent data objects, which consist of attribute-value pairs and arrays.

**LLaMa** Large Language Model Meta AI refers to a series of large language models developed and released by Meta AI.

**Temperature** Temperature determines the level of randomness in the response.

**Tokenization** Tokenization is the process of breaking text into smaller units, such as words, subwords, or characters, that the model can understand and process.



# Chapter 1

## Introduction

This thesis explores how Artificial Intelligence (AI) and Large Language Models (LLMs) can be used to extract information from unstructured data, with a focus on small and medium enterprise (SMEs) documents. This is a critical challenge due to the rapid increase in data across various formats, such as PDF files, Microsoft PowerPoint files, etc. The research is conducted in collaboration with and supported by the CIM4.0 Competence Center.

The aim of this chapter is to provide an overview of the subject, problem statement, and the overall purpose of this study.

### 1.1 Motivation

ML and Neural Networks have roots in the 1950s. Over time, machine learning, particularly DL, has made significant advancements, often surpassing human performance in fields like NLP, computer vision, medicine, robotics, and more [10].

In today's fast-changing digital world, the demand for advanced AI systems is growing rapidly. AI is not just about handling routine tasks but also about improving efficiency and speeding up processes. As businesses and industries adopt AI to stay competitive, there is a rising need for tools that can analyze and extract useful insights from large amounts of unstructured data. These tools are transforming how companies work, make decisions, interact with customers, and automate tasks across different industries [35].

### 1.2 Problem Statement

The digital age is marked by an explosion of data, much of which exists as unstructured text from diverse sources. Extracting meaningful insights from this massive volume of data is a crucial task [11].

On the one hand, organizations rely heavily on data from diverse sources such as invoices, surveys, and legal documents to support their operations. Extracting relevant information from this data, a process known as Information Extraction (IE), is crucial for making informed decisions [5].

On the other hand, LLMs can be vulnerable to passive privacy leaks. Users may unintentionally share sensitive information when inputting data into interfaces like ChatGPT. For instance, Samsung Electronics faced incidents where confidential company data was inadvertently exposed through ChatGPT on three separate occasions [48]. Such risks have led many enterprises to avoid using proprietary LLMs and adopt a more cautious approach, hesitating to share their data with these vendors.

This thesis proposes to explore the customization of LLMs for enterprises to develop a specialized AI-application tailored to information extraction of particular unstructured documents in a structured format.

### 1.3 Importance of the research

The customization of LLMs for AI applications holds significant implications for various industries and domains. By tailoring the application to specific tasks or domains, organizations can provide more efficient and personalized user interactions, leading to enhanced customer satisfaction and productivity gains. This research contributes to expands the possibilities of leveraging LLMs for real-world applications.

### 1.4 Research objectives

The objectives of this thesis are:

- To investigate the existing tools and techniques available for customizing LLMs for specific tasks.
- To explore methods for processing domain-specific document formats, with a focus on PDF documents.
- To develop a framework for integrating customized LLMs into a user-interface application.
- To evaluate the performance and usability of the customized application.

### 1.5 Thesis structure

This chapter provided an overview of the research context and highlighted the importance of incorporating LLMs for automating information extraction. The following chapters will thoroughly examine the development, implementation, and effectiveness of LLM applications within specific domains.

The second chapter lays the theoretical foundation by exploring essential concepts in artificial intelligence and machine learning, with a specific focus on their applications in natural language processing (NLP). This discussion is crucial for understanding how Large Language Models (LLMs) are applied to extract information effectively. It also reviews related works, highlighting advancements in the use of LLMs for information extraction applications.

The third chapter details the scientific methodologies and outlines various implementation approaches discussed in the literature, with a specific emphasis on the method utilized for developing the application presented in this thesis. This chapter details the process followed for implementing the application. It specifically explains the development approach, and provides a comprehensive description of the pipeline.

Chapter four introduces the benchmarking approach designed to assess the performance of the LLM application in accurately and reliably extracting structured information. This chapter also provides a detailed explanation of the evaluation methodology used. Also, the error analysis has been used to highlight the areas that should be focused on.

The fifth chapter summarizes the conclusions drawn from the implementation described, challenges encountered during this research, and explores potential future implementations in this field.



## Chapter 2

# Background and related works

This chapter establishes a foundational understanding of essential theoretical concepts and examines recent advancements in key areas, including Artificial Intelligence (AI), Natural Language Processing (NLP), Large Language Models (LLMs), also customizing LLM techniques including fine-tuning, Retrieval-Augmented Generation (RAG), and prompt engineering.

### 2.1 Introduction to Artificial Intelligence

#### 2.1.1 Artificial Intelligence (AI)

The world is currently experiencing unprecedented technological advancements, particularly in Artificial Intelligence (AI). AI is the science of replicating human abilities by creating and using algorithms in dynamic computing environments. Its goal is to develop machines capable of tasks that usually require human intelligence, such as decision-making, problem-solving, language understanding, and pattern recognition [15].

In recent years, significant progress in AI has led to the development of tools like ChatGPT, which generates text, and creates images and videos. These innovations have made AI more accessible to the public and integrated it into daily life. Without such tools, performing routine tasks can feel inefficient and outdated, emphasizing AI's growing importance in simplifying and modernizing everyday activities [8].

Also, European Commission's Communication on AI, has defined it as systems that can act smartly by understanding their surroundings and making decisions on their own to reach certain goals. AI systems might be software-only, working in the digital world (like voice assistants, tools for analyzing images, search engines, or systems for recognizing speech and faces). AI can also be built into hardware, such as advanced robots, self-driving cars, drones, or Internet of Things (IoT) devices [3].

### 2.1.2 Machine Learning (ML)

Machine Learning (ML), a branch of Artificial Intelligence, uses algorithms to learn from data, recognize patterns, and make decisions with minimal human input. Unlike traditional programming, which relies on predefined instructions, ML adapts based on the outcomes of its actions, continuously improving its predictions with accumulated data [15].

Machine learning mainly involves creating and using computer algorithms in this area. It focuses on identifying complex patterns in data and making accurate and useful predictions based on those patterns [14].

ML is divided into four main types: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning uses labeled data to make predictions, while unsupervised learning identifies patterns in unlabeled data. Semi-supervised learning combines both labeled and unlabeled data to guide classifications, and reinforcement learning learns by trial and error to develop optimal strategies or recommendations [8].

Common ML algorithms include neural networks, linear and logistic regression, clustering, decision trees, and random forests. Its applications are vast, ranging from speech recognition and customer service chatbots to computer vision, recommendation systems, automated stock trading, and fraud detection [15].

### 2.1.3 Deep Learning (DL)

Artificial Neural Networks (ANNs) form the basis of Deep Learning (DL), a specialized subset of machine learning. Modeled after the human brain, these networks consist of interconnected nodes, or neurons, organized in layers. The goal is to mimic how biological nervous systems process information through complex neural connections. Deep learning has revolutionized machine learning by effectively handling diverse data types with minimal human input while achieving high accuracy compared to traditional techniques [15].

The term "deep" in deep learning refers to the number of layers in an artificial neural network (ANN). There are three types of layers: the input layer, which takes in the data; the output layer, which gives the result; and hidden layers, which find patterns in the data. A deep ANN has many hidden layers, unlike a simple ANN with just one hidden layer, and can handle more complex tasks [26].

Key variants of deep learning include Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs are designed for perceptual tasks like image processing, analyzing pixel patterns to identify features and classify images. RNNs, on the other hand, are built with looped connections that enable data to flow both forward and backward, making them ideal for processing sequences like text, speech, or images. They excel in tasks such as sentiment analysis, sequence prediction, and fraud detection, where understanding the context of prior data is critical [8].

As data moves through the hidden layers, simple features combine to form more complex ones. Deep learning works very well with unstructured data and is more accurate than machine learning. However, it needs a lot of training data and costly hardware and software to function effectively [26].



### 2.1.4 Natural Language Processing (NLP)

Natural Language Processing (NLP), a branch of AI, centers on enabling computers to interact with human language effectively. It involves creating algorithms and computational models that allow machines to understand, interpret, and generate language in a meaningful and context-aware manner. NLP covers tasks such as language comprehension, sentiment analysis, translation, and speech recognition, aiming to bridge the communication gap between humans and machines [10].

Information Extraction (IE) is an essential area of Natural Language Processing (NLP) that transforms unstructured text into structured data, such as identifying entities, relationships, and events. [47].

### 2.1.5 Generative AI

Generative AI, a subfield of artificial intelligence, focuses on developing models capable of creating new and coherent data based on input [41]. It includes systems that can generate various forms of content, such as text, sound, and visual media, trained on either real or synthetic datasets. A prominent example of generative AI is Generative Adversarial Networks (GANs), which excel in producing images, videos, and even literary content, showcasing its versatility across multiple domains [22].

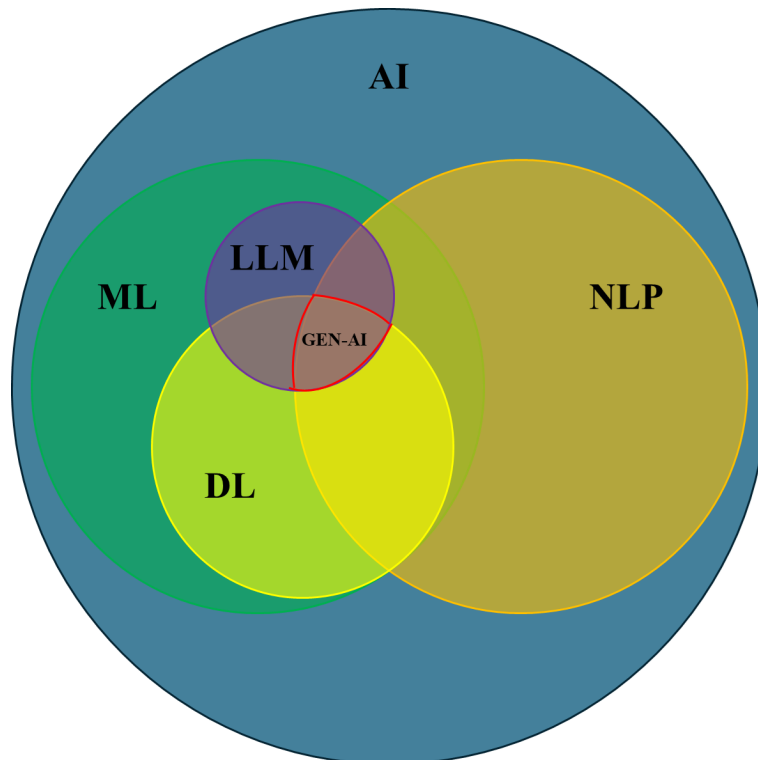


Figure 2.1. Relationship between AI, ML, DL, NLP, and LLM.

## 2.2 Introduction to LLMs

Language Modeling (LM) is a core task in NLP focused on predicting the next word or character in a sequence. It involves creating algorithms and models capable of understanding and generating coherent human language. The main goal is to learn the probability distribution of words in a language, enabling the model to generate new text, complete sentences, and estimate the likelihood of various word sequences [20].

Large Language Models (LLMs) are advanced language models powered by neural networks with billions of parameters. They are trained on massive amounts of unlabeled text data through a self-supervised learning approach, enabling them to understand and generate complex and contextually accurate text [36]. The development of Large Language Models (LLMs) advanced significantly with the introduction of the Transformer architecture in 2017, presented in the groundbreaking paper "Attention is All You Need" by Vaswani et al [43]. This architecture, based on the self-attention mechanism, improved the handling of long-range dependencies in language while enabling efficient parallel training across multiple GPUs. This innovation made it feasible to train much larger models and marked a turning point in NLP [20].

### 2.2.1 Architecture of transformer model

The Transformer architecture serves as the foundation of LLMs, designed to efficiently process sequential data without relying on iterative methods. Instead, it uses an attention-based mechanism to identify global input-output relationships. This allows the model to handle inputs of varying lengths and adjust its focus accordingly. The architecture is composed of seven key components, each contributing to its functionality [36].

The attention mechanism is a critical component of LLMs, enabling the model to focus on the most relevant parts of the input data. Not all tokens equally contribute to understanding the context or meaning of a sentence. The attention mechanism assigns varying weights to token embeddings based on their importance within the context. For example, in the sentence, 'The captain, against the suggestions of his crew, chose to save the pirate because he was touched by his tale,' key tokens like 'captain,' 'save,' and 'pirate' are assigned higher weights as they are essential to the overall meaning. This prioritization of significant components enhances the model's ability to interpret and generate contextually accurate outputs.

The Transformer architecture processes text data by breaking it into tokens (words or sub-words), converting these into numerical representations called input embeddings. These embeddings operate like a dictionary, enabling the model to grasp the meaning of words by positioning them in a mathematical space where similar phrases are placed near each other. The model is trained to create these embeddings such that vectors of the same dimensions represent words with comparable meanings. Positional encoding adds sequence information, ensuring the model understands word order, which is vital for grammatical and semantic accuracy.

The encoder processes input text using self-attention layers to create meaningful hidden states representing relationships within the input. The decoder, crucial for generating

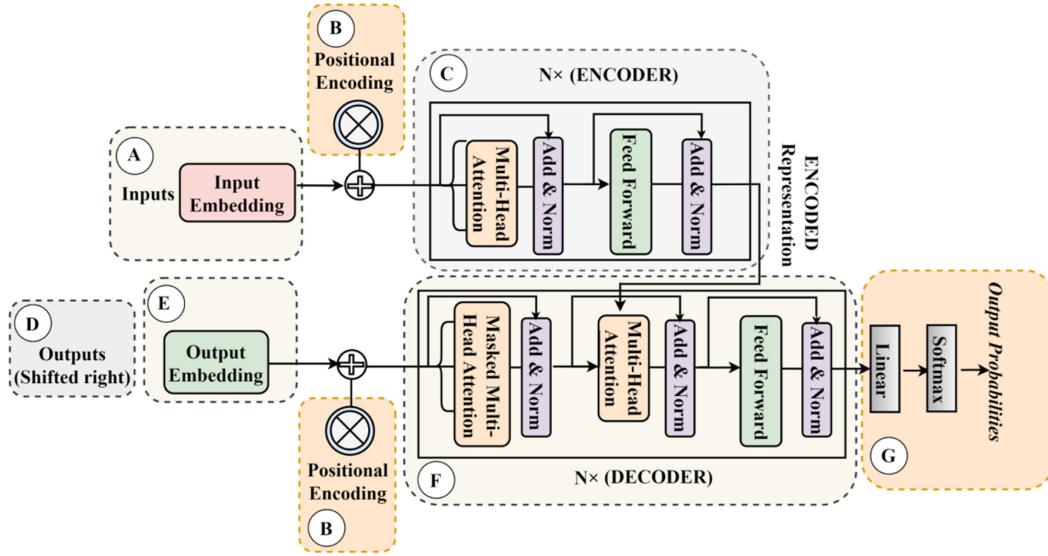


Figure 2.2. Architecture of transformer models [36].

output sequences, uses positional encoding and attention mechanisms to process embeddings and produce coherent text. In GPT models, the decoder functions independently, leveraging a masked self-attention mechanism to predict sequences without explicitly using an encoder.

Input embeddings are transformed into output embeddings, ensuring the model’s predictions align with the input format. During training, the loss function adjusts model parameters to reduce errors and improve accuracy. The linear layer and softmax function transform output embeddings into a probability distribution over vocabulary tokens, generating final outputs.

Transformers rely on core components such as tokenization, attention mechanisms, positional encoding, encoders, and decoders, combined with advanced training techniques, to generate accurate and contextually relevant text [36].

### 2.2.2 Taxonomy of LLM tasks

Large Language Models (LLMs) are useful for many natural language tasks like writing, summarizing, translating, and finding information [20]. This section explains how LLMs are used in different areas.

- **Question Answering (QA):** QA systems let people ask questions in plain language and get direct answers. LLMs are important for making these systems strong and reliable. They are first trained on a large amount of text and then adjusted using specific question-answer datasets. This helps them understand questions and find or create answers from text. QA systems powered by LLMs can be used in voice

assistants, search engines, and other tools to give quick and clear answers through natural conversations.

- **Text Generation:** LLMs can automatically create text for many purposes, like writing articles, blogs, research papers, social media posts, product descriptions, source code, and emails. These models understand and generate natural language, producing accurate and clear content.
- **Language Translation:** LLMs can translate text between languages with high accuracy and fluency.
- **Text Classification:** LLMs can organize and classify text into categories or topics. This makes them useful for tasks like analyzing sentiment, detecting spam, moderating content, and processing customer feedback.
- **Summarization:** LLMs can create summaries of content like news, research papers, legal documents, and more. This is helpful for quickly understanding key information.
- **Chatbots and Virtual Assistants:** LLMs play a big role in chatbots and virtual assistants by understanding user questions, giving relevant answers, and holding natural conversations. These tools help with customer support, recommendations, answering queries, and automating tasks, improving user experience and efficiency.
- **Information Extraction (IE):** LLMs can pull out structured information from text to create knowledge graphs. This helps businesses and organizations manage large amounts of data more effectively.
- **Dialogue Systems:** LLMs are transforming how people interact with technology by creating more engaging and efficient conversational experiences.
- **Semantic Search:** LLMs improve search systems by understanding the meaning behind words instead of just matching keywords. This leads to more accurate and useful search results.
- **Speech Recognition:** LLMs also help with converting spoken words into text, which is important for voice assistants and transcription tasks [20].

### 2.2.3 Domain-specific applications of LLMs

Pre-trained LLMs can be fine-tuned or trained to handle specific tasks across different fields. Research has shown their effective use in various areas like healthcare, finance, education, forecasting, and natural language processing. These experiments are changing how AI is used in these domains. This section explains how LLMs are applied in different areas.

- **Bio-Medical and Healthcare:** GPT-3 is very helpful in healthcare. It can be trained to handle customer service tasks, reducing the need for human staff. For example,

robots powered by GPT-3 can replace human receptionists, which was especially important during the COVID-19 pandemic to reduce infection risks.

