Design and Development of an AI-Driven RAG-Based Intelligent Tender Document Analysis System for ALTEN Morocco

Final Year Project Report

Presented for the completion of the degree of
State Engineer at the School of Information Sciences

MAATI Manal

Under the supervision of:

Dr. SENNOUNI Amine
Mr. SENNAH Marouane

Jury Members:
President: .....
Member 1: .....
Member 2: .....

Class of 2026

Dedication

This work is dedicated, with the deepest love and utmost gratitude, to my wonderful family—my ***mother, father, sister, and brother****. Your* ***unconditional love, profound sacrifices, and unwavering support*** *have been the very bedrock of my academic journey. Through every late night and every moment of doubt, your absolute trust in my capabilities served as a constant sanctuary and my greatest source of inspiration. Thank you for providing the foundation upon which I could build my dreams, for celebrating my small victories, and for offering the endless encouragement that pushed me to transcend my limits and achieve this milestone. I am infinitely grateful for the sacrifices you made to ensure my success.*

I also extend this dedication to my esteemed ***teachers, professors, and mentors****. Your invaluable guidance, wisdom, and tireless commitment to excellence have not only shaped the trajectory of this project but have also profoundly influenced my intellectual and personal development. You did not just teach; you inspired a genuine passion for learning. Alongside my educators, I honor my dear* ***friends****, who walked this path with me. Thank you for your unwavering camaraderie, for the shared laughter that lightened the heaviest days, and for turning an arduous academic pursuit into a beautiful chapter filled with unforgettable moments. Your support made all the difference.*

Finally, I dedicate this achievement to ***myself****—acknowledging the quiet resilience, discipline, and relentless determination it took to navigate this path. This project stands as a testament to the sleepless nights, the inner strength summoned during moments of exhaustion, and the unwavering faith required to overcome every obstacle. I honor the growth, the perseverance, and the spirit that refused to give up, ultimately transforming a distant vision into a successful reality.*

Acknowledgements

First and foremost, I would like to express my sincere gratitude and deepest appreciation to the Data Team at Alten for welcoming me so warmly and for providing an exceptionally professional, collaborative, and motivating environment throughout the realization of this project. Being immersed in such an innovative atmosphere allowed me to bridge the gap between academic theory and real-world application, making this experience truly invaluable to my professional journey.
I would especially like to extend my heartfelt thanks to my supervisor for their outstanding

mentorship. Your valuable guidance, continuous support, and constructive advice during every phase of this work were instrumental to its success. Thank you for your endless patience, for sharing your technical expertise so generously, and for always challenging me to approach problems with a critical and analytical mindset.

I also wish to convey my sincere gratitude to the esteemed professors, coordinators, and administration of the École des Sciences de l'Information (ESI). Thank you for the exceptional quality of education, rigorous academic supervision, and unwavering support provided throughout my studies. The knowledge and principles instilled in me by this institution have been the foundation upon which this project was built.

Finally, I would like to express my deepest, most heartfelt gratitude to my family. Your boundless patience, endless encouragement, and unconditional love have been my ultimate source of strength. Thank you for your quiet sacrifices and for being a constant pillar of support throughout this entire journey, ensuring I had everything I needed to succeed.

Abstract

The modern professional landscape demands rapid, accurate processing of complex legal and commercial documentation, among which tender documents represent a significant bottleneck due to their density, varied formatting, and unstructured nature. This Final Year Project presents the design, development, and implementation of an intelligent, end-to-end software solution specifically engineered for automated tender document analysis. Combining principles of software engineering, advanced data processing, and intelligent systems, the primary objective of this project is to optimize document management workflows, drastically reduce manual verification efforts, and enhance information retrieval accuracy within professional environments.

Methodologically, this work explores and evaluates multiple cutting-edge Retrieval-Augmented Generation (RAG) paradigms—including traditional RAG, GraphRAG, RAPTOR-based architectures, and Structured Retrieval-Augmented Generation (SRAG)—to identify the optimal framework for semantic data extraction from dense PDFs. The finalized production pipeline incorporates robust text extraction for both native and scanned documents via Optical Character Recognition (OCR), text preprocessing, semantic chunking, and vector embedding generation utilizing a FAISS-based vector index. Leveraging a Large Language Model (LLM) to yield highly accurate, structured summaries, the entire intelligent pipeline is seamlessly integrated into a responsive web application powered by the Django framework. Ultimately, this project serves as a real-world validation of advanced AI in document processing, while fostering deep technical competencies in information systems, intelligent semantic retrieval, and modern full-stack application development.

Keywords: Intelligent Document Analysis, Tender Documents, Retrieval-Augmented Generation (RAG), Structured Retrieval (SRAG), Data Processing, Software Engineering, Django, Intelligent Systems.

Résumé

Le monde professionnel actuel exige un traitement rapide et précis de documentations juridiques et commerciales complexes, parmi lesquelles les documents d'appel d'offres représentent un défi majeur en raison de leur densité, de la diversité de leurs formats et de leur nature non structurée. Ce Projet de Fin d'Études présente la conception, le développement et la mise en œuvre d'une solution logicielle intelligente et complète spécialement conçue pour l'analyse automatisée des documents d'appel d'offres. En combinant les principes du génie logiciel, du traitement avancé des données et des systèmes intelligents, l'objectif principal de ce projet est d'optimiser les flux de gestion documentaire, de réduire considérablement les efforts de vérification manuelle et d'améliorer la précision de la recherche d'informations dans des environnements professionnels.

D'un point de vue méthodologique, ce travail explore et évalue plusieurs paradigmes de pointe du Retrieval-Augmented Generation (RAG) — notamment le RAG traditionnel, le GraphRAG, les architectures basées sur RAPTOR, ainsi que le Structured Retrieval-Augmented Generation (SRAG) — afin d'identifier le cadre optimal pour l'extraction sémantique de données à partir de fichiers PDF complexes. Le pipeline de production final intègre une extraction robuste du texte pour les documents natifs et numérisés grâce à la reconnaissance optique de caractères (OCR), le prétraitement du texte, le découpage sémantique ainsi que la génération d'embeddings vectoriels à l'aide d'un index vectoriel basé sur FAISS. En exploitant un Large Language Model (LLM) pour produire des résumés structurés et hautement précis, l'ensemble du pipeline intelligent est intégré de manière fluide dans une application web réactive développée avec le framework Django. Finalement, ce projet constitue une validation concrète de l'utilisation avancée de l'intelligence artificielle dans le traitement documentaire, tout en favorisant l'acquisition de compétences techniques approfondies dans les systèmes d'information, la recherche sémantique intelligente et le développement moderne d'applications full-stack.

Mots-clés : Analyse Intelligente de Documents, Documents d'Appel d'Offres, Retrieval-Augmented Generation (RAG), Structured Retrieval (SRAG), Traitement des Données, Génie Logiciel, Django, Systèmes Intelligents.

List of Figures

List of Tables

List of Abbreviations

RAG	Retrieval-Augmented Generation
LLM	Large Language Model
OCR	Optical Character Recognition
FAISS	Facebook AI Similarity Search
PDF	Portable Document Format
Django	Web framework in Python
SRAG	Structured Retrieval-Augmented Generation
R&D	Research and Development
IT	Information Technology
AI	Artificial Intelligence
NLP	Natural Language Processing
NER	Named Entity Recognition
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
PCA	Principal Component Analysis
t-SNE	t-Distributed Stochastic Neighbor Embedding
CPU	Central Processing Unit
GPU	Graphics Processing Unit

Contents

Chapter 1

General Framework and Research Methodology

1.1 Presentation of the Host Organization

1.1.1 Introduction

The successful completion of a final year project requires a clear understanding of its organizational context and the methodological approach adopted to conduct the research. Establishing this framework helps ensure that the project objectives are clearly defined and achieved in a structured manner.

This chapter presents the general framework of the project and the methodology followed during its realization. It begins with an overview of the host organization, followed by the presentation of the problem statement that motivates this work. The chapter then outlines the research objectives, research questions, and the methods adopted to carry out the study. Finally, it introduces the project planning and situates the research within the field of artificial intelligence and intelligent document analysis.

1.1.2 General Description

Back in 1988, a tech-focused company took shape in France - now known worldwide by the name ALTEN. This firm specializes in engineering tasks alongside research and development work for clients across continents. From its early days it branched into information technology support, building reputation through hands-on solutions. Global demand grew steadily due to consistent delivery on complex projects. Its foundation was local but ambitions stretched far beyond borders right from the start. From the start, this team kept expanding, thanks to a solid role in advanced engineering fields while shifting easily when industries changed around it. Growth never slowed as technology moved forward and factories evolved worldwide.