- **Education:** According to Kasenci et al., LLMs have made a big difference in education by supporting personalized learning, automating grading, and making educational resources more accessible. They can also create error-free essays, summaries, and articles in different formats. XLNet is particularly useful for understanding texts and documents, which benefits academic work.
- **Social Media:** LLMs improve various aspects of social media, including creating content, moderating discussions, and analyzing sentiment. They can write posts, classify text, and even generate full blogs and articles. Additionally, they perform tasks like named entity recognition (NER) and provide content suggestions. They also help identify and filter harmful or inappropriate content, creating a safer online environment.
- **Business:** In business, LLMs enhance decision-making, production, operations, and customer service. They assist companies in providing 24/7 support by answering customer queries, offering advice, and analyzing market trends, customer sentiments, risks, and competition. Models like GPT, XLNet, and BERT help create documents, product descriptions, and manage tasks efficiently, saving time and effort. Frederico et al. explored how ChatGPT can impact supply chain management.
- **Agriculture:** LLMs like GPT-3, BERT, and XLNet are highly useful in agriculture. They analyze large datasets, including soil, crop, and weather data, as well as satellite images. These models give advice on planting schedules, irrigation, fertilization, and resource optimization. They also help farmers stay updated on market trends, predict crop prices, anticipate natural disasters, and document agricultural data.

## 2.3 Foundation model selection criteria

When selecting a model, it is essential to consider factors such as its architecture, size (number of parameters), and the diversity and scope of its training data. After identifying a suitable base model, the next step involves determining an appropriate customization method. Depending on the complexity of the task and the desired level of performance, approaches such as fine-tuning, Retrieval-Augmented Generation (RAG), or prompt engineering can be employed. [32].

### Model size

The size of a model, typically measured by the number of parameters, plays a crucial role in determining its performance and resource requirements. Larger models are capable of capturing complex patterns and producing more accurate results but require greater computational power for both training and inference. Selecting the appropriate model size involves balancing the desired level of accuracy with available computational resources. For simpler tasks or when resources are limited, smaller models may suffice, whereas

more complex tasks often benefit from larger models. The number of parameters, which represent the weights and biases adjusted during training, also provides an estimate of the computational cost and inference speed. Generally, a higher number of parameters increases the performance cost and slows down inference speed, making the choice of model size dependent on the specific use case [32].

### Modality

Modality refers to the type of data a model processes, such as audio, text, images, or video (which combines audio and sequential images, making it inherently multimodal). Traditionally, most AI models have been designed to handle a single type of data for a specific task. In contrast, multimodal models can process and integrate multiple types of data at the same time, enabling them to deliver more comprehensive and accurate results [39].

Large Language Models (LLMs) represent the latest advancement in the effort to make Artificial Intelligence (AI) understand and use human language effectively. Text is the most common input for LLMs due to its abundance and ease of processing. However, Multimodal Language Models (MLMs) are emerging as a promising area of AI research, combining various input types to improve understanding and generate more sophisticated responses [39].

### Computational resources and model capability

A major challenge in customizing LLMs is achieving a balance between the available computational resources and the desired model capabilities. Large models need substantial computational power for both training and inference, which can be a significant limitation for many organizations. Customization methods, such as fine-tuning and retrieval-augmented generation, often require even more resources. To make LLM customization more accessible, advancements in efficient training techniques and model architectures are crucial [32].

### Datasets for model training

Pre-training LLMs involves using diverse datasets to improve their generalization capabilities. Commonly, these datasets include web text, conversational data, and books, with additional specialized data, such as code or scientific information, to enhance performance in specific domains. Below is a summary of key data sources often used for LLM training [30]:

- **Books:** Datasets like BookCorpus and Gutenberg provide a variety of genres, including novels, essays, poetry, and philosophy. These datasets help models understand language across diverse topics and writing styles.
- **CommonCrawl:** CommonCrawl provides an extensive web archive that contains over 250 billion web pages accumulated since 2007, with monthly updates that add 3 to 5 billion new pages. Due to the presence of low-quality data, pre-processing

is essential. The widely used filtered datasets derived from CommonCrawl include C4, CC-Stories, CC-News, and RealNews.

- **Reddit Links:** The Reddit system of user-voted posts makes it a valuable resource to create high-quality data sets, using community moderation to filter content.
- **Wikipedia:** With comprehensive content on various topics, Wikipedia is extensively used for training LLMs. Its availability in multiple languages makes it valuable for multilingual training.
- **Code:** Publicly available code datasets are limited and primarily sourced from platforms like GitHub and Stack Overflow through web scraping of open-source licensed code.

These diverse datasets, when carefully curated and preprocessed, serve as foundational resources for training LLMs to achieve broad language understanding and task performance [30].

### Model architecture

The Transformer architecture is ideal for scaling up models, with research showing that increasing the size of models or training datasets significantly improves performance. Modern LLMs, built on the Transformer framework, scale to tens of billions or even trillions of parameters, pushing the limits of Pre-trained Language Model (PLM) performance [30].

PLM architectures are categorized into Encoder-decoder and Decoder-only types, as Encoder-only models are no longer used in recent LLMs. Here is an overview of the two primary architectures:

- **Encoder-decoder Architecture (Sequence-to-Sequence Models):** This architecture consists of an encoder, which encodes the input sequence using multiple layers of Multi-Head Self-Attention, and a decoder, which generates output by attending to the encoder’s representations. Encoder-decoder models, like T5, flan-T5, and BART, are pre-trained by masking certain words in the input text and predicting them. These models are ideal for tasks like translation, text summarization, and generative question answering.
- **Decoder-only Architecture (Autoregressive Language Models):** Decoder-only models, such as GPT-4, rely solely on the decoder component of the Transformer. They generate text by predicting the next word based on previous tokens, making them well-suited for text generation tasks. This architecture includes two variants: the Causal Decoder, which generates text step by step, and the Prefix Decoder, which leverages additional context. Since 2021, most new LLMs have adopted the decoder-only design, with companies like Microsoft still producing encoder-decoder models while others focus on decoder-only systems [9].

In summary, encoder-decoder models excel in tasks requiring understanding and generation, such as translation and summarization, while decoder-only models dominate text generation tasks due to their autoregressive design.

Model series	Type
GPT	Decoder-only
LLaMa	Decoder-only
Mistral	Decoder-only
Bart	Encoder-decoder
Falcon	Decoder-only

Table 2.1. LLMs architecture types

## Licenses

A commercially licensed model can be utilized for business applications, allowing it to be integrated into a commercial platform for various purposes.

A proprietary license is a non-open source license that grants restricted rights to use the software under specific terms and conditions. Typically, it requires users to pay a fee or obtain permission for access and usage. Such licenses often include limitations on how the software can be used or modified and may prohibit sharing or distributing the software or its outputs without prior authorization.

Apache 2.0 License requires users to credit the original authors, include a copy of the license, and disclose any modifications made to the software. Additionally, users are prohibited from using the software’s trademarks or logos without prior permission.

Meta Llama Community License Agreement allows free usage for organizations with fewer than 700 million users. However, it restricts the use of LLaMA outputs for training other LLMs, except for LLaMA itself or its derivatives [49].

The Falcon License 2.0 is a modified version of the Apache 2.0 License, introduced for the Falcon language model. This license includes additional clauses, notably an Acceptable Use Policy (AUP) that users must adhere to. The AUP can be updated periodically, and users are responsible for ensuring their compliance with the most current version [23].

Model Series	License	Usable for Commercial Purposes?
GPT-4	Closed-source	Yes (via API access)
LLaMA	Custom LLaMA License	Yes (approval needed for large entities)
Mistral	Apache 2.0	Yes
Falcon	Custom Falcon License 2.0	Yes (Subject to accept use policy)

Table 2.2. Overview of LLMs and Their Licenses



### Cost of initial set-up

The initial setup cost includes expenses related to storing the model and processing predictions for query requests.

- For API Access Solutions: Popular providers like OpenAI, Cohere, and Vertex AI offer API access with varying pricing models based on the use case (e.g., summarization, classification, embedding) and the selected model. Billing is typically determined by the number of input tokens (input cost) and generated output tokens (output cost). In some cases, there may also be a charge per request. For instance, using Cohere models for classification tasks costs 0.20 dollar per 1,000 classifications.
- For On-Premise Solutions: Hosting open-source models on-premise can be challenging due to the large size of model parameters, which places significant demands on IT infrastructure. The primary setup cost involves establishing an appropriate infrastructure capable of hosting the model [27].

Model Categories:

- Light-weight LLMs: With around 7 billion parameters, these can run locally on personal computers.
- Large-sized LLMs: These require hosting on cloud platforms such as AWS, Google Cloud Platform, or dedicated GPU servers due to their higher resource requirements [27].

## 2.4 Strategies and techniques for customizing LLMs

Customizing Large Language Models (LLMs) for specific tasks or applications is essential for their effective use across different domains. This process adjusts the model output to match the required context, greatly improving its usefulness and efficiency. This section explores key techniques for tailoring LLMs, emphasizing their importance and role in improving performance for specialized tasks.

### 2.4.1 Fine-tuning

Transfer learning enables a model to apply the general knowledge it gained during pre-training to a specific task. Pre-training provides the model with a broad understanding of patterns and structures in data, which serves as a strong foundation. During transfer learning, the model fine-tunes this knowledge, requiring minimal adjustments to perform well on the new task. This approach is more efficient than training a model from scratch, as it needs less data and computational power, and it often delivers better results, especially when task-specific data is limited. This two-step process, pre-training and fine-tuning is a key factor behind LLMs' exceptional performance across diverse tasks and data types.

Fine-tuning involves taking a pre-trained model, like OpenAI’s GPT series, and training it further using a smaller dataset specific to a particular domain. This method leverages the model’s existing knowledge to improve its performance on specialized tasks while requiring less data and computational resources. By transferring the patterns and features learned during pre-training, fine-tuning enhances task-specific performance and minimizes the need for extensive training data. It is widely used in NLP tasks such as text classification, sentiment analysis, and question-answering [33]. Fine-tuning adapts a model’s knowledge for specific applications, such as text classification or language generation, by adjusting its parameters using task-specific data [10].

### **Supervised Fine-Tuning (SFT)**

Supervised Fine-Tuning (SFT) adapts pre-trained foundational models, such as LLMs and MLLMs, to domain-specific tasks by using labeled training data. While these models are typically trained on large-scale unsupervised data, SFT adjusts their weights and parameters to optimize performance on specific tasks within a particular domain [12].

SFT is a more efficient alternative to training a model from scratch, requiring less training data and computational resources while achieving significant results. It simplifies the process of aligning models with specialized tasks, making them highly effective in domain-specific applications [12].

### **Transfer learning**

Transfer learning improves machine learning performance on new tasks by utilizing the knowledge of a pre-trained model. Instead of training from scratch, the model uses previously learned representations as a foundation, saving time, data, and resources. This method enhances accuracy and helps the model adapt more effectively to new tasks or domains, leveraging prior learning to achieve better results on the target task [8].

### **Parameter-Efficient Fine-Tuning (PEFT)**

Parameter Efficient Fine-Tuning (PEFT) optimizes the fine-tuning process by reducing computational demands and training time, updating only a small subset of a pre-trained model’s parameters while maintaining comparable performance to full fine-tuning. Unlike traditional fine-tuning, which adjusts all model parameters, PEFT freezes most parameters and modifies specific parts, enabling efficient fine-tuning on standard consumer hardware with significantly lower costs. It also mitigates risks like catastrophic forgetting [12].

- **Low-Rank Adaptation (LoRA):** LoRA applies low-rank updates to the model’s parameters, allowing for efficient fine-tuning on standard hardware. This approach adjusts smaller matrices (LoRA adapters) to approximate the larger weight matrices of the model [17].
- **Quantized Low-Rank Adaptation (QLoRA):** QLoRA extends LoRA by incorporating quantization, which reduces the precision of numerical values in the model, further compressing the model while maintaining performance. It enables fine-tuning

of large models, such as a 65B parameter model, using significantly less memory (e.g., on a single 48GB GPU). By combining low-rank updates and quantization, QLoRA reduces storage and memory requirements, making fine-tuning more efficient [12].

Overall, PEFT approaches enhance the adaptability of Large Language Models for downstream tasks while saving computational resources and preserving model performance [10].

### 2.4.2 Retrieval-Augmented Generation (RAG)

In mid-2020, Lewis et al. [29] introduced RAG (Retrieval-Augmented Generation), a notable development in improving generative tasks using LLMs. RAG incorporates an initial step where the model retrieves relevant information from an external data source before generating text or answering questions. This approach enhances the accuracy and relevance of outputs by integrating external data retrieval into the generative process. By dynamically accessing knowledge bases during inference, RAG enables a more informed and evidence-based text generation process, reducing the likelihood of hallucinations and improving overall output quality [6].

A basic RAG pipeline can be divided in three sections including ingestion, retrieval, and synthesis.

#### Ingestion

In this step, documents are divided into chunks (each chunk being a group of consecutive sentences) using a text splitter. An embedding function then generates embeddings for each chunk, creating an n-dimensional vector for every section. These chunks, along with their corresponding embedding vectors, are stored in a vector database. The vector database facilitates the retrieval of relevant chunks based on a query, enabling efficient processing of lengthy documents [4]. Alternatively, knowledge graphs (KGs) can be used instead of vector databases for retrieval and generation processes. For example, GraphRAG (Graph-based Retrieval-Augmented Generation) is an innovative method that leverages KGs to improve NLP tasks like question-and-answer (QA) systems. By integrating KGs with RAG techniques, GraphRAG enables more accurate and context-aware responses using structured information extracted from financial documents. However, it may underperform in abstractive QA tasks or when the question lacks an explicitly mentioned entity [38].

#### Retrieval

In the Retrieval-Augmented Generation (RAG), efficient retrieval of relevant documents from the data source is essential [16]. To identify information relevant to a user query, the system generates an embedding for the query (an n-dimensional vector representation of the query) and computes its similarity with the embeddings of all chunks in the vector database. Common similarity metrics, such as cosine similarity, are used to rank the chunks in descending order of relevance [10]. In contrast to traditional RAG, GraphRAG

retrieves graph elements containing relational knowledge relevant to the query from a pre-built graph database. These graph elements can include nodes, triples, paths, or subgraphs, providing a structured approach to accessing and utilizing knowledge for answering queries [34].

### Synthesis (response generation)

Finally, after retrieval, the retrieved context is combined with the query to create a prompt, which is then provided to the LLM. In this step, the model integrates the retrieved information with its pre-trained knowledge to produce coherent and contextually appropriate responses [4].

### Multi-modal RAG

The most prominent and widely used LLMs are primarily text-based, although some include multi-modal capabilities. A specialized category of these models, known as vision models, is designed to integrate text and image data into a unified embedding space. Vision models enable seamless applications such as automatic text-to-image generation and image captioning [12].

Image-grounded multi-modal LLMs (MLLMs), often referred to as Large Vision-Language Models, typically consist of a vision encoder, a language encoder, and a cross-modal alignment network. These models provide a more robust approach to multi-modal generative AI compared to traditional text-based LLMs, some of which incorporate multi-modal features as an add-on rather than being inherently designed for it [12].

Multimodal Retrieval-Augmented Generation (RAG) expands the capabilities of LLMs by integrating text, images, audio, and video into the generation process. These systems retrieve relevant information from external sources and incorporate it into the model's prompt, allowing for more accurate and context-aware responses. This approach is especially valuable for processing and generating content from various data formats.

There are different approaches to building multi-modal RAG pipelines:

- Embed all modalities into the same vector space: It simplifies integrating text and images in a retrieval pipeline. Models like CLIP can encode both text and images into the same vector space, allowing the use of text-only RAG infrastructure with minimal adjustments. The main change involves swapping the embedding model to support additional modalities. For generation tasks like question answering, the large language model (LLM) is replaced with a Multimodal LLM (MLLM). While this approach streamlines the pipeline, it requires a model capable of accurately embedding diverse data types, including handling complex features like text within images and detailed tables [42].
- Ground all modalities into one primary modality: It streamlines the retrieval and processing pipeline by focusing on the dominant data type relevant to the application. For instance, in a text-based QA system over PDFs, text is processed conventionally, while images are preprocessed into text descriptions and metadata,

with the original images stored for future reference. During inference, retrieval primarily relies on the text descriptions and metadata, with answers generated using a combination of LLMs and MLLMs, depending on the retrieved image type. This approach leverages the rich metadata from images to answer objective questions effectively, avoiding the need to train new embedding models or re-rank results across modalities. However, it comes with trade-offs, including the cost of preprocessing and the potential loss of nuanced information from images [42].

- Have separate stores for different modalities: In this case, each modality is stored independently, and queries retrieve the top-N chunks from each store. A multimodal re-ranker then identifies the most relevant chunks across all modalities. This method simplifies the modeling process by avoiding the need to align a single model across multiple modalities. However, it introduces added complexity by requiring a re-ranker to organize and prioritize the combined results from multiple modalities, which can include up to  $M*N$  chunks (N chunks per M modalities) [42].

Multi-modal RAG approaches	Ingestion	Retrieving	Generation
Embed all modalities into the same vector space	Multi-modal embedding	Retrieve raw image and text	Multi-modal generation
Ground all modalities into one primary modality	Text embedding	Retrieve image summary and text	Text generation
Have separate stores for different modalities	Text embedding	Retrieve image summary and text + Raw image from document store	Multi-modal generation

Table 2.3. Options to create multi-modal RAG

### Advanced RAGs

Evaluating the components of a RAG system helps researchers pinpoint areas that need improvement, enabling more efficient and accurate information retrieval and response generation in LLMs. Although RAG is a practical addition to LLMs, its effectiveness depends heavily on the relevance and accuracy of the retrieved documents and generation [50]. This is where advanced RAG systems are needed.

- CRAG: Corrective Retrieval Augmented Generation (CRAG) is a lightweight retrieval evaluation system designed to assess the quality of documents retrieved for a query. It provides a confidence score, which can trigger different knowledge retrieval actions. To address limitations of retrieval from static or restricted corpora, large-scale web searches are incorporated to enhance retrieval results. Additionally, an algorithm is used to refine retrieved documents, selectively extracting key information while filtering out irrelevant details [50].

- **Self-RAG:** Self-Reflective Retrieval-Augmented Generation (Self-RAG) improves the quality and factual accuracy of language models by combining retrieval and self-reflection. This approach trains a single language model to dynamically retrieve relevant passages as needed and reflect on both the retrieved content and its own generated responses using special markers called reflection tokens. These tokens make the model more controllable during inference, allowing it to adjust its behavior to meet various task requirements [7].

### 2.4.3 Prompt engineering

Prompt engineering has become a key method for improving the performance of pre-trained large language models (LLMs) and vision-language models (VLMs). It involves creating task-specific instructions, called prompts, to guide the model’s output without modifying its parameters. This approach enhances the adaptability of LLMs and VLMs, enabling them to handle a wide variety of tasks and domains effectively [37].

Unlike traditional methods that require retraining or extensive fine-tuning, prompt engineering adjusts model behavior through carefully crafted instructions, offering a more efficient and flexible alternative. It allows these models to excel across diverse applications without significant modifications. Continued research in this area introduces innovative techniques, highlighting the growing importance of prompt engineering in shaping AI’s adaptability and utility across sectors [37].

### In-Context Learning (ICL)

In-context learning is a method where language models can learn tasks by being provided with only a few examples as demonstrations [13]. In-context learning, introduced in the original GPT-3 paper, enables language models to perform tasks by providing only a few examples. In this approach, a prompt is created containing input-output pairs that demonstrate the task. A test input is then added at the end of the prompt, and the model predicts the next tokens based on the provided examples. To make accurate predictions, the model must understand the input distribution (e.g., financial or general news), output distribution (e.g., positive/negative sentiment or topics), input-output relationships (e.g., sentiment or topic classification), and formatting [46].

- **Zero-shot prompting:** Zero-shot prompting eliminates the need for extensive training data, instead relying on well-designed prompts to direct the model towards performing new tasks. In this approach, the model is provided with a task description within the prompt but does not have labeled data for specific input-output examples. The model then uses its pre-trained knowledge to make predictions for the new task based solely on the given prompt.
- **Few-shot prompting:** Few-shot prompting involves providing a model with a small number of input-output examples to help it understand a specific task, unlike zero-shot prompting, which offers no examples. Even a few high-quality examples can significantly improve model performance on complex tasks. However, including these examples requires additional tokens, which can become a limitation for longer

inputs. The selection and arrangement of examples in the prompt are also crucial, as they can influence the model’s behavior, with potential biases like favoring common words still affecting the results. While few-shot prompting enhances the capabilities of large models, careful prompt design is essential to optimize performance and minimize unintended biases [37].

- **Instruction-based prompting:** Instruction Prompting focuses on the ability of generative AI models, particularly large language models (LLMs), to follow instructions written in natural language. This approach enables models to handle new and previously unseen tasks by interpreting and executing these instructions without needing task-specific training data.

### **Improve reasoning capabilities**

Large Language Models (LLMs) often struggle with complex reasoning, which restricts their potential. However, research has demonstrated that when prompted effectively, sufficiently large models can develop reasoning abilities to address such challenges [37].

- **Chain-of-thought (CoT):** it reasoning enables LLMs to perform step-by-step reasoning, improving their ability to handle complex tasks through in-context learning. CoT prompting breaks problems into smaller steps, provides an interpretable reasoning process, and is widely applicable to tasks like math problems, common-sense reasoning, and symbolic manipulation. It can be easily implemented in large pre-trained models by including CoT examples in few-shot prompts, making it a valuable method for enhancing reasoning capabilities [45].
- **Tree of Thoughts (ToT):** The Tree of Thoughts (ToT) approach builds upon and generalizes the Chain of Thoughts (CoT) method, enabling language models to explore coherent units of text, referred to as ‘thoughts,’ which act as intermediate steps in problem-solving. ToT actively maintains a structured tree of these thoughts, where each thought represents a meaningful language sequence. This structure allows the model to self-evaluate progress by assessing how each intermediate idea contributes to solving the problem through a deliberate reasoning process expressed in natural language [51].
- **ReAct:** The ReAct approach enables the generation of reasoning traces and task-specific actions in an interleaved manner, creating a stronger synergy between the two. Reasoning traces help the model develop, monitor, and adjust action plans while addressing exceptions. Actions, on the other hand, allow the model to interact with external sources like knowledge bases or environments to gather additional information. ReAct has been applied to a wide range of language and decision-making tasks, demonstrating superior performance compared to state-of-the-art methods while enhancing human interpretability and trust in the model [52].



## 2.5 Tools for creating LLM applications

With the growing popularity of LLMs, numerous tools, techniques, and methods have emerged for building LLM-based assistants. These tools vary in their level of control, customization options, and ease of use. Below is a brief overview of some existing approaches for developing AI solutions using LLMs [10].

### 2.5.1 LLM-app platforms

An LLM App Platform is designed to simplify the creation, deployment, and optimization of applications powered by Large Language Models (LLMs). These platforms offer a comprehensive suite of tools and services that support building, evaluating, and deploying LLMs for various practical uses. They provide advanced language models and the necessary infrastructure, including memory and computational resources, to facilitate a wide range of applications such as chatbots, content generation, search optimization, and data analysis. By leveraging these platforms, developers and businesses can seamlessly integrate powerful AI capabilities into their products and services, ensuring efficient and scalable performance. Additionally, these platforms are crucial for tasks like training, fine-tuning, and deploying LLMs at scale, making them essential for NLP and machine learning workflows.

#### Hugging Face

Hugging Face is a company focused on developing tools for building machine learning applications. It is best known for its Transformers library, which is widely used for natural language processing tasks. The platform also allows users to share machine learning models and datasets, providing a dedicated space to showcase their projects. Hugging Face additionally offers an implementation of the LoRA technique, which can be used for fine-tuning models efficiently [10]. The Transformers library offers APIs and tools to easily access and train state-of-the-art pre-trained models. Utilizing pre-trained models can significantly lower computational costs, reduce environmental impact, and save the time and resources needed to train a model from the ground up.

#### Open-AI platform

The OpenAI Developer Platform enables developers to create and utilize various AI solutions powered by GPT models. It provides both a web-based graphical interface and an API, supporting tasks such as fine-tuning GPT models and developing RAG-based solutions [10].

#### Groq

Groq is designed to address the growing demand for AI model deployment and inference, offering instant and efficient intelligence for developers and enterprises. It provides fast AI inference both in the cloud and on-premises AI compute centers. The core of this technology is the Groq Language Processing Unit (LPU), specifically built for AI inference and



language tasks, unlike GPUs, which were originally created for graphics processing. The LPU delivers high-speed performance, cost-effectiveness, and energy efficiency at scale. Groq's technology is accessible through GroqCloud for general users, while enterprises and partners can opt for deployment in either cloud-based or on-premises AI compute centers.

### **Ollama**

One way to use open models is by downloading them from the Hugging Face platform and running them through Python scripts. Alternatively, software like Ollama now provides an easier method by enabling users to access these models within a dedicated environment. With Ollama, users can specify the desired model or models and run them locally on their computers without the need for complex scripting [19].

### **Llama.cpp**

llama.cpp is an open-source C++ library by Georgi Gerganov that streamlines the deployment and inference of large language models (LLMs), addressing their high computational demands. Its main goal is to optimize LLM performance using advanced quantization techniques, reducing model size and computational requirements for faster and more efficient inference. Supporting models like Meta AI's LLaMA family, llama.cpp makes LLMs accessible across various platforms, including personal computers, laptops, and mobile devices, expanding their usability even in environments with limited computational resources [31].

### **Bitnet.cpp**

bitnet.cpp is an inference framework designed for 1-bit LLMs, such as BitNet b1.58 models. It enables lossless inference while optimizing speed and energy efficiency. The initial version of bitnet.cpp supports inference on CPUs [44]. At the moment of writing this thesis, bitnet.cpp only supports 3 models.

## **2.5.2 Vector databases**

A vector database is a specialized type of database that stores data as high-dimensional vectors, which are mathematical representations of features or attributes. The number of dimensions in each vector can vary from tens to thousands, depending on the complexity and detail of the data being represented [21].

Vector databases offer distinct advantages over traditional databases, particularly for modern AI and data science applications:

- **Efficient Similarity Search:** Vector databases excel in finding similar or relevant data using vector distances, enabling applications like natural language processing, computer vision, and recommendation systems. Unlike traditional databases, which rely on exact matches or predefined criteria, vector databases capture semantic and contextual meanings.

- **Support for Complex and Unstructured Data:** They handle complex, unstructured data such as text, images, audio, and video by converting these into high-dimensional vectors that represent their features or attributes. This flexibility overcomes the limitations of rigid schemas in traditional databases.
- **Scalability and High Performance:** Vector databases are optimized for large-scale, real-time data processing.

In summary, vector databases are highly suited for AI-driven tasks, offering efficient, flexible, and scalable solutions for handling complex and large datasets [21].

### 2.5.3 LLM orchestration frameworks

LLM orchestration frameworks offer a high-level interface for managing and controlling large language models (LLMs). These frameworks simplify complex tasks such as prompt generation, resource management, and performance monitoring, allowing developers to interact with LLMs more easily. By streamlining the development and deployment process, orchestration frameworks enhance the performance, reliability, and efficiency of LLM-based applications [28].

#### **LlamaIndex**

LlamaIndex is a framework designed to streamline the process of developing LLM-based applications. These applications often face challenges in managing data from diverse sources, which can vary widely in format, ranging from highly structured to unstructured. LlamaIndex focuses on efficiency and simplicity, particularly in search and retrieval tasks, offering a conversational interface for seamless interaction. More than just a tool, it serves as an intuitive platform that enables developers to manage, search, and summarize documents effectively by leveraging LLMs and innovative indexing techniques [18].

#### **LangChain**

LangChain is an open-source Python framework that provides a high-level API for designing workflows with large language models (LLMs). It simplifies complex interactions such as prompting, chaining, and conditional branching, making it easier to build and deploy LLM-powered applications. By offering pre-built chains and essential building blocks, LangChain streamlines the development process, enabling developers to create LLM-based solutions efficiently and effectively [40].

#### **LangGraph**

LangGraph is a library designed for building stateful, multi-actor applications with LLMs, supporting the creation of agent and multi-agent workflows. It stands out from other LLM frameworks by offering key features: cycles, controllability, and persistence. Unlike DAG-based solutions, LangGraph enables the definition of flows with cycles, which are essential for agent-based architectures. As a low-level framework, it provides precise control over both the workflow and application state, making it ideal for developing reliable agents.

Additionally, LangGraph includes built-in persistence, supporting advanced features like human-in-the-loop workflows and memory management. Built on top of LangChain, LangGraph adds cyclic computational capabilities, allowing for more dynamic, agent-like behaviors where LLMs can operate in loops and determine the next action iteratively.

## 2.6 Related work

The advent of Large Language Models (LLMs) marks a remarkable advancement in natural language processing, enabling sophisticated analysis and comprehension of complex text data. Recent studies highlight their growing role in tasks like information extraction, where LLMs demonstrate the ability to identify, categorize, and summarize relevant content with high accuracy. This section examines the advancements in LLM technology, emphasizing their applications in automating processes such as document analysis, data extraction, and knowledge representation, particularly in domains requiring nuanced understanding, like legal and administrative documents.

### 2.6.1 Large Language Models for Generative Information Extraction: A Survey

Xu, et al (2024) in their research showed that information extraction (IE) focuses on deriving structured knowledge from unstructured natural language texts. Recent advancements in generative Large Language Models (LLMs) have showcased their exceptional abilities in text comprehension and generation, inspiring the adoption of a generative approach for IE tasks. Several studies have explored the integration of LLMs in this domain, offering insights into their application across various IE subtasks and techniques. A comprehensive review of these developments highlights emerging trends, evaluates state-of-the-art methods, and identifies potential research directions. These studies provide valuable insights into techniques and opportunities for further exploration in the field [47].

### 2.6.2 Exploring Open Information Extraction for Portuguese Using Large Language Models

The study conducted by Cabral, et al (2024), has revealed notable potential despite the field’s primary focus on English. While OpenIE methods have been extensively optimized for English, few studies investigate their cross-lingual and multilingual applications, with Portuguese OpenIE remaining an underexplored area. Recent work has addressed this gap by assessing the use of both open and commercial LLMs with few-shot prompt engineering for Portuguese OpenIE tasks. The findings indicate that LLMs can achieve performance metrics comparable to state-of-the-art systems. Moreover, advancements like the development and fine-tuning of an open LLM model, PortOIE-Llama, demonstrate that tailored models can outperform commercial alternatives, highlighting the promise of LLMs in Portuguese OpenIE and the potential benefits of further refinement and training of larger models in this context [11].

### **2.6.3 Exploring the Potential of Lightweight LLMs for Medication and Timeline Extraction**

Fornasiere (2023), in his work, explores the significant growth in clinical text data due to the widespread adoption of electronic health records, presenting both challenges and opportunities in healthcare. This unstructured data, crucial for enhancing clinical decisions, research, and patient care, is cumbersome and time-consuming to process manually. His dissertation investigates the use of simplified language models (LLMs) to automate the extraction of medication and timeline information from clinical texts. He evaluated various prompting techniques such as zero-shot, few-shot, and sequential prompting, alongside different output formats. His findings indicate that LLMs are effective in extracting medication details and adept at handling diverse date formats through strong contextual understanding. He also introduced a line-number referencing system to improve the transparency and reliability of the process. Fornasiere’s research demonstrates that focusing on prompt tuning rather than extensive model training can greatly enhance the efficiency and accuracy of clinical data processing with LLMs [15].

# Chapter 3

## Methodology

The methodology section explains the systematic approach used in this research, describing the main steps and components of the pipeline designed to extract medication and timeline information from clinical texts. Additionally, a demo application is introduced to demonstrate the pipeline's features through an interface.

### 3.1 Case study

Manual data collection from unstructured sources is often time-consuming and labor-intensive, particularly for Italian SMEs, which must dedicate significant effort to identifying suitable grants and funds by manually reviewing various documents. To address this challenge, this case study explores the use of an LLM-enabled application to enhance the efficiency and accuracy of data extraction. The proposed application processes PDF files related to grants from Italian regions and extracts key information into a structured JSON format, referred to as "Rule Card". This approach demonstrates the potential of LLM-based systems in automating and streamlining the grant discovery process for SMEs.

This thesis aims to develop an LLM-based application to extract a structured JSON file for "Rule Card" with the following fields:

- **titolo (title):** A standardized title derived from the document, adhering to naming conventions (e.g., provider, fund type, specific intervention). It includes:
  - Grant provider/type of funds/regional program/ministry (e.g., CSR, PR FESR, PR FSE, CCIAA, Regional Law no./year, Foundation x, municipality of, etc.)
  - If present, the name of the grant (e.g., Digitalization Voucher Grant 2024)
  - Specific intervention/action (e.g., CSR 2023-2027. Interventions SRD01 and SRD02 or PR FESR Action 1.3.1.)
  - Type of contribution, choosing from the following: Non-repayable grant, low-interest loan, zero-interest loan, guarantee, tax credit, tax bonus
  - Subject of the grant (e.g., non-repayable grant to support investments for SMEs and professionals).

If specified: year and/or edition of the grant

- SEO: A brief, plain-text summary combining the title and description for SEO purposes.
- bandi-descrizione (grant description): Details about the general objectives, purpose, and any specific project goals.
- candidati-idonei (eligible applicants): Information about eligible sectors, applicant types, and special requirements.
- interventi-ammissibili (eligible interventions): A list of allowed projects or interventions under the grant, including expense limitations.
- risorse-finanziarie (financial resources): Details about available funding options, such as grants or loans, and their minimum/maximum limits.
- scadenza (deadline): Key dates, including the start and end dates for applications.

## 3.2 Choice of LLM

The goal is to select an open-source model while preserving privacy by avoiding data sharing with third parties and maintaining full control over the data. Additionally, we aim to minimize resource usage; therefore, models with light-weights are considered. The selection prioritized models under 10 billion parameters, as this size aligned with the objectives of this study.

### LlaMa-3.1-8B

The Meta Llama 3.1 collection consists of multilingual large language models (LLMs) available in 8B, 70B, and 405B parameter sizes, designed for both pretrained and instruction-tuned generative tasks (text input/output). The instruction-tuned models are optimized for multilingual dialogue applications and outperform many open-source and proprietary chat models on key industry benchmarks.

Llama 3.1 is an auto-regressive language model built on an optimized transformer architecture. Its tuned versions leverage supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align with human preferences for helpfulness and safety.

#### Model Details

- Developer: Meta
- Model Type: auto-regressive language model
- Language Support: English, German, French, Italian, Portuguese, Hindi, Spanish, and Thai.
- Context window: 128K

- License: A custom commercial license, the Llama 3.1 Community License.
- Release Date: July 2024

### **Mixtral-7B**

Mixtral 7B is licensed under Apache 2.0. It outperforms Llama 2 70B on most benchmarks while offering 6x faster inference. As one of the strongest open-weight models with a permissive license, Mixtral provides excellent cost/performance trade-offs and matches or exceeds GPT-3.5 on many standard benchmarks.

#### **Model Details**

- Developer: Mistral AI
- Model Type: Decoder only
- Language Support: multilingual
- Context length: 32K
- License: Apache 2.0
- Release Date: September 2023

### **Falcon3-7B-Instruct**

This model achieves state-of-the-art performance (as of its release) in tasks involving reasoning, language understanding, instruction following, coding, and mathematics. The Falcon3-7B-Instruct variant supports four languages English, French, Spanish, and Portuguese and offers a context length of up to 32K tokens.

#### **Model Details**

- Developer: Technology Innovation Institute (TII)
- Model Type: Causal decoder-only
- Language Support: English (EN), French (FR), Spanish (ES), Portuguese (PT)
- Context length: 32K
- License: TII Falcon-LLM License 2.0
- Release Date: December 2024

In summary, Falcon 7B do not support the Italian language, making it less suitable for this project. On the other hand, Llama 3.1, is one of the newest models, offers the longest context window, a crucial feature for this use case. The ability to process long text from PDF files is essential for extracting information accurately, as it allows the model to better understand the context and nuances of the text.

### 3.3 Model customization

Techniques for customizing LLMs defined in previous chapter, in this section, they will be compared to select the best technique for the case study, focusing on the advantages and disadvantages of each technique. This comparison is essential for understanding when and how to apply these methods effectively. Let's dive into the comparison to uncover what makes each approach unique [1].

Fine-tuning a language model is resource-intensive and costly but worthy for achieving high accuracy in specific domains. It enables customization, improves relevance through specialized datasets, and adapts to niche topics or recent information. However, it demands significant computational power, advanced technical skills, and a well-curated dataset.

- Pros: Customizable for specific domains or styles. Improves accuracy and relevance with specialized datasets. Adapts to niche or updated information.
- Cons: High cost due to resource demands. Requires technical expertise in machine learning. Needs a substantial and curated dataset.

Retrieval-Augmented Generation (RAG) is ideal for providing up-to-date and detailed information by combining external data with the model's capabilities. It balances ease of use and customization but requires additional tools and computational resources. The choice of vector database, significantly impacts cost and performance.

- Pros: Provides dynamic, relevant information and balances simplicity with customization.
- Cons: Complex to implement, resource-intensive, and dependent on data quality.

Prompt Engineering is a simple and user-friendly approach that does not require technical expertise, making it accessible to most users. While effective for general topics and quick answers, it relies heavily on the model's initial training and may not provide the most specific or up-to-date information.