Now found in over thirty nations throughout Europe, Asia, the Americas, and Africa, ALTEN keeps step with client needs. Because of where it is located, the company sharpens how engineering support works - using distant teams when smart, nearby ones when better. From sketching ideas to running tests, ALTEN works alongside big companies in industry and services. Whether it is shaping concepts or fine-tuning systems, their role stays steady. Through modeling phases or real-world rollouts, support continues. Even once things are live, help remains available. Maintenance, updates, problem solving - these follow naturally.

Last year saw the company pull in about €4.14 billion in combined income, showing just how wide its reach has become. Roughly 57,700 people work within the network, sup-

porting over 6,500 customers across continents. Out here, squads mix engineers with data folks, consultants, and tech experts - each crew spread thin over industries like planes, cars, power systems, telecoms, money operations, even military work. Groups form differently every time, stitching roles together depending on what the project needs that week. Some tackle flight software Monday, switch to bank networks by Thursday. No two setups look alike, yet each keeps pace through constant reshuffling of who does what. One team might lean heavy on coders for a satellite rollout, next month pivot hard into electric vehicle security. The pattern shifts but never breaks. Every field gets covered without ever sticking to one formula. Across eleven nations, ALTEN runs thirty-five offshore delivery hubs - each one boosting reach for big-scale work. These centers help stretch capacity where it matters most.

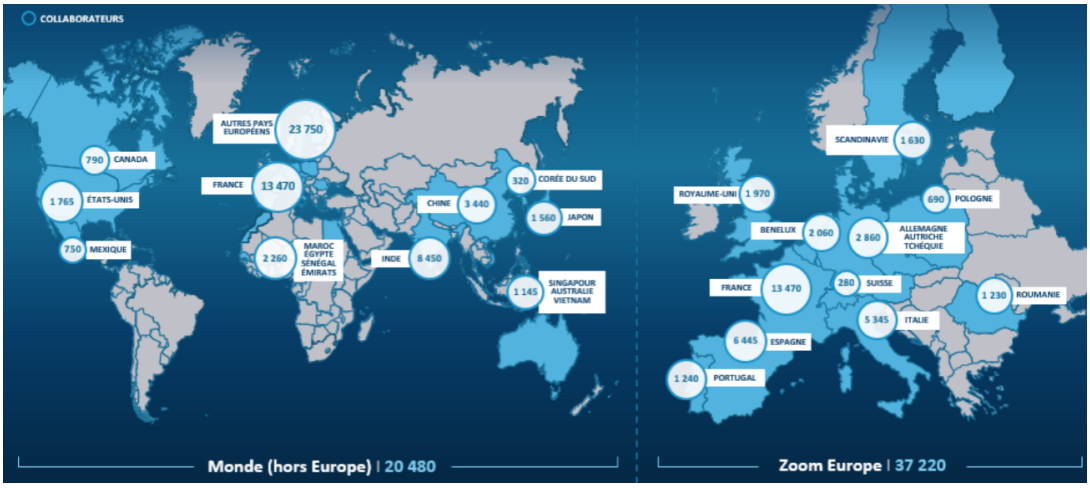

Figure 1.1: Worldwide presence

1.1.3 ALTEN Maroc

Among the key branches of the ALTEN Group is ALTEN Maroc, which plays an important role in the group's global engineering delivery model. It acts as a strategic hub that supports the execution of engineering projects across different regions.

Offshore activities carried out in Morocco contribute significantly to global projects, allowing expertise developed locally to be leveraged internationally. This strengthens ALTEN's ability to deliver complex engineering solutions at scale.

ALTEN Maroc operates through four main sites in Morocco, which serve as operational hubs for daily activities. Each site contributes to the organization's objectives by supporting different aspects of project delivery.

Approximately 2,200 collaborators work within ALTEN Maroc, including around 2,100 consultants dedicated to Offshore Delivery Center (ODC) activities. This makes ALTEN Maroc a key component of ALTEN's global delivery structure.

As part of the ALTEN network, ALTEN Maroc benefits from strong engineering expertise and close collaboration between teams. This environment promotes knowledge sharing and enhances the quality of delivered solutions.

ALTEN Maroc provides high-value engineering and IT services across several domains, including embedded systems, software engineering, data engineering, and cybersecurity.