- Pros: Easy to use with minimal technical skills. Cost-effective with low computational requirements. Flexible, allowing quick adjustments to prompts.
- Cons: Inconsistent response quality depending on prompt phrasing. Limited customization compared to fine-tuning. Outputs depend on the model's existing knowledge, reducing effectiveness for specialized or current topics.

To sum up, the selection of a customization technique for LLMs in this thesis prioritizes cost-effectiveness and practicality. Among the available approaches, prompt engineering is the most suitable choice due to its minimal computational costs and ease of use. Since the information extraction task in this research does not require the model to learn highly specific or domain-focused tasks, the flexibility and efficiency of prompt engineering make it an ideal solution. This approach allows the application to achieve its objectives without the need for resource-intensive methods like fine-tuning or complex integrations such as Retrieval-Augmented Generation.



## 3.4 Pipeline definition

The Figure 3.1 illustrates the process designed for extracting information ("Rule Card") from PDF files.

- `lettore-documenti` (document reader): This component verifies the document format and proceeds with the workflow if the document is a PDF file. It is designed to accommodate future development, allowing support for other document types with customized workflows if required.
- `pdf-parser`: This component handles loading the document, parsing the PDF content, and dividing it into multiple chunks, if the provided PDF file is searchable.
- `estrattore-immagini-llm` (image extractor): For scanned PDF files, this component extracts the pages as images using Pillow and PyMuPDF, which relies on Poppler-utils for PDF manipulation. It checks the orientation of the images to optimize text extraction. Tesseract OCR, along with PyTesseract configured for the Italian language, is then used to extract text from the images. Finally, an LLM is utilized to correct any OCR errors, ensuring the output is clean text.
- `documenti-corretto` (correct document): This component evaluates each text chunk to determine whether it contains relevant information about grants and funds for "Rule Card". If the entire document is unrelated, it prevents unnecessary LLM calls and token usage, reducing costs. Additionally, chunks without relevant information are filtered out to streamline further processing.
- `estrattore` (extractor): This component leverages the function-calling capabilities of modern LLMs to extract structured information from text chunks.
- `verificatore-formato` (format verifier): This component performs a final check on the extracted information, consolidates the outputs from different chunks, and generates the final JSON-formatted result.
- `estrai-json` (JSON extractor): This component saves the final JSON file generated from the input PDF.

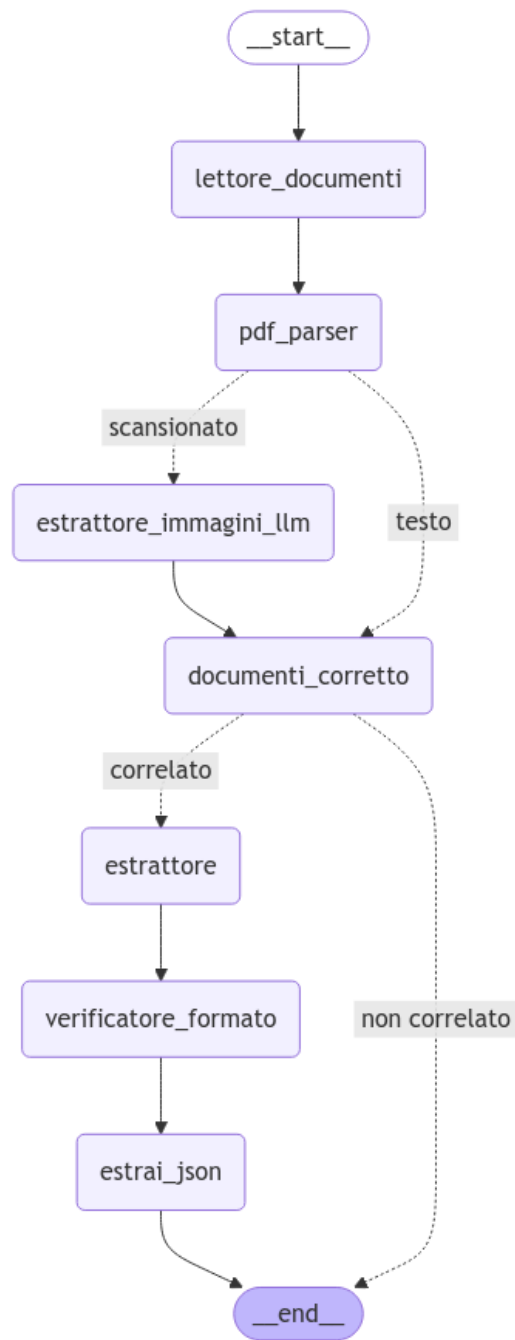


Figure 3.1. The pipeline

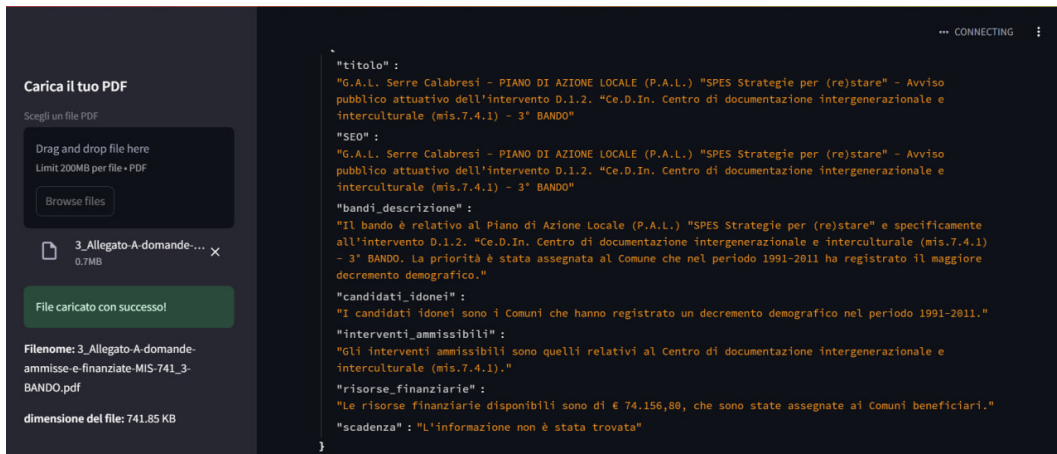


Figure 3.2. The user interface of the application

The application provides a user-friendly interface that allows users to interact with the model seamlessly. Through this interface, users can upload PDF files related to Italian grants and funds and receive the extracted "Rule Card" in a structured format. Acting as a wrapper around the pipeline, it offers a simple and intuitive way to interact with the model and obtain the desired results. The interface is developed using Streamlit. The Figure 3.2 shows a snapshot of the user interface.



# Chapter 4

## Results

This chapter presents analysis of the experimental results aimed at evaluating the model’s performance in extracting structured information from grant documents. It includes assessments of precision, recall, and F1 score, providing insights into the model’s effectiveness in identifying and organizing key grant-related data.

### 4.1 Benchmark


To evaluate the performance and accuracy of the proposed LLM-based application, a custom benchmark was designed, tailored to the specific domain of Italian regional grants and funds documents. The benchmark dataset is mined from official regional websites. However, not all files were relevant to the scope of the application, as some contained unrelated content or generic administrative information.

**Dataset Preparation** The preparation of the benchmark dataset involved a two-step process:

- **Document Filtering:** Each document was preprocessed to extract text and subsequently analyzed to determine its relevance to grants and funds. This was achieved using a relevance-checking module integrated into the pipeline. Only documents deemed relevant based on their textual content were retained for further processing.
- **Ground Truth Creation:** A subset of the filtered documents was manually annotated to create a ground truth dataset. This involved identifying and extracting key information such as title, deadline, financial resources, and other essential sections. The ground truth data served as a baseline for comparison against the system’s output.

### 4.2 Case study outcome

In this section, an example is provided to illustrate the input and output of the developed pipeline. While the dataset primarily consists of multi-page PDF documents, for



**G. A. L. Serre Calabresi**  
**Agenzia di Sviluppo Locale**

Allegato "A" alla delibera del  
 C.d.A. n. 235/7 del 24/03/2023

**PIANO DI AZIONE LOCALE (P.A.L.) "SPES Strategie per (re)stare"**  
 Avviso pubblico attuativo dell'intervento D.1.2. "Ce.D.In. Centro di documentazione intergenerazionale e interculturale (mis.7.4.1) - 3° BANDO

**GRADUATORIA DEFINITIVA DELLE DOMANDE AMMISSIBILI**

	Domanda	Cuaa	Beneficiario	CUP	INVESTIMENTO PREVISTO	INVESTIMENTO AMMESSO	CONTRIBUTO CONCESSO	PUNTEGGIO	NOTE
1	24250096385	90000510793	COMUNE DI CENADI	F74J22000490005	€ 24.744,21	€ 24.410,53	€ 24.410,53	35	
2	24250103041	85000330796	COMUNE DI OLIVADI	F84D22006690006	€ 24.862,50	€ 19.719,47	€ 19.719,47	30 (*)	
3	24250088226	326880796	COMUNE DI DAVOLI	E75E23000000004	€ 21.975,38	€ 18.497,15	€ 18.497,15	30	Di cui € 7.731,59 a valere sulle risorse aggiuntive 2021-2022 del PAL SPES
4	24250108594	164790792	COMUNE DI BADOLATO	I92F22000800005	€ 25.000,00	€ 11.529,65	€ 11.529,65	22	A valere sulle risorse aggiuntive 2021-2022 del PAL SPES
<b>TOTALE</b>					<b>€ 74.156,80</b>	<b>€ 74.156,80</b>			

Priorità: la priorità è stata assegnata al Comune che nel periodo 1991-2011 ha registrato il maggiore decremento demografico calcolato facendo riferimento ai dati ISTAT secondo la seguente formula: (pop.residente 2011-pop.residente 1991/pop.residente 2011) x 100 (Rif. Parag. 12 Disposizioni attuative)

Gruppo di Azione Locale G.A.L. Serre Calabresi s.c. a r.l.  
 C.da Foresta - 88064 Chiaravalle C.le (CZ) Tel. e fax 0967-998023  
 e-mail: [info@galserre calabresi.it](mailto:info@galserre calabresi.it) PEC: [galserre calabresi@pec.it](mailto:galserre calabresi@pec.it)  
 Sito internet: [www.serrecalabresi.it](http://www.serrecalabresi.it)

Figure 4.1. The PDF file example

simplicity and clarity, a single-page PDF file is used as an input example, as shown in Figure 4.1. This file is processed through the pipeline, demonstrating its capability to extract structured information.

The corresponding output, a "Rule Card," is presented in Figure 4.2. This output showcases the effectiveness of the pipeline in transforming unstructured data from the input document into a structured JSON format, highlighting its functionality and practical application.

## 4.3 Prompts

This section categorizes and identifies the prompts used throughout various stages of the pipeline, aligning them with different prompt engineering techniques.

### 4.3.1 Zero-shot learning

Based on the "Rule card" format, the zero-shot learning prompt would be something like Figure 4.3.

It is obvious that zero-shot learning in this case cannot be satisfying because the keywords are very abstract and it would be hard for the model to extract desired output without any further information.

```
"titolo" :  
"G.A.L. Serre Calabresi - PIANO DI AZIONE LOCALE (P.A.L.) "SPES Strategie per (re)stare" - Avviso  
pubblico attuativo dell'intervento D.1.2. "Ce.D.In. Centro di documentazione intergenerazionale e  
interculturale (mis.7.4.1) - 3° BANDO"  
"SEO" :  
"G.A.L. Serre Calabresi - PIANO DI AZIONE LOCALE (P.A.L.) "SPES Strategie per (re)stare" - Avviso  
pubblico attuativo dell'intervento D.1.2. "Ce.D.In. Centro di documentazione intergenerazionale e  
interculturale (mis.7.4.1) - 3° BANDO"  
"bandi_descrizione" :  
"Il bando è relativo al Piano di Azione Locale (P.A.L.) "SPES Strategie per (re)stare" e specificamente  
all'intervento D.1.2. "Ce.D.In. Centro di documentazione intergenerazionale e interculturale (mis.7.4.1)  
- 3° BANDO. La priorità è stata assegnata al Comune che nel periodo 1991-2011 ha registrato il maggiore  
decremento demografico."  
"candidati_idonei" :  
"I candidati idonei sono i Comuni che hanno registrato un decremento demografico nel periodo 1991-2011."  
"interventi_ammissibili" :  
"Gli interventi ammissibili sono quelli relativi al Centro di documentazione intergenerazionale e  
interculturale (mis.7.4.1)."  
"risorse_finanziarie" :  
"Le risorse finanziarie disponibili sono di € 74.156,80, che sono state assegnate ai Comuni beneficiari."  
"scadenza" : "L'informazione non è stata trovata"  
}
```

Figure 4.2. The extracted "Rule Card"

Estrarre le seguenti informazioni dal testo:

- titolo
- SEO
- bandi-descrizione
- candidati-idonei
- interventi-ammissibili
- risorse-finanziarie
- scadenza

Fornire l'output in formato JSON.

Figure 4.3. Zero-shot prompt for JSON extraction

### 4.3.2 Few-shot learning

#### OCR improvement

For the "estrattore-immagini-llm," which is responsible for extracting text from scanned documents and improving OCR errors, this technique has been used, as shown in Figure 4.4.

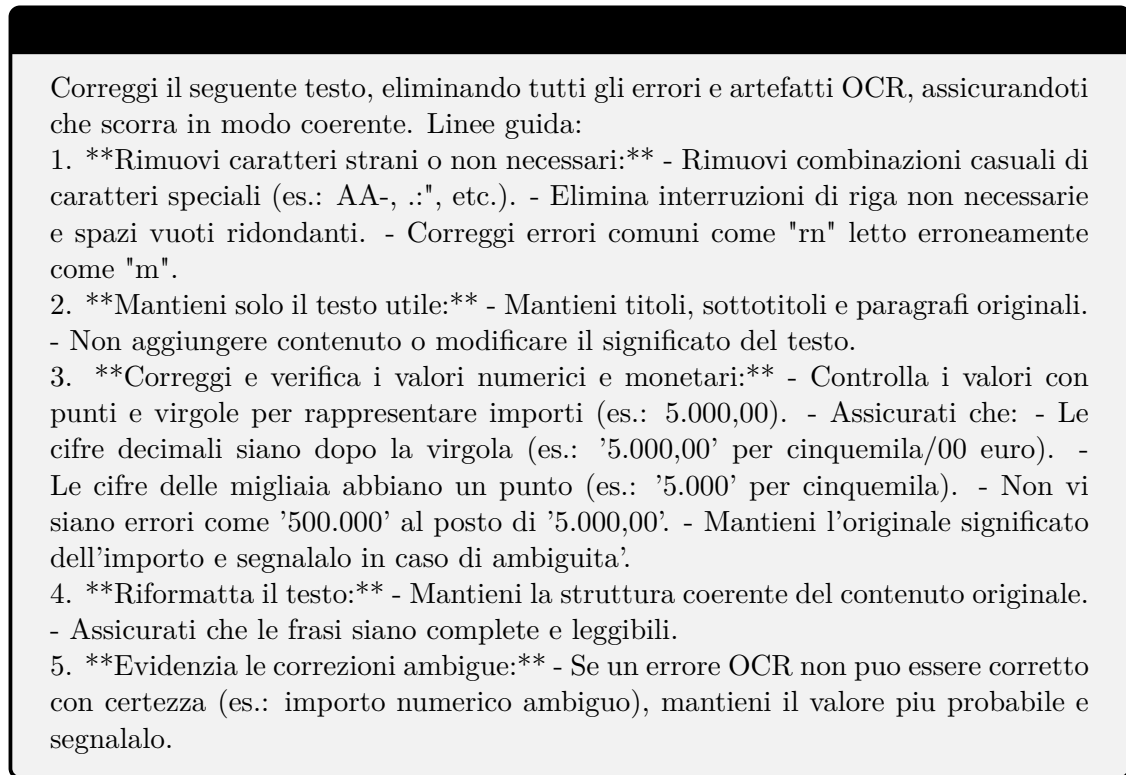


Figure 4.4. Few-shot prompt for OCR improvement

#### Filtering documents

For the "documenti-corretto," which is used to filter out documents or chunks of text that are not related to grants and funds, a few-shot learning prompt has been used, as shown in Figure 4.5.

### 4.3.3 Instruction-based learning

Because of the characteristics of input and output of this application, using few-shot prompting is not possible for the extraction part, so instead, an instruction of what should have been done will be passed to the LLM.

It provides detailed instructions on how to approach the task, including specific fields to extract and how to handle missing information. Unlike few-shot prompting, it does



Sei un assistente per il filtraggio di documenti. Verifica se nel testo sono presenti informazioni chiave riguardanti le "Regole per la creazione delle schede di sintesi dei bandi". Se le informazioni sono presenti, rispondi 'si'; altrimenti, rispondi 'no'. Le informazioni chiave includono:

**\*\*Regole per la scrittura del titolo:\*\*** Il titolo deve seguire un formato standardizzato, includendo elementi come l'ente erogatore, il nome del bando, l'azione o intervento specifico, il tipo di agevolazione e l'oggetto del bando.

**\*\*Esempi di titoli standardizzati:\*\***

- "Fondo Regionale per la Crescita Campania. PR FESR Azione 1.3.1. Finanziamento a fondo perduto e finanziamento agevolato a sostegno di investimenti per PMI e professionisti FRC II Edizione."
- "CCIAA di Bergamo. Finanziamento a fondo perduto in sostegno a investimenti delle imprese del settore apistico. Anno 2024."

**\*\*Descrizione del bando:\*\*** Obiettivi generali e specifici, inclusa la localizzazione geografica dell'intervento.

**\*\*Soggetti beneficiari:\*\*** Chi può partecipare e ricevere il sostegno, specificando le caratteristiche dei beneficiari ed eventuali requisiti specifici.

**\*\*Tipologie di interventi ammissibili:\*\*** Interventi e spese ammissibili, obblighi dei beneficiari, limiti di spesa e scadenze per la rendicontazione.

**\*\*Entità e forma dell'agevolazione:\*\*** Risorse finanziarie disponibili, tipo di agevolazione (es. contributo a fondo perduto, finanziamento a tasso agevolato), ripartizione delle risorse e limiti massimi e minimi.

**\*\*Scadenza:\*\*** Termine di apertura e chiusura per la presentazione delle domande.

Figure 4.5. Few-shot prompt for filtering texts

not include concrete examples of input-output pairs. Instead, it sets clear guidelines for how to process and format the extracted information. It emphasizes schema adherence (organizing data into a specific JSON format) and includes nuanced instructions, such as handling empty fields and performing mathematical calculations. This style of prompting is common for defining precise behavior in LLM tasks, especially when the goal is structured data extraction without relying on example demonstrations. It is sometimes used in workflows involving orchestration frameworks to fine-tune task specifications.

### Information extraction

Figure 4.6 shows the instruction-based prompt used to extract the desired information from chunks of text.

Instruction-based learning can be useful in this case. Although it is a long prompt, and it is going to fill a significant part of the model's context window, it would be helpful to guide the model to extract desired output.

Sei un assistente intelligente incaricato di analizzare il testo di un documento relativo alle bande di finanziamento e di estrarre le seguenti informazioni secondo lo schema specificato. Il tuo compito è identificare e compilare ciascuno dei campi richiesti basandoti sul contenuto fornito.

**Schema per il contenuto estratto (FormatoFinale):**

- **"titolo"**: stringa contenente il Fornitore di sovvenzioni, il nome della sovvenzione, Intervento/azione specifica, Tipologia di contributo (Contributo a fondo perduto, Finanziamento a tasso agevolato, Finanziamento a tasso zero, Garanzia, Credito d'imposta, Bonus fiscale).
- **"SEO"**: stringa contenente il titolo del bando concatenato con la descrizione in testo semplice (senza formattazione).
- **"bandi-descrizione"**: stringa contenente la descrizione del bando, inclusi obiettivi generali e specifici, e eventuali parametri geografici.
- **"candidati-idonei"**: stringa contenente le informazioni su chi può presentare domanda, incluse caratteristiche dei richiedenti, requisiti specifici, e vincoli finanziari.
- **"interventi-ammissibili"**: stringa contenente le informazioni sugli interventi ammissibili, obblighi per i beneficiari, spese ammissibili e finanziate, limiti di spesa, e scadenze per la rendicontazione.
- **"risorse-finanziarie"**: stringa contenente le informazioni sulle risorse finanziarie disponibili, tipo di assistenza, allocazione delle risorse tra i beneficiari, e limiti di allocazione. \*(Sei un esperto di matematica per calcolare le operazioni matematiche).\*
- **"scadenza"**: stringa contenente le scadenze per la presentazione delle domande.

**Istruzioni:**

1. Analizza attentamente il testo fornito.
2. Estrai le informazioni corrispondenti a ciascuno dei campi sopra elencati.
3. Organizza tutte le informazioni estratte in un oggetto JSON.

Figure 4.6. Instruction-based prompt for JSON extraction

### Format verification

This prompt works as a verification step and finalizes the extracted information, ensuring consistency and accuracy, as shown in Figure 4.7.

Sei un esperto nella convalida dell'estrazione di informazioni. I testi forniti per ogni parola chiave sono aggregati da diversi blocchi di un documento, il che significa che alcune sezioni potrebbero non contenere dettagli rilevanti o potrebbero includere informazioni irrilevanti. Il tuo compito è quello di esaminare attentamente ogni sezione per assicurarti che contenga le informazioni corrette e necessarie in base al titolo, rimuovere qualsiasi contenuto estraneo che non si adatta e riassumere quelli rilevanti.

**\*\*Per ogni sezione, assicurati che:\*\***

- Contenga informazioni chiave pertinenti alla sua categoria.
- Non contenga informazioni non correlate o ridondanti.
- Sia coerente con il contesto generale del documento.

**\*\*Ecco le sezioni da esaminare:\*\***

- **\*\*Titolo\*\***: dovrebbe contenere il fornitore della sovvenzione, il nome della sovvenzione, l'intervento specifico e il tipo di sovvenzione (ad esempio, "sovvenzione", "credito d'imposta").
- **\*\*SEO\*\***: dovrebbe essere costituito dal titolo e dalla descrizione della sovvenzione, formattati come testo normale senza punti elenco o caratteri speciali.
- **\*\*Bandi-descrizione\*\***: dovrebbe descrivere gli obiettivi generali e specifici della sovvenzione, inclusi eventuali criteri di ammissibilità geografica.
- **\*\*Candidati-idonei\*\***: dovrebbe specificare i candidati ammissibili, incluse le caratteristiche e i criteri di ammissibilità.
- **\*\*Interventi-ammissibili\*\***: dovrebbe descrivere gli interventi ammissibili, gli obblighi del beneficiario, le tempistiche, le spese minime e massime.
- **\*\*Risorse-finanziarie\*\***: dovrebbe specificare le risorse finanziarie disponibili, il tipo di intervento, la distribuzione tra i beneficiari e i limiti. \*(Per questa parte sei un esperto di matematica.)\*
- **\*\*Scadenza\*\***: dovrebbe indicare le scadenze per l'apertura e la chiusura delle domande. Se non sei sicuro, scrivi "Le informazioni non sono state trovate".

**\*\*Rivedi ogni sezione e restituisci il contenuto essenziale e pertinente in dettaglio:\*\***

- **\*\*Titolo\*\***:
- **\*\*SEO\*\***:
- **\*\*Bandi-descrizione\*\***:
- **\*\*Candidati-idonei\*\***:
- **\*\*Interventi-ammissibili\*\***:
- **\*\*Risorse-finanziarie\*\***:
- **\*\*Scadenza\*\***: scadenza

Figure 4.7. Instruction-based prompt for verification of the extracted information

## 4.4 Evaluation approach

This section describes the evaluation methodology used to assess the performance of the LLM-based application for information extraction tasks. Evaluating the quality of the model’s output proved to be one of the most challenging aspects of the research due to the generative nature of LLMs [15].

Evaluation approaches:

- **Automated Evaluation:** This method uses predefined metrics and algorithms to assess the performance of LLM applications without human intervention. It relies on computational techniques to compare the LLM’s output with a reference or specific criteria.
- **Human Evaluation:** This approach involves human judgment to assess the output generated by LLMs, considering aspects that automated metrics cannot effectively capture.
- **LLM-Based Evaluation:** A newer method where one LLM evaluates the output of another. This approach utilizes the language understanding and generation capabilities of LLMs, offering a more automated and potentially scalable alternative to human evaluation.

The study by Bhashithe Abeysinghe and Ruhan Circi (2024) emphasizes that each evaluation approach has its strengths and limitations. Automated evaluation is consistent and repeatable but less reliable, human evaluation is highly accurate but costly and subjective, while LLM-based evaluation shows promise but requires further research to validate its reliability [2].

### 4.4.1 Performance metrics

There are various established methods for calculating metric scores, some rely on neural networks, including embedding models and LLMs, while others are purely statistical. These metric scorers are illustrated in the Figure 4.8. Statistical methods often perform poorly in scenarios requiring reasoning, making them too inaccurate for many LLM evaluation criteria. Scorers based solely on NLP models tend to be more accurate but can be less reliable due to their probabilistic nature [24].

A combination of statistical and model-based scorers leads to metrics that utilize embedding models or LLMs to evaluate the performance of LLM outputs.

In this thesis, the BERTScore metric is used to evaluate precision, recall, and F1 scores. BERTScore relies on pre-trained language models like BERT and calculates cosine similarity between the contextual embeddings of words in the reference and generated texts. These similarities are aggregated to produce a final evaluation score.

With BERTScore, precision, recall, and F1-score values were computed by embedding candidate and reference texts and measuring their cosine similarity. As this approach leverages contextual understanding, it is considered more appropriate for judging the semantic fidelity of extracted fields.

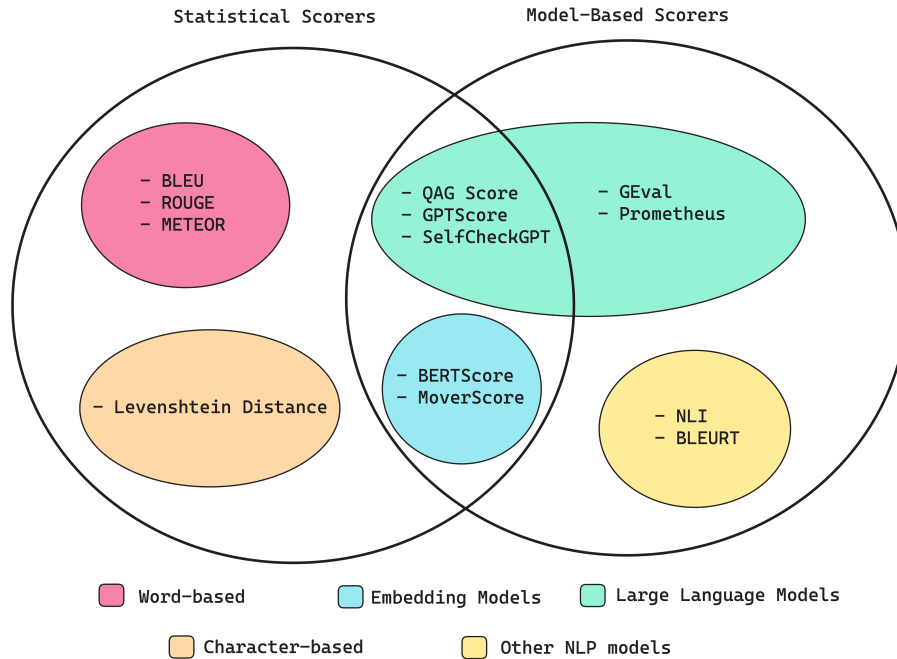


Figure 4.8. Types of metric scorers [25]

### Precision, Recall, F1score

Precision measures how much of the extracted content is relevant and correct. High Precision means most of the extracted tokens are correct, with few irrelevant or extra tokens.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

- True Positives (TP): The number of correctly extracted tokens (or words) that match the ground truth.
- False Positives (FP): The number of incorrectly extracted tokens that are not part of the ground truth but are included in the extraction.

Recall measures how much of the relevant content from the ground truth was correctly extracted. High Recall means the system captures most of the relevant tokens, even if it includes some extra content.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

- True Positives (TP): The number of correctly extracted tokens (or words) that match the ground truth.

- False Negatives (FN): The number of relevant tokens in the ground truth that were not extracted.

Finally, F1score is used to normalize precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

#### 4.4.2 Evaluation

In order to evaluate the performance of the proposed LLM-based extraction pipeline, 10 documents of the dataset related to grants and funding announcements was compiled. These documents were chosen to represent a range of complexities, including variations in length, organizational structure, searchability of PDFs, and level of detail. It was because of the time-consuming and difficulties of creating ground truth manually. The ground-truth structured data were annotated manually ensuring that a reliable reference could be established for comparing automated extractions.

To benchmark the proposed approach, a direct prompt-based extraction using ChatGPT (GPT-4) and NotebookLM (Gemini pro) was employed. The baseline system was provided with instructions similar to those given to the proposed pipeline, although it did not incorporate the tailored chunking or filtering modules. In this manner, the impact of these design choices on extraction quality was evaluated by comparing results produced by the pipeline against those generated by those platforms.

In Table 4.1, the average BERTScore results for the proposed pipeline are presented. Although the absolute metric values appear moderate rather than high, it should be noted that extracting structured data from complex, domain-specific documents remains a challenging task. Perfect alignment between automatically extracted and annotated information is difficult to achieve. Nonetheless, the proposed pipeline consistently outperformed Gemini Pro in nearly all documents and demonstrated strong compatibility with GPT-4 outputs.

Table 4.1. Performance of LLMs on Documents using BERTScore

Documents	LLM-applications	Precision	Recall	F1score
Document 1	<b>Pipeline</b>	<b>0.75</b>	<b>0.72</b>	<b>0.73</b>
	GPT-4	0.78	0.76	0.77
	Gemini pro	0.68	0.66	0.66
Document 2	<b>Pipeline</b>	<b>0.70</b>	<b>0.68</b>	<b>0.68</b>
	GPT-4	0.66	0.66	0.66
	Gemini pro	0.41	0.48	0.44
Document 3	<b>Pipeline</b>	<b>0.78</b>	<b>0.75</b>	<b>0.76</b>
	GPT-4	0.77	0.69	0.73
	Gemini pro	0.76	0.73	0.74
Document 4	<b>Pipeline</b>	<b>0.66</b>	<b>0.72</b>	<b>0.68</b>
	GPT-4	0.70	0.75	0.72
	Gemini pro	0.65	0.74	0.69
Document 5	<b>Pipeline</b>	<b>0.75</b>	<b>0.61</b>	<b>0.67</b>
	GPT-4	0.78	0.77	0.77
	Gemini pro	0.41	0.41	0.41
Document 6	<b>Pipeline</b>	<b>0.76</b>	<b>0.71</b>	<b>0.73</b>
	GPT-4	0.75	0.75	0.75
	Gemini pro	0.47	0.49	0.47
Document 7	<b>Pipeline</b>	<b>0.64</b>	<b>0.64</b>	<b>0.64</b>
	GPT-4	0.70	0.71	0.71
	Gemini pro	0.43	0.45	0.44
Document 8	<b>Pipeline</b>	<b>0.75</b>	<b>0.63</b>	<b>0.68</b>
	GPT-4	0.82	0.78	0.80
	Gemini pro	0.48	0.50	0.49
Document 9	<b>Pipeline</b>	<b>0.76</b>	<b>0.75</b>	<b>0.75</b>
	GPT-4	0.70	0.71	0.71
	Gemini pro	0.47	0.53	0.50
Document 10	<b>Pipeline</b>	<b>0.73</b>	<b>0.67</b>	<b>0.70</b>
	GPT-4	0.81	0.81	0.81
	Gemini pro	0.48	0.50	0.49

## 4.5 Error analysis

This section highlights examples where the model did not successfully extract the correct information. While the model generally performs well in extracting accurate data, understanding the reasons behind occasional failures remains challenging. Below are additional instances where the model encountered difficulties in extracting the correct information. To further investigate these issues, Table 4.2 presents the performance metrics for each keyword of the "Rule Card" across different documents, providing insights into where the pipeline struggled to extract specific information accurately. This analysis helps identify recurring patterns of errors and potential areas for improvement in the extraction process.

Table 4.2. Performance metrics for various attributes of the documents

Documents	Metrics	Title	SEO	Grant description	Eligible applicants	Eligible interventions	Financial resource	Deadline
Document 1	Precision	0.85	0.68	0.67	0.71	0.69	0.76	0.90
	Recall	0.60	0.68	0.73	0.76	0.72	0.73	0.84
	F1score	0.70	0.68	0.70	0.73	0.70	0.75	0.87
Document 2	Precision	0.80	0.71	0.78	0.63	0.61	0.65	0.73
	Recall	0.63	0.68	0.76	0.79	0.64	0.61	0.64
	F1score	0.71	0.69	0.77	0.70	0.62	0.63	0.68
Document 3	Precision	0.89	0.88	0.77	0.75	0.72	0.71	0.75
	Recall	0.86	0.74	0.69	0.66	0.70	0.69	0.88
	F1score	0.88	0.81	0.73	0.70	0.71	0.70	0.81
Document 4	Precision	0.67	0.65	0.73	0.71	0.70	0.66	0.51
	Recall	0.69	0.65	0.73	0.73	0.68	0.70	0.87
	F1score	0.68	0.65	0.73	0.72	0.69	0.68	0.65
Document 5	Precision	0.84	0.84	0.74	0.73	0.79	0.60	0.67
	Recall	0.57	0.51	0.69	0.64	0.66	0.61	0.60
	F1score	0.68	0.63	0.71	0.68	0.72	0.61	0.64
Document 6	Precision	0.87	0.87	0.76	0.67	0.75	0.64	0.74
	Recall	0.77	0.70	0.73	0.65	0.72	0.68	0.70
	F1score	0.82	0.78	0.74	0.66	0.74	0.66	0.72
Document 7	Precision	0.43	0.65	0.66	0.61	0.64	0.77	0.70
	Recall	0.49	0.61	0.66	0.62	0.64	0.69	0.73
	F1score	0.46	0.63	0.66	0.62	0.64	0.73	0.72
Document 8	Precision	0.78	0.79	0.68	0.82	0.76	0.58	0.85
	Recall	0.64	0.68	0.79	0.69	0.68	0.50	0.47
	F1score	0.70	0.73	0.73	0.75	0.72	0.54	0.61
Document 9	Precision	0.89	0.84	0.77	0.63	0.74	0.66	0.77
	Recall	0.72	0.75	0.76	0.68	0.75	0.73	0.89
	F1score	0.80	0.79	0.76	0.65	0.75	0.70	0.83
Document 10	Precision	0.72	0.65	0.73	0.78	0.66	0.80	0.79
	Recall	0.55	0.57	0.69	0.63	0.66	0.71	0.87
	F1score	0.62	0.61	0.71	0.70	0.66	0.76	0.83



### 4.5.1 Metrics visualization

In this section, the performance metrics for each document are visualized and analyzed in detail. The metrics specifically evaluate the output of the designed pipeline and are subsequently compared to the ground truth to highlight critical points. This comparison provides insights into the strengths and limitations of the pipeline, offering a deeper understanding of its performance in real-world scenarios.

#### Document 1

The Figure 4.9 shows the metrics for the first document.

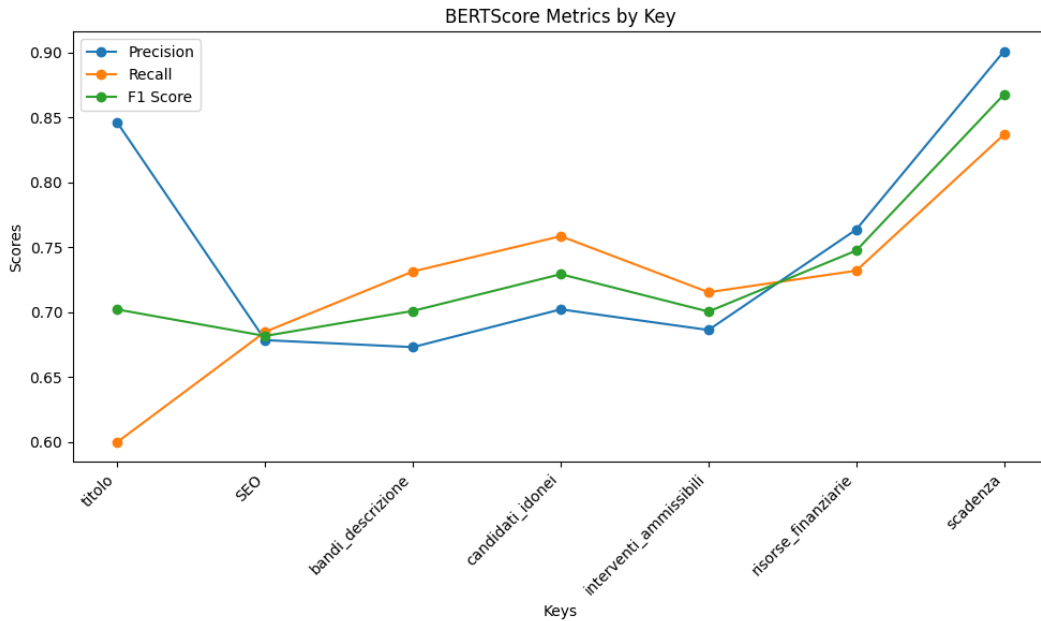


Figure 4.9. Document 1 metrics.