The teams contribute to the design, development, and maintenance of complex systems used by international industrial clients. Their work supports different stages of the

product lifecycle, from development to continuous improvement.

The Moroccan subsidiary collaborates with several major international companies, including Stellantis, Renault Group, Alstom, Bouygues Telecom, Telefónica, Carrefour, Worldline, and CNP Assurances. These partnerships reflect strong confidence in ALTEN Maroc's ability to deliver large-scale and critical engineering projects.

In addition, ALTEN Maroc follows international quality and security standards, including ISO certifications and TISAX compliance, ensuring high levels of reliability, security, and operational excellence.

Thanks to its integration within ALTEN's global network, ALTEN Maroc ensures smooth collaboration between offshore teams and international operations.

Overall, ALTEN Maroc plays a key role in connecting European industrial needs with high-performance engineering and digital solutions.

1.1.4 Activities of the Group

The ALTEN Group's activities are mainly structured around engineering consulting, technological innovation, and digital transformation services. It supports clients across the full lifecycle of technological systems, including design, development, industrialization, deployment, maintenance, and continuous improvement.

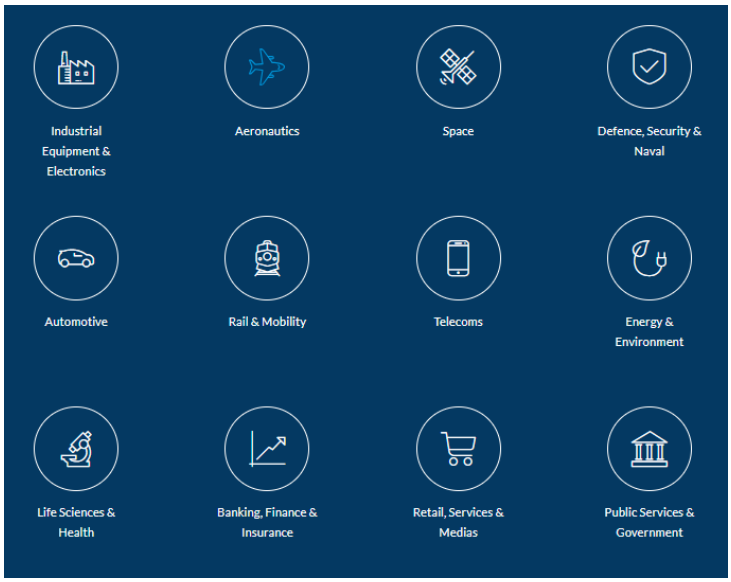

Figure 1.2: Activities of the ALTEN Group

ALTEN offers a diversified portfolio of services that helps organizations address complex engineering challenges, improve efficiency, and accelerate digital transformation. These activities are organized into several core domains:

- **Engineering and R&D Services:** Advanced engineering solutions across aerospace, automotive, railway, and energy sectors, including design, simulation, embedded systems, testing, and validation to improve system performance and reliability.
- **Industrial Process Engineering:** Optimization of industrial operations through process improvement, supply chain management, quality assurance, and technical documentation, ensuring efficiency and compliance with standards.

- **Digital Transformation and Information Systems:** Development of digital platforms, enterprise applications, and IT architectures to modernize information systems and improve organizational performance.
- **Artificial Intelligence and Data Engineering:** Data-driven solutions using AI, machine learning, and analytics to automate processes and extract insights from structured and unstructured data.
- **Cybersecurity and Cloud Computing:** Secure infrastructures and cloud solutions ensuring data protection, system resilience, and continuity, including risk management and secure development practices.
- **Project Management and Agile Consulting:** Support for Agile transformation, DevOps adoption, and project management practices to improve collaboration, delivery speed, and software quality.

1.1.5 Partners and Clients of the Group

ALTEN collaborates with a wide range of international organizations, industrial leaders, and service-sector companies across multiple strategic domains. Through its multidisciplinary engineering and consulting expertise, the group supports its partners in the design, development, and optimization of complex technological systems, contributing to large-scale industrial transformation projects worldwide.