For the most of the attributes, the Precision, Recall, and F1 Scores are more balanced, indicating that the pipeline is performing consistently in extracting these parts. For "scadenza", all three metrics are higher, suggesting that the pipeline is very effective at extracting this information.

For "titolo" Precision is very high which indicates that the extracted "titolo" information is mostly accurate and matches the reference closely. However, recall is much lower, it suggests that the extracted content might be missing some relevant parts of the "titolo" present in the reference. This gap indicates that while the system outputs very precise content for the "titolo," it is not exhaustive, potentially due to strict matching criteria or incomplete extraction.

The texts of title in ground truth and extracted title are shown respectively on the Figure 4.10 and the Figure 4.11.

**\*\*"titolo" ground truth of document 1:\*\***  
 "G.A.L. Serre Calabresi, Avviso Pubblico Progetti di Cooperazione Territoriale 2 bando, Intervento A.1.1 'Le colture della storia'. Tipologia di contributo: Contributo a fondo perduto."

Figure 4.10. Document-1 "titolo" ground truth

**\*\*Extracted "titolo" of document 1:\*\***  
 "G.A.L. Serre Calabresi, Avviso Pubblico Progetti di Cooperazione Territoriale 2 bando"

Figure 4.11. Document-1 "titolo" extracted data

## Document 2

The metrics for Document 2 has shown on the Figure 4.12.

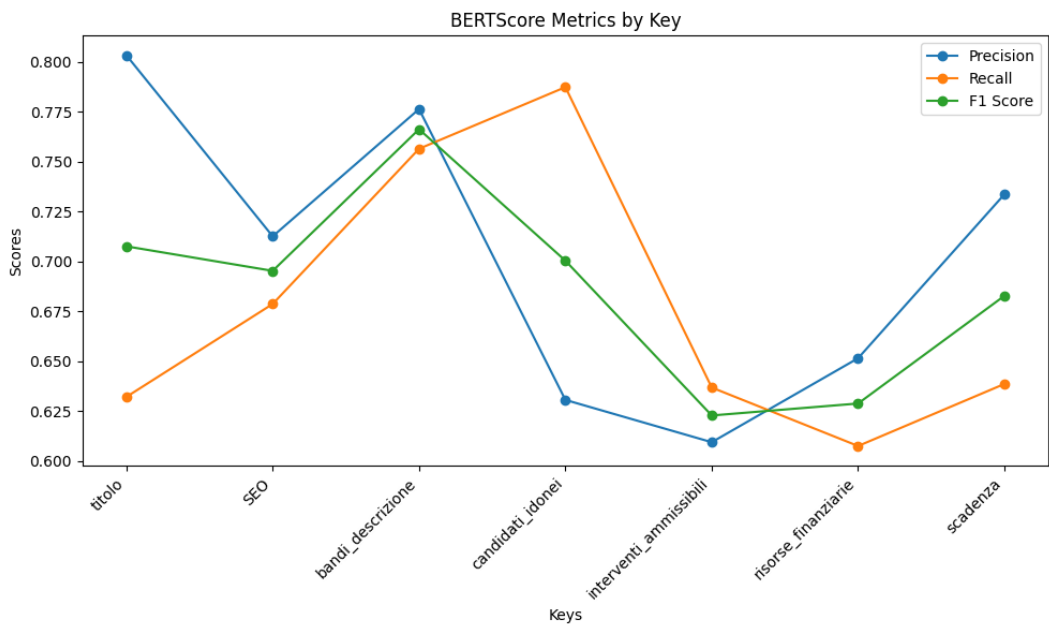


Figure 4.12. Document 2 metrics.

It is shown that for "interventi-ammissibili" (eligible interventions) there is the least precision. It can be seen in the comparison of extracted information and the ground truth information that the text is different but both point to a school-work projects but the information about the form of project presentation which is by video is missing in the extracted information by the pipeline. The texts of eligible interventions in ground truth and extracted eligible interventions are shown respectively on the Figure 4.13 and the Figure 4.14.

```
**"interventi-ammissibili" ground truth of document 2:**  
"Realizzazione di progetti di alternanza scuola-lavoro presentati sotto forma di  
video accompagnati da una descrizione sintetica. Sono ammesse anche collabo-  
razioni con aziende ed enti esterni per arricchire l'esperienza formativa."
```

Figure 4.13. "Interventi-ammissibili" ground truth of document 2

```
**Extracted "interventi-ammissibili" of document 2:**  
"Il Premio e' suddiviso in due categorie distinte per tipologia di Istituto scolastico  
partecipante: Licei e Istituti tecnici e professionali."
```

Figure 4.14. "Interventi-ammissibili" extracted data of document 2

### Document 3

The Figure 4.15 shows the metrics for the third document.

For document 3 the least metrics belong to the "risorse-finanziarie" (financial resource), so this part of the extracted information will be analyzed. The comparison shows that both ground truth and extracted information show the same amount of finance (74,156.80 euro) but the period for additional resources is not defined in the extracted information. The texts of financial resource in ground truth and extracted financial resource are shown respectively on the Figure 4.16 and the Figure 4.17.

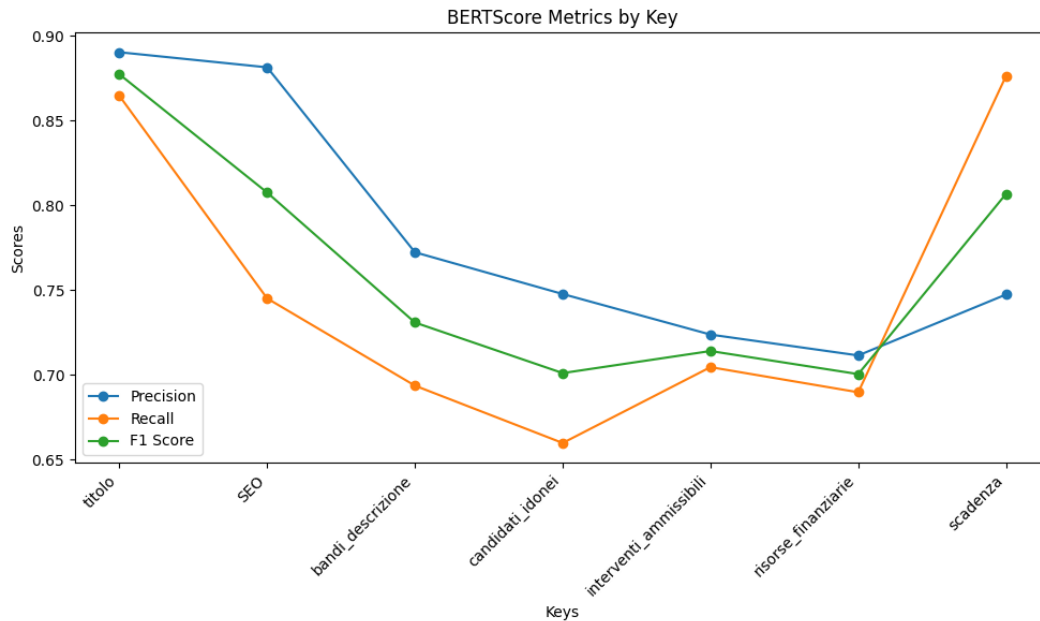


Figure 4.15. Document 3 metrics.

**\*\*"Risorse-finanziarie" ground truth of document 3:\*\***

"Le risorse totali disponibili ammontano a 74.156,80, suddivise tra i beneficiari con priorit  in base al decremento demografico. Il contributo concesso varia per progetto, in base alle risorse richieste e ammissibili. Sono incluse risorse aggiuntive per il periodo 2021-2022."

Figure 4.16. "Risorse-finanziarie" ground truth of document 3

**\*\*Extracted "risorse-finanziarie" of document 3:\*\***

"Le risorse finanziarie disponibili sono di 74.156,80, destinati a sostenere la creazione di un centro di documentazione intergenerazionale e interculturale. La tipologia di intervento   un contributo a fondo perduto."

Figure 4.17. "Risorse-finanziarie" extracted data of document 3

## Document 4

The Figure 4.18 shows the metrics for the fourth document.

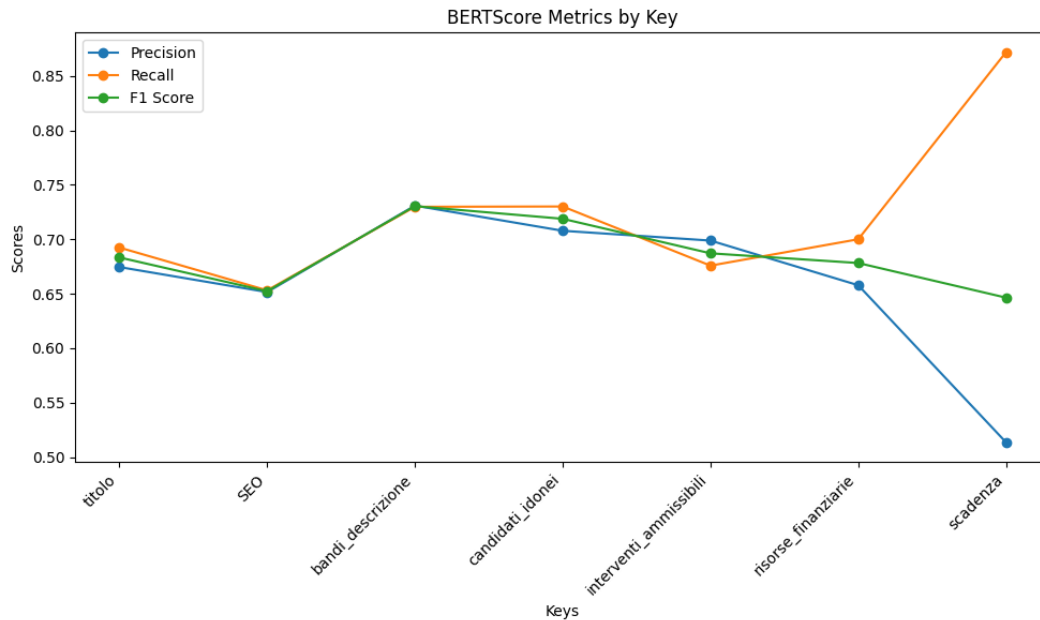


Figure 4.18. Document 4 metrics.

The metrics shows that "scadenza" (deadline) has the least precision, but by analyzing this part of the extracted information, it can be conclude that the date and time of the deadline is correct, and the only difference relies on the redundant text.

**\*\*"Scadenza" ground truth of document 4:\*\* "18/11/2019 - ore 12.00"**

Figure 4.19. "Scadenza" ground truth of document 4

**\*\*Extracted "scadenza" of document 4:\*\* "La data scadenza per la presentazione delle candidature e' il 18/11/2019 - ore 12.00."**

Figure 4.20. "Scadenza" extracted data of document 4

## Document 5

The Figure 4.21 shows the metrics for the fifth document.

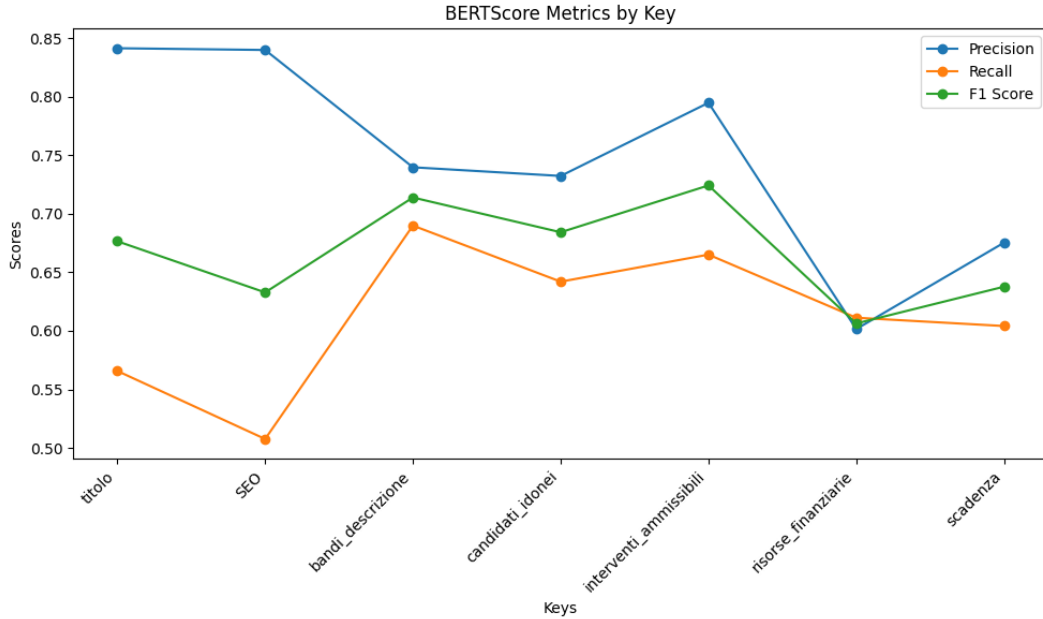


Figure 4.21. Document 5 metrics.

It can be observed that the least metrics belong to "risorse-finanziarie" that will be analyzed. In this case, the amount of finance does not exist on the document and that is why the pipeline hallucinated and extracted [amount] euro. The texts of financial resource in ground truth and extracted financial resource are shown respectively on the Figure 4.22 and the Figure 4.23.

\*\*"Risorse-finanziarie" ground truth of document 5:\*\*  
 "Risorse disponibili a valere sul PSR Calabria 2014-2022, con contributi a fondo perduto. Il contributo pubblico e' vincolato ai criteri di selezione e prioritari definiti."

Figure 4.22. "Risorse-finanziarie" ground truth of document 5

**\*\*Extracted "risorse-finanziarie" of document 5:\*\***  
 "Il finanziamento e' di [importo] euro e sara' erogato a carico dei Fondi PSR."

Figure 4.23. "Risorse-finanziarie" extracted data of document 5

## Document 6

The Figure 4.24 shows the metrics for the sixth document.

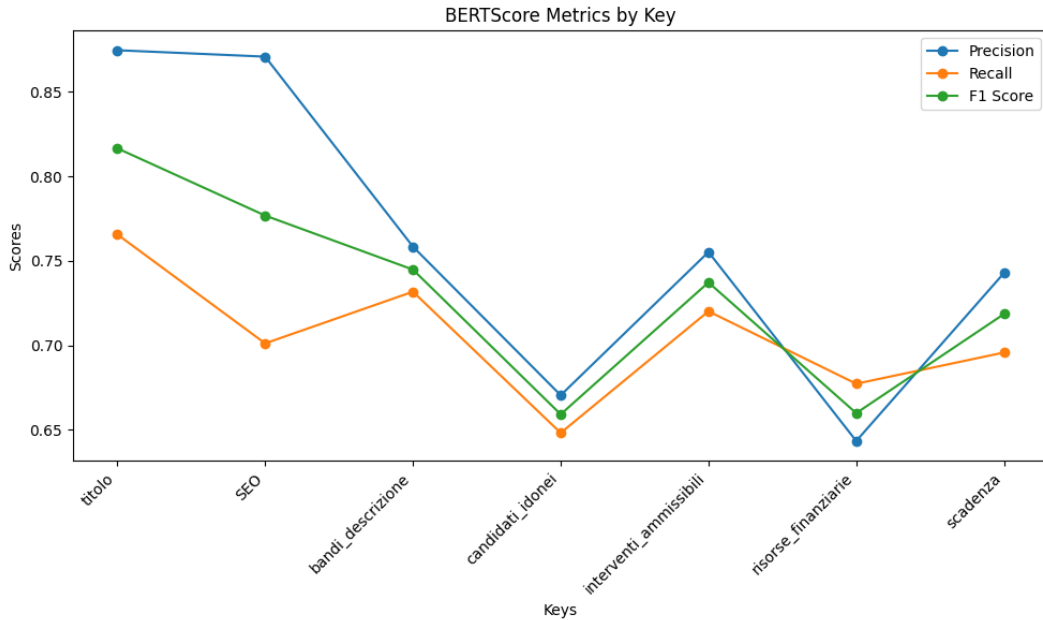


Figure 4.24. Document 6 metrics.

For "candidati-idonei" (eligible candidates) the problem is that the extracted information is not detailed. In other words, both extracted and ground truth information for this part, refers to agricultural companies operating in the irrigation and withdrawal water resources, but in the ground truth it is pointed that they should be able to implement specific system. The texts of eligible candidates in ground truth and extracted eligible candidates are shown respectively on the Figure 4.27 and the Figure 4.28.

**\*\*"Candidati-idonei" ground truth of document 6:\*\***  
"Consorti di bonifica con autorizzazione valida per il prelievo di risorse idriche, che abbiano rispettato le normative regionali e nazionali e che siano in grado di implementare il sistema SIGRIAN."

Figure 4.25. "Candidati-idonei" ground truth of document 6

**\*\*Extracted "candidati-idonei" of document 6:\*\***  
"Le imprese agricole e le aziende che operano nel settore dell'irrigazione possono presentare domanda di sostegno."

Figure 4.26. "Candidati-idonei" extracted data of document 6

Also, for "risorse-finanziarie" (financial resource), the desired information does not exist on the document and the pipeline hallucinated and the extracted information is not correct. The texts of financial resource in ground truth and extracted financial resource are shown respectively on the Figure 4.25 and the Figure 4.26.

**\*\*"Risorse-finanziarie" ground truth of document 6:\*\***  
"Contributo a fondo perduto fino al 100% delle spese ammissibili, incluse opere di costruzione e acquisizione di beni strumentali. Le risorse sono allocate in base ai criteri di selezione del bando."

Figure 4.27. "Risorse-finanziarie" ground truth of document 6

**\*\*Extracted "risorse-finanziarie" of document 6:\*\***  
"Le risorse finanziarie disponibili sono di 100.000,00 euro, con una ripartizione del 70% per l'acquisto di misuratori e del 30% per l'implementazione di tecnologie."