The group operates across diverse industrial and technological sectors, reflecting its ability to address complex engineering challenges. These sectors include:

- **Aerospace and Space:** development, simulation, and optimization of aeronautical systems, satellites, and space technologies.
- **Defense, Security & Naval:** mission-critical systems, cybersecurity, and naval engineering solutions.
- **Automotive, Railway & Mobility:** embedded systems, autonomous mobility, and intelligent transportation solutions.
- **Energy & Environment:** renewable energy systems, smart grids, and sustainability-focused engineering.
- **Life Sciences & Healthcare:** digital and engineering solutions for medical systems and healthcare innovation.
- **Industrial Equipment & Electronics:** industrial automation, embedded systems, and electronic design.
- **Telecommunications:** network engineering, 5G technologies, and communication infrastructures.
- **Banking, Finance & Insurance:** digital transformation, data engineering, and cybersecurity solutions.
- **Retail, Services & Government:** digital platforms, e-commerce, and e-governance systems.

Through its presence in these sectors, ALTEN delivers high-value engineering solutions tailored to industry needs. This positioning enables the group to maintain strong partnerships with global clients while continuously supporting innovation and digital transformation.

1.1.6 Host Team and Work Environment

Working on this project meant joining ALTEN Maroc's Data Team. Within the context of digital evolution, their work centers on using data to spark new approaches. Tasks span areas like analyzing information, building smart systems, generating business insight, and organizing how data flows across platforms.
The group brings together people from different backgrounds - some focused on interpreting trends, others on modeling algorithms, structuring pipelines, or ensuring reliability. Though each person handles distinct responsibilities along the data chain, outcomes emerge through shared effort. Insights take shape gradually, shaped by diverse skills working in sequence. Results serve real-world needs, turning unprocessed inputs into meaningful outputs.

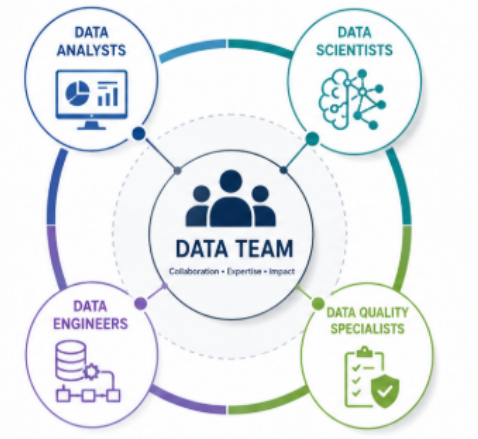

Figure 1.3: ALTEN Morocco's Data Teams

Working with numbers, data analysts study how businesses operate to uncover patterns over time. Through detailed examination of records, they produce summaries that guide choices within organizations. Their role centers on turning raw information into clear insights using structured methods. Focused on accuracy, these professionals help teams understand performance through evidence-based findings.

Starting with patterns hidden in information, data scientists build tools that forecast outcomes through machine learning instead of guesswork. Their work leans heavily on statistics, shaping raw numbers into usable insights. Often beginning with messy datasets, they apply refined methods to clean and interpret complex inputs. Rather than relying on intuition, these professionals use structured algorithms to uncover relationships within data. With an emphasis on precision, their models support decisions across industries by revealing what might happen next.

Building systems to gather, store, and move information falls to Data Engineers. These professionals shape how data flows through organizations. Their work supports reliable access while keeping structures running smoothly. Through careful development, they enable timely processing across platforms. Maintaining architecture becomes a core

task alongside initial design efforts. Efficiency often guides their choices in pipeline construction.

Reliability of data often depends on specialists who check how consistent it appears throughout different systems. Accuracy becomes clearer when these professionals examine patterns where details might otherwise slip through. Integrity matters just as much, especially once information moves between platforms. Projects tend to run smoother if someone is watching for errors before they grow. What stands out most is how careful review shapes trustworthy results behind the scenes.

Working together closely, the group applies current development techniques along with Agile approaches to maintain smooth progress and ongoing refinement. Because teamwork is central, insights spread easily, creativity grows, and results meet what clients actually need.

Being part of this team allowed me to gain practical exposure to real-world data engineering and artificial intelligence projects while improving my technical, analytical, and professional skills. It also enabled me to better understand the importance of data governance, data quality, and scalable data infrastructures in the development of intelligent systems and digital transformation initiatives. Within this context, the following section introduces the problem addressed in this project

1.2 Problem Statement

In modern organizations, managers and decision-makers are frequently required to handle large volumes of textual documents in order to support both operational and strategic decisions. Among these documents, tender documents and calls for proposals are particularly critical, as they directly influence business opportunities and contractual outcomes.