Figure 4.28. "Risorse-finanziarie" extracted data of document 6



## Document 7

The Figure 4.29 shows the metrics for the seventh document.

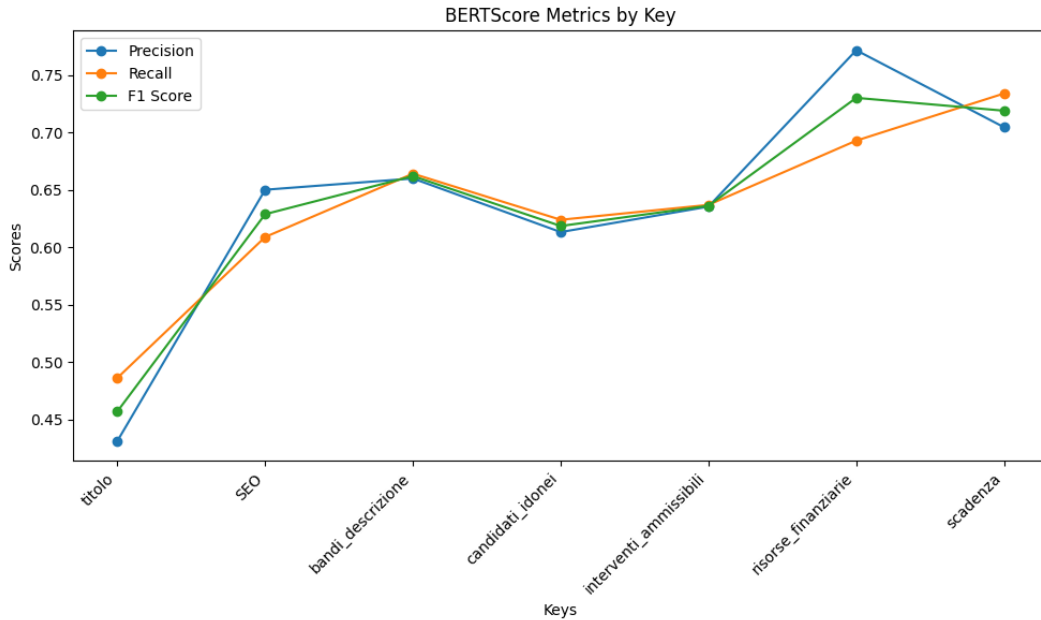


Figure 4.29. Document 7 metrics.

For this document as "titolo" (title) has the least metrics, this part of the information is going to be analyzed. It can be seen that the problem here is rooted in the previous steps of parsing text from pdf documents which leads to an unsuitable format of letters however there is no significant semantic differences. Also, the type of contribution is missed. The texts of title in ground truth and extracted title are shown respectively on the Figure 4.30 and the Figure 4.31.

**\*\*Titolo" ground truth of document 7:\*\***  
 "PSR Calabria 2014-2022, Piano degli Interventi, Misura 4.3.2 'Gestione Risorse Irrigue', Tipologia di contributo: Contributo a fondo perduto."

Figure 4.30. "Titolo" ground truth of document 7

**\*\*Extracted "Titolo" of document 7:\*\***  
**"P R O G R A M M A D I S V I L U P P O R U R A L E D E L L A R E G I O N E  
 C A L A B R I A 2 0 1 4 - 2 0 2 2 S o s t e g n o a g l i i n v e s t i m e n t i p e r l o s v i l u p p o r u r a l e  
 i n C a l a b r i a ."**

Figure 4.31. "Titolo" extracted data of document 7

### Document 8

The Figure 4.32 shows the metrics for the eighth document.

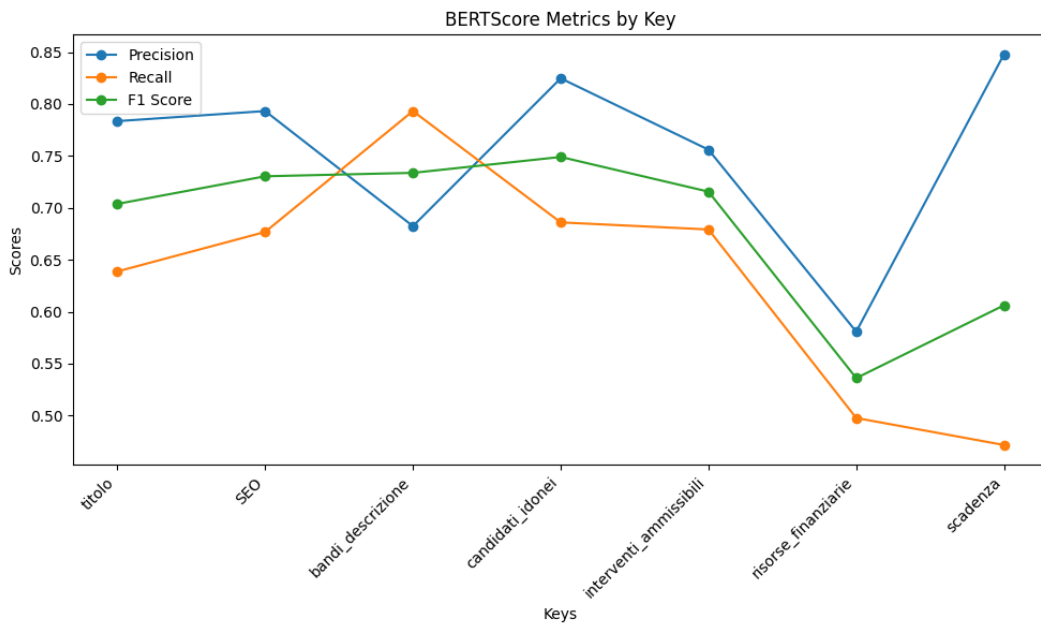


Figure 4.32. Document 8 metrics.

For "scadenza" (deadline) a huge gap can be found between metrics but the error analysis shows that the pipeline has extracted this information correctly and this gap is only because of differences between texts. The texts of deadline in ground truth and extracted deadline are shown respectively on the Figure 4.33 and the Figure 4.34.

**\*\*"Scadenza" ground truth of document 8:\*\***  
 "Il progetto si e' svolto dal 08/11/2021 al 10/12/2021, con termine dell'attività formativa entro tale periodo."

Figure 4.33. "Scadenza" ground truth of document 8

**\*\*Extracted "Scadenza" of document 8:\*\***  
 "Dal 08/11/2021 al 10/12/2021."

Figure 4.34. "Scadenza" extracted data of document 8

## Document 9

The Figure 4.35 shows the metrics for the ninth document.

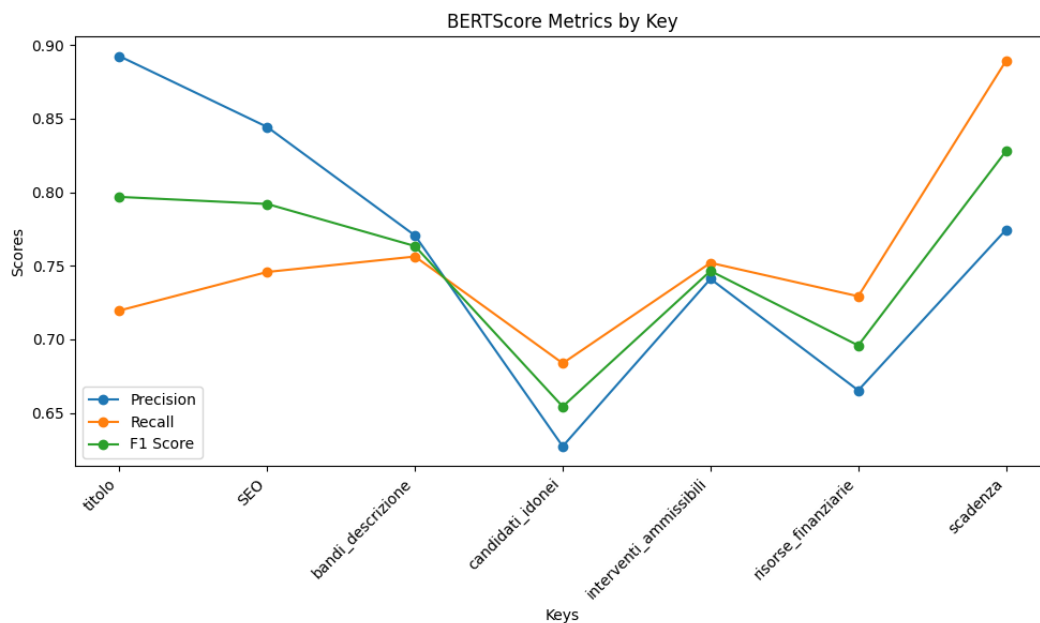


Figure 4.35. Document 9 metrics.

For title "titolo" it is shown that however the name of the grant is extracted correctly, but the contribution type is missed. The texts of title in ground truth and extracted title are shown respectively on the Figure 4.36 and the Figure 4.37.

**\*\*"Titolo" ground truth of document 9:\*\***

"Piano di Azione Locale (PAL SPES), Intervento C.2.1 'Acqua e pietra', Recupero delle infrastrutture storiche e caratterizzazione del paesaggio rurale, Tipologia di contributo: Contributo a fondo perduto."

Figure 4.36. "Titolo" ground truth of document 9

**\*\*Extracted "Titolo" of document 9:\*\***

"Intervento C.2.1 'Acqua e pietra'. Recupero delle infrastrutture storiche caratterizzanti il paesaggio rurale."

Figure 4.37. "Titolo" extracted data of document 9

## Document 10

The Figure 4.38 shows the metrics for the last document.

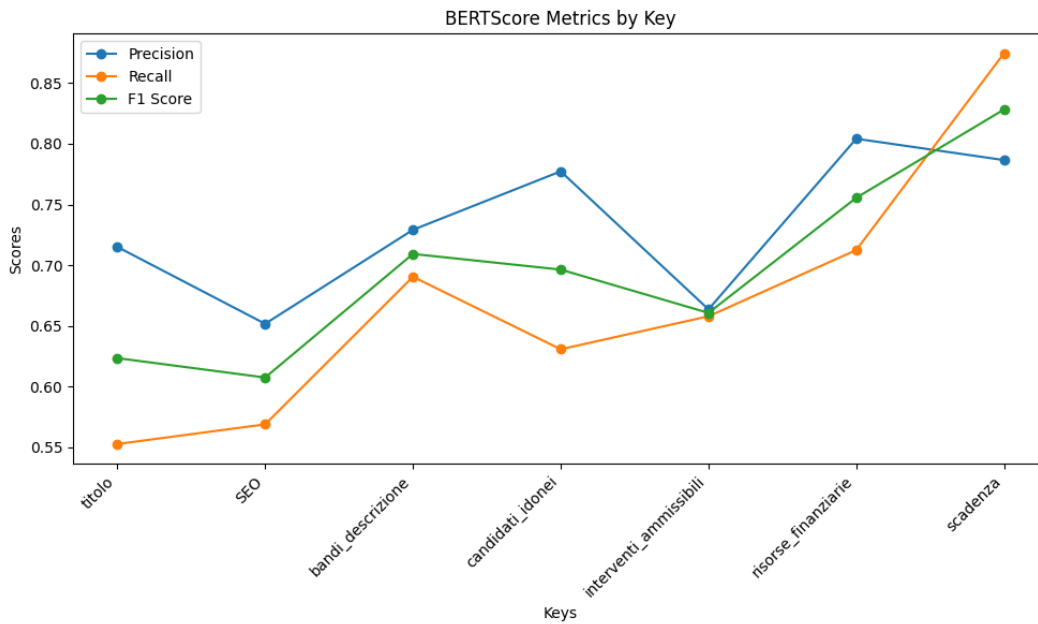


Figure 4.38. Document 10 metrics.

For title ("titolo") it can be seen that again the type of contribution is missed, but the title itself is semantically supported in the extracted information. The texts of title in ground truth and extracted title are shown respectively on the Figure 4.39 and the Figure 4.40.

```

**"Titolo" ground truth:**
"GAL Serre Calabresi, Misura 4.1.1 'Investimenti nelle aziende agricole', Intervento
A.1.1/d 'Le colture della storia. Sostegno alle produzioni agricole e zootecniche del
territorio', Tipologia di contributo: Contributo a fondo perduto."

```

Figure 4.39. "Titolo" ground truth of document 10

```
**Extracted "Titolo":**  
"Sostegno agli investimenti per le aziende agricole dell'area del Gal Serre Calabresi."
```

Figure 4.40. "Titolo" extracted data of document 10

Also, for eligible candidates ("candidati-idonei") it can be seen that although sentences are different but semantically both refers to the right groups of candidates including farmers and associations of farmers. The texts of eligible candidates in ground truth and extracted eligible candidates are shown respectively on the Figure 4.41 and the Figure 4.42.

```
**"Candidati-idonei" ground truth:**  
"Agricoltori e associazioni di agricoltori (imprese agricole), regolarmente costituite e iscritte alla Camera di Commercio, in possesso della qualifica di imprenditore agricolo professionale (IAP) o coltivatore diretto (CD)."
```

Figure 4.41. "Candidati-idonei" ground truth of document 10

```
**Extracted "Candidati-idonei":**  
"Agricoltori, associazioni di agricoltori secondo le forme previste e stabilite dalla legge."
```

Figure 4.42. "Candidati-idonei" extracted data of document 10

# Chapter 5

## Discussion

This chapter presents discussions and analyses, beginning with an evaluation of the results, followed by a review of the methodologies used. Then, the challenges faced during the research and the lessons learned will be discussed. Next, it suggests ideas for future work, showing how the study could be improved or expanded. Finally, it reviews how the research objectives were met, linking the results to the main goals and setting the stage for future studies.

### 5.1 Analysis of results

Building on the findings described in the Results chapter, this section examines the strengths and weaknesses of the proposed LLM-based extraction pipeline. The discussion reflects how the pipeline’s design, implementation, and evaluation outcomes align with the research objectives, while highlighting both its notable achievements and areas needing improvement.

#### Strengths

- **Modular Pipeline Design.** The systematic approach-consisting of text extraction, OCR correction, chunk filtering, extraction via instruction-based prompting, and final verification, allows each stage to be independently refined or replaced. This modularity enables targeted improvements without changing the entire system.
- **Domain-Specific Customization** By tailoring prompts and filters specifically for Italian grants and funds, the pipeline effectively narrows its focus to relevant content, thereby reducing noise and improving the quality of extracted information. This emphasis on domain-specific chunking and filtering likely contributed to the more accurate extractions compared to generic approaches.
- **Efficiency and Cost-Effectiveness** The decision to use light-weight LLMs and prompt engineering over resource-intensive and more expensive techniques like full fine-tuning or Retrieval-Augmented Generation reflects a practical balance between customization and computational overhead. This choice keeps customization and

infrastructure costs lower, an important consideration for small and medium-sized enterprises (SMEs).

- **Flexibility for Various Document Types.** The architecture accommodates both searchable and scanned PDFs. The OCR-based module-supported by Tesseract, PyPDF, and other tools-demonstrates the pipeline's adaptability to real-world scenarios where document formats and quality can vary widely.
- **Improved Performance through Filtering.** The addition of a filtering step for irrelevant document chunks reduced the likelihood of extra LLM calls. This not only saves computational resources but also improves the precision of extracted data by focusing on text segments most likely to contain relevant grant information which was shown by evaluating the results.

### Weaknesses

- **Dependence on OCR Quality.** While the integration of Tesseract and PyPDF addresses scanned documents, the accuracy of the overall pipeline in such cases directly depends on the clarity of the source files. Substandard scans or image artifacts can degrade OCR performance, leading to incomplete or erroneous extractions.
- **Vulnerability to Model Hallucination.** Despite the pipeline's structured prompts and modular approach, the LLM can still generate fabricated or "hallucinated" details, especially for fields where the document lacks explicit information. Such errors were evident in "risorse-finanziarie" field in one of the documents.
- **Prompt Specificity and Coverage.** When prompts attempt to capture multiple types of information at one keyword, the model may miss details that are insufficiently aligned with the target keywords. In such cases, certain key elements can be overlooked, leading to incomplete extractions.
- **Context Window Constraints.** Although Llama 3.1 supports a lengthy context window, lengthy prompts and large documents can still approach or exceed these limits, risking truncation or reduced performance. This necessitates careful prompt design and chunking strategies, which can introduce complexity in maintaining context coherence.
- **Manual Annotation.** The reliance on manual annotation for ground truth creation constrained the size and diversity of test documents. With only 10 benchmark documents, capturing the full range of potential document structures remains a challenge. A larger annotated dataset would likely reveal further insights into the pipeline's strengths and failure modes.



## 5.2 Challenges encountered

During this research, several challenges emerged that influenced both the design and performance of the proposed LLM-based information extraction pipeline. These challenges can be broadly categorized into issues related to text extraction from diverse PDFs, maintaining semantic consistency across multiple chunks, combining extracted text from disparate sections, and handling unpredictable model behaviors. Additionally, the evaluation process itself posed significant difficulties due to the complexity of the text and the limitations of traditional NLP metrics.

### Text Extraction Difficulties

- **Complicated or Unusual Document Layouts:** Announcements for grants and funds often have irregular formats, such as tables, sidebars, or multiple columns. Extracting text from these layouts, especially in scanned PDFs, can cause broken or mixed-up information. This makes the text incomplete or hard to process further.
- **OCR Limitations:** Despite using Tesseract and preprocessing steps (e.g., orientation checks, noise removal), low-resolution scans or highly stylized documents can reduce Optical Character Recognition (OCR) accuracy, thereby introducing errors early in the pipeline. These errors propagate through subsequent stages, ultimately affecting the integrity of the extracted data.

### Maintaining Semantic Consistency Across Chunks

- **Chunking Trade-offs:** Splitting large documents into manageable pieces enables the model to handle more text than its context window allows. However, determining the optimal chunk size remains difficult. Too-large chunks risk exceeding token limits or reducing focus on local context; too-small chunks can fragment semantically related content, leading to information loss or confusion.
- **Context Overlap:** In order to preserve meaning, some overlap between chunks is necessary. Yet excessive overlap can introduce redundancy and inflate computational costs. Balancing these two factors is challenging and often domain-specific.

### Combining Extracted Data from Multiple Chunks

- **Fragmentation of Information:** Information relevant to a single section of the final "Rule Card" (e.g., "candidati-idonei") can be scattered throughout multiple PDF pages or chunks. Reconciling partial insights from different segments into a single, coherent field requires careful aggregation and validation.
- **Potential for Conflicting Details:** If multiple chunks contain overlapping or partially contradictory information, the system must adjust these conflicts. Such inconsistencies sometimes cause the model to generate incorrect summaries or omit critical elements in favor of the most relevant chunk.

### Unpredictable Model Behavior

- **Hallucination and Fabrication:** Even with carefully designed prompts, the model may "invent" details-particularly in fields that lack explicit information. This behavior, known as hallucination, influences the reliability of extracted data and demands a robust post-processing or verification step.
- **Prompt Sensitivity:** Small modifications in wording or structure of prompts can yield significantly different outputs. Fine-tuning prompts to consistently yield accurate, structured results requires iterative testing, which is both time-consuming and prone to guesswork.
- **Token Limits and Cost:** Large or highly detailed prompts and extensive text inputs increase the likelihood of hitting token limits, risking truncated outputs or higher computational expenses. These constraints can limit the depth of contextual information passed to the model.

### Evaluation Complexities

- **Unreliability of Traditional Metrics:** Classical NLP metrics (e.g., BLEU, ROUGE) are less effective for generative LLM outputs, where the focus is on capturing specific semantic elements rather than matching exact word sequences. Even more advanced methods, such as BERTScore, can struggle with nuanced, domain-specific content.
- **Complexity of Texts:** Grant documents vary in length, style, and complexity, making direct comparisons with ground-truth difficult. Automated metrics can fail to capture these complexities, leading to misleading scores.
- **Manual Ground Truth Creation:** Constructing a reliable gold standard requires detailed domain expertise and is time-consuming. The limited size of manually annotated datasets restricts the variety of scenarios tested, making it challenging to draw robust conclusions about the pipeline's performance or generalizability.

## 5.3 Implications for future work

The promising results and insights gained from this research point to several opportunities for enhancement and extension of the current LLM-based information extraction pipeline. Future work may focus on refining OCR capabilities, incorporating new data modalities, and expanding the pipeline's scope to additional document formats, domains and languages. Some important areas to explore include:

### Fine-Tuning Open-source MLLMs for Italian OCR

- **Open-source Multimodal LLMs Advancements:** It made them capable of handling OCR tasks in English, there is potential to extend its image-processing capabilities to support Italian text.

- **Training Data and Domain Specificity:** Fine-tuning models would require assembling a comprehensive dataset of Italian scans, typed and handwritten text, and diverse document formats to ensure robust performance. This step could significantly reduce reliance on separate OCR tools and improve accuracy for scanned PDF files.

### **Extending the Pipeline to Other Data Types**

- **Integration with Audio and Video:** As many data are published in multiple formats, incorporating speech-to-text and video analysis flows into the LangGraph framework could broaden the pipeline’s scope.
- **Adaptive Graph Structures:** LangGraph’s modular design allows for new nodes and flows dedicated to different media types, enabling more comprehensive information extraction. This could open the door to capturing details from multimedia sources that are currently outside the system’s scope.

### **Adapting to Other Domains and Languages**

- **Generalizing Beyond Grants:** While this research focuses on Italian regional grants and funds, the underlying principles of domain-specific filtering, chunk processing, and instruction-based extraction can be applied to other specialized domains (e.g., legal documents, insurance claims, scientific literature).
- **Language Expansion:** Customizing the pipeline for additional languages would necessitate domain-specific prompt translations and potentially new OCR modules.

### **Advanced Evaluation Strategies**

- **LLM-Based Self-Evaluation:** Testing and validating LLM-based evaluators under controlled conditions might yield improved evaluation reliability.

## **5.4 Meet the Research Objectives**

Recalling the research objectives outlined at the beginning of this thesis, this section summarizes how each goal was addressed through the methodology, implementation, and evaluation processes:

### **Investigate the existing tools and techniques available for customizing LLMs for specific tasks**

- **Comparative Overview of Customization Techniques:** Customization techniques and tools have been reviewed in the second chapter. In the methodology chapter, different customization strategies, fine-tuning, Retrieval-Augmented Generation (RAG), and prompt engineering were examined for their advantages and limitations. Emphasis was placed on cost-effectiveness, resource requirements, and technical complexity.

- **Selection Justification:** Prompt engineering was chosen for its balance between simplicity and applicability. The detailed comparison showed how more resource-intensive options like fine-tuning might yield higher accuracy but at significantly higher cost and complexity, especially for smaller enterprises.

### **To explore methods for processing domain-specific document formats, with a focus on PDF documents**

- **Focus on PDF Processing:** The methodology and results primarily addressed PDFs relevant to grant and funding announcements, demonstrating how to parse both searchable and scanned PDF documents. OCR was integrated for Italian texts, recognizing the challenges involved in extracting domain-specific content.
- **Potential Extensions:** While the scope of this thesis did not extend to video or audio, the Discussions chapter highlights clear pathways for incorporating other modalities, such as speech-to-text or video analysis flows into the existing LangGraph framework.

### **Develop a framework for integrating customized LLMs into a user-interface application**

- **Modular Pipeline and Interface:** A step-by-step pipeline was created to parse, filter, and extract structured data, while Streamlit was used to build a user-friendly web application. This setup allows non-technical users to upload documents and retrieve "Rule Card" in JSON format without needing to interact with the underlying technical components.
- **Scalable Architecture:** The pipeline's modularity (including a dedicated document reader, chunking strategy, and verification steps) enables future adaptation to new data types, new LLMs or different user requirements, fulfilling the goal of creating a flexible, integrable framework.

### **Evaluate the performance and usability of the customized application**

- **Performance Evaluation:** A custom benchmark dataset was assembled, and BERTScore was employed to measure precision, recall, and F1 score for key fields. These evaluations provided quantitative evidence of the pipeline's relative strengths and weaknesses.
- **Usability Considerations:** Through a user-interface designed in Streamlit, the application proved straightforward to operate, allowing quick uploads and automated extraction. While manual ground-truth creation remained a bottleneck.

By systematically addressing each of these objectives, this thesis demonstrated that prompt-engineered, domain-focused pipelines can effectively leverage light-weight Large Language Models for structured information extraction, albeit with recognized limitations and clear avenues for further exploration in OCR, multimodality, and broader applicability.

## Chapter 6

# Conclusion

This thesis set out to explore how Large Language Models (LLMs) can be effectively leveraged for automating information extraction from PDF documents, with a specific focus on Italian regional grants and funds documents. Addressing a clear need in many organizations, particularly SMEs where manual review of documents is both time-consuming and error-prone, the research demonstrated the feasibility and value of an LLM-based pipeline to automate the process.

A key achievement lies in the design of a modular workflow that tackles each stage of the extraction process: from PDF parsing and optical character recognition (OCR) for scanned files to prompt engineering for structured data extraction. By selectively integrating an open-source model (Llama 3.1) and focusing on prompt engineering rather than fine-tuning or Retrieval-Augmented Generation (RAG), the solution balanced both performance and practicality. This approach allowed the pipeline to remain flexible, cost-effective, and broadly applicable, making it an attractive option for smaller enterprises with limited computational resources.

Moreover, adopting open-source and lightweight LLMs gives SMEs the advantage of running the pipeline locally on-premise. This alleviates concerns regarding sharing proprietary or sensitive information with external providers, thereby preserving enterprise data privacy. Such an approach supports the organizational need for privacy compliance and offers an alternative to commercial, closed-box solutions.

The results showed the pipeline's ability to capture vital grant information such as project deadlines, financial resources, and eligible applicants at a high level of semantic accuracy. Incorporating a tailored chunking strategy, domain-specific filtering, and robust prompt engineering enabled the system to surpass generic extraction methods and reduce issues related to token limits. Meanwhile, careful post-processing (including validation prompts) mitigated some common pitfalls in LLM outputs, such as hallucination or incomplete extractions. While challenges remain particularly in handling suboptimal OCR results, ensuring semantic consistency across multiple text chunks, and evaluating outputs with limited ground-truth datasets the methodology validated the potential of domain-focused LLM applications.

Looking forward, the thesis suggests exciting opportunities for future improvements. These include fine-tuning open-source multimodal LLMs for tasks involving Italian OCR,

adding support for other data types such as audio and video, and extending the solution to handle other important documents besides grants and funds. Using advanced evaluation methods, could make the system more reliable and allow for better ways to measure model performance. Exploring these ideas can help make LLM-based pipelines more reliable, scalable, and useful in different real-world applications.

In summary, this thesis underscores how domain-aware prompt engineering and strategic pipeline design can significantly improve the automatic extraction of structured information from unstructured documents. The research not only demonstrates practical benefits for organizations seeking to streamline their document-processing workflows but also highlights the evolving capabilities of LLMs in specialized contexts. Crucially, by leveraging light-weight and open-source models, SMEs can automate document-processing tasks on-premise, safeguarding privacy while gaining efficiency and accuracy in their operations.

# Bibliography

- [1] Prompt engineering vs fine-tuning vs rag. <https://myscale.com/blog/prompt-engineering-vs-finetuning-vs-rag/>, 2024. Accessed: March 26, 2024.
- [2] Bhashithe Abeysinghe and Ruhan Circi. The challenges of evaluating llm applications: An analysis of automated, human, and llm-based approaches. *arXiv preprint arXiv:2406.03339*, 2024.
- [3] HLEG AI. High-level expert group on artificial intelligence. *Ethics guidelines for trustworthy AI*, 6, 2019.
- [4] Md Adnan Arefeen, Biplob Debnath, and Srimat Chakradhar. Leancontext: Cost-efficient domain-specific question answering using llms. *Natural Language Processing Journal*, 7:100065, 2024.
- [5] Muhammad Arslan and Christophe Cruz. Business-rag: Information extraction for business insights. *ICSBT 2024*, page 88, 2024.
- [6] Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. A survey on rag with llms. *Procedia Computer Science*, 246:3781–3790, 2024.
- [7] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.
- [8] Michele Basilico. *Design, Implementation and Evaluation of a Chatbot for Accounting Firm: A Fine-Tuning Approach With Two Novel Dataset*. PhD thesis, Politecnico di Torino, 2024.
- [9] Pavan Belagatti. Llms architecture, 2024. Accessed: May 29, 2024.
- [10] Cecylia Borek. Comparative evaluation of llm-based approaches to chatbot creation. 2024.
- [11] Bruno Cabral, Daniela Claro, and Marlo Souza. Exploring open information extraction for portuguese using large language models. In *Proceedings of the 16th International Conference on Computational Processing of Portuguese*, pages 127–136, 2024.
- [12] Kilian Carolan, Laura Fennelly, and Alan F Smeaton. A review of multi-modal large language and vision models. *arXiv preprint arXiv:2404.01322*, 2024.
- [13] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.
- [14] Brad Kyoshi Donohoo. Machine learning techniques for energy optimization in mobile embedded systems. Master’s thesis, Colorado State University, 2012.

- [15] Raffaello Fornasiere. Exploring the potential of lightweight llms for medication and timeline extraction. 2023.
- [16] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- [17] Cornelis Geerligs. *Information Extraction from Contracts Using Large Language Models*. PhD thesis, 2024.
- [18] Samira Ghodrattnama and Mehrdad Zakershahra. Adapting llms for efficient, personalized information retrieval: Methods and implications. In *International Conference on Service-Oriented Computing*, pages 17–26. Springer, 2023.
- [19] Johannes B Gruber and Maximilian Weber. rollama: An r package for using generative large language models through ollama. *arXiv preprint arXiv:2404.07654*, 2024.
- [20] Muhammad Usman Hadi, Qasem Al Tashi, Abbas Shah, Rizwan Qureshi, Amgad Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, et al. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 2024.
- [21] Yikun Han, Chunjiang Liu, and Pengfei Wang. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*, 2023.
- [22] Kadhim Hayawi, Sakib Shahriar, Hany Alashwal, and Mohamed Adel Serhani. Generative ai and large language models: A new frontier in reverse vaccinology. *Informatics in Medicine Unlocked*, page 101533, 2024.
- [23] Technology Innovation Institute. Terms and conditions, 2023. Accessed: May 29, 2024.
- [24] Jeffrey Ip. Llm evaluation metrics: The ultimate llm evaluation guide. <https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation>, 2024. Accessed: November 07, 2024.
- [25] Jeffrey Ip. Llm evaluation metrics: The ultimate llm evaluation guide, 2025. Accessed: January 27, 2025.
- [26] Deepack Jakhar and Ishmeet Kaur. Artificial intelligence, machine learning and deep learning: definitions and differences. *Clinical and experimental dermatology*, 45(1):131–132, 2020.
- [27] La Javaness. Llm large language model cost analysis. <https://lajavaness.medium.com/llm-large-language-model-cost-analysis-d5022bb43e9e>, 2023. Accessed: July 17, 2024.
- [28] Ruhma Khawaja. Orchestration frameworks: Simplify llm-apps with langchain and llama index. <https://datasciencedojo.com/blog/orchestration-frameworks/>, 2023. Accessed: September 05, 2024.
- [29] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.



- [30] Yiheng Liu, Hao He, Tianle Han, Xu Zhang, Mengyuan Liu, Jiaming Tian, Yutong Zhang, Jiaqi Wang, Xiaohui Gao, Tianyang Zhong, et al. Understanding llms: A comprehensive overview from training to inference. *arXiv preprint arXiv:2401.02038*, 2024.
- [31] Hectorn Martinez. llama.cpp: The ultimate guide to efficient llm inference and applications. <https://pyimagesearch.com/2024/08/26/llama-cpp-the-ultimate-guide-to-efficient-llm-inference-and-applications/>, 2024. Accessed: August 26, 2024.
- [32] Dorota Owczarek. Customizing large language models: A comprehensive guide. <https://nexocode.com/blog/posts/customizing-large-language-models-a-comprehensive-guide/#the-process-of-customizing-llms>, 2024. Accessed: March 05, 2024.
- [33] Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.
- [34] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024.
- [35] Lucas Persson. Automating data extraction from documents using large language models: A study exploring how ai can be used to transform unstructured data into structured formats, 2024.
- [36] Mohaimenul Azam Khan Raiaan, Md Saddam Hossain Mukta, Kaniz Fatema, Nur Mohammad Fahad, Sadman Sakib, Most Marufatul Jannat Mim, Jubaer Ahmad, Mohammed Eunus Ali, and Sami Azam. A review on large language models: Architectures, applications, taxonomies, open issues and challenges. *IEEE Access*, 2024.
- [37] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.
- [38] Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616, 2024.
- [39] Christoph Schmidt. Multimodality in large language models â how ai is becoming more humanlike. <https://lamarr-institute.org/blog/multimodality-llms/>, 2024. Accessed: June 19, 2024.
- [40] Aditi Singh, Abul Ehtesham, Saifuddin Mahmud, and Jong-Hoon Kim. Revolutionizing mental health care through langchain: A journey with a large language model. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0073–0078. IEEE, 2024.
- [41] Stefano Strippoli. *Natural Language Processing with Generative AI Models: A Methodological Approach for Their Application*. PhD thesis, Politecnico di Torino, 2023.

- [42] Annie Surla, Aditi Bodhankar, and Tanay Varshney. An easy introduction to multimodal retrieval-augmented generation. <https://developer.nvidia.com/blog/an-easy-introduction-to-multimodal-retrieval-augmented-generation/>, 2024. Accessed: March 20, 2024.
- [43] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- [44] Jinheng Wang, Hansong Zhou, Ting Song, Shaoguang Mao, Shuming Ma, Hongyu Wang, Yan Xia, and Furu Wei. 1-bit ai infra: Part 1.1, fast and lossless bitnet b1. 58 inference on cpus. *arXiv preprint arXiv:2410.16144*, 2024.
- [45] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [46] Sang Michae Xie and Sewon Min. How does in-context learning work? a framework for understanding the differences from traditional supervised learning. <https://ai.stanford.edu/blog/understanding-incontext/>, 2022. Accessed: August 01, 2024.
- [47] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357, 2024.
- [48] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. On protecting the data privacy of large language models (llms): A survey. *arXiv preprint arXiv:2403.05156*, 2024.
- [49] Eugene Yan. Open llms: A collection of open large language models, 2024. Accessed: December 19, 2024.
- [50] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.
- [51] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [52] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

# Appendix A

## Project Repository

GitHub Repository: <https://github.com/Hesamedin2010/Information-Extractor>

### Project Structure

The `information_extractor` project is organized into several directories, each responsible for a specific part of the information extraction pipeline. Below is an overview of the project structure:

```
information_extractor
|- Chains (LangChain's chains for LLM tasks)
|  |- __init__.py
|  |- document_checker.py: Checks document relevance.
|  |- format_controller.py: Verifies and consolidates extracted information.
|  |- information_extractor.py: Extracts structured information using LLMs.
|  |- ocr_improver.py: Corrects OCR errors in scanned documents.
|
|- Nodes (LangGraph nodes for pipeline stages)
|  |- __init__.py
|  |- document_reader.py: Detects document format.
|  |- extract_json.py: Saves structured JSON output.
|  |- extractor.py: Calls the information extraction chain.
|  |- filter_documents.py: Filters irrelevant documents.
|  |- format_verification.py: Finalizes extracted information.
|  |- image_to_text.py: Performs OCR and preprocessing.
|  |- pdf_parser.py: Parses text from searchable PDFs.
|
|- Root Directory
|  |- .env.template: Template for environment variables.
|  |- consts.py: Constants for LangGraph node functions.
|  |- Evaluation_BERTScore.py: Output evaluation using BERTScore.
|  |- graph.py: Constructs the LangGraph pipeline.
|  |- pipfile & pipfile.lock: Dependency management.
```

```
| |- run.py: Executes the pipeline.  
| |- state.py: Defines LangGraph state formats.  
| |- ui.py: Streamlit-based user interface.  
| |- variables.py: Configurable settings for different cases.  
| |- visualize_graph.py: Graph structure visualization.  
| |_.gitignore: Specifies files to ignore in version control.
```

## Environment Setup

To set up the project environment, follow these steps:

1. **Install Pipenv:** Ensure that Pipenv is installed on your system.
2. **Install Dependencies:** Run the following command to install all required dependencies:

```
pipenv install
```

3. **Activate the Environment:** Enter the virtual environment with:

```
pipenv shell
```

4. **Set Up Environment Variables:**

- Copy `.env.template` to `.env`.
- Fill in the required API keys and other configurations.

## Running the Application

Once the environment is set up, you can run the application:

- **Run the Pipeline:** Process PDF documents by executing:

```
python run.py --pdf_dir <directory_of_pdfs>
```

- **Launch the UI:** Start the Streamlit-based user interface with:

```
streamlit run ui.py
```

For further details, please refer to the **README** file in the repository.