However, these documents are typically long, complex, and highly structured, containing technical specifications, administrative requirements, financial constraints, and legal obligations. This high level of detail makes their manual analysis both time-consuming and cognitively demanding, as users must carefully extract relevant information from dense and heterogeneous content.

As a result, manual document review often leads to reduced efficiency, longer processing times, and an increased risk of human error or missing key information. These limitations become even more significant when large numbers of tenders must be processed under strict deadlines, where delays or inaccuracies can directly impact decision-making.

In recent years, advancements in Artificial Intelligence (AI), particularly in Natural Language Processing (NLP), have opened new possibilities for automating the understanding of unstructured textual data. Large Language Models (LLMs) have demonstrated strong capabilities in interpreting and generating human-like text, while Retrieval-Augmented Generation (RAG) architectures further enhance these models by grounding responses in relevant external knowledge sources.

Despite these advances, there is still a clear need for dedicated systems tailored specifically to tender document analysis, where precision, completeness, and reliability are essential. General-purpose solutions remain insufficient when applied to such domain-specific and high-stakes documents.

Therefore, this project aims to design and implement an intelligent system capable of automatically analyzing tender documents and extracting key information relevant to decision-makers. By leveraging state-of-the-art AI techniques, the proposed solution

seeks to reduce document processing time, improve information accessibility, and enhance the accuracy and usability of extracted data.

1.3 Methodological Framework of the Research

The methodological framework adopted in this project provides a structured and systematic way to address the research problem. It combines theoretical study with practical analysis in order to design and develop an intelligent system for automated tender document processing.

Rather than relying on a single approach, this framework brings together research objectives, research questions, and data collection methods. These elements guide the design, development, and evaluation of the proposed solution in a coherent way.

1.3.1 Research Objectives (Operational Objectives)

The main goal of this project is to design and develop an intelligent system that helps managers analyze tender documents by automatically extracting relevant information using artificial intelligence techniques.

To reach this goal, several more specific objectives have been defined:

- Understand the organization's needs regarding the processing and analysis of tender documents
- Explore existing techniques in Natural Language Processing (NLP), Large Language Models (LLMs), and intelligent document analysis
- Study relevant approaches such as Retrieval-Augmented Generation (RAG), Named Entity Recognition (NER), and document processing systems
- Design and implement a system that allows users to upload documents and automatically extract useful information
- Evaluate the proposed solution in terms of accuracy, efficiency, and ease of use.

1.3.2 Research Questions

This research is guided by the following key questions:

- What are the main challenges when analyzing large volumes of tender documents?
- Which artificial intelligence techniques are most effective for extracting relevant information from these documents?
- How can large language models and RAG architectures improve document understanding and processing?
- Which tools and technologies are best suited for building such a system?
- To what extent can this system improve the efficiency of managers in their document analysis tasks?

1.3.3 Agile Methodology

The development of this project followed an Agile methodology, emphasizing iterative delivery, continuous improvement, and close collaboration with stakeholders. This approach was adopted due to the exploratory nature of the project and the need to progressively refine the solution based on feedback and evolving requirements.

A structured weekly follow-up was established with the supervisor at Alten to ensure consistent monitoring of project progress. These review sessions provided a formal governance framework for tracking deliverables, validating ongoing developments, and identifying potential risks or blocking points at an early stage.

This iterative execution model enabled continuous alignment with project objectives, facilitated rapid decision-making, and ensured incremental enhancement of the system throughout the development lifecycle, in accordance with industry best practices in software engineering.

1.4 Research Instruments

To carry out this final-year project effectively and provide reliable answers to the research questions, a clear and structured approach was adopted. This approach combines different research methods in order to balance theoretical understanding with practical needs.

The methodology is based on two main components: documentary research and field investigation through discussions with stakeholders. Together, these methods help build a complete understanding of both the technical background and the real needs of the system.

1.4.1 Documentary Method

The documentary method involves searching, selecting, and analyzing existing scientific and technical resources related to the topic. It plays an important role in building a solid theoretical foundation for the project.

This work included reviewing academic papers, research articles, technical reports, and industry publications in areas such as Artificial Intelligence, Natural Language Processing, and intelligent document processing systems.

The sources used in this study fall into two main categories:

- Internal documentation provided by the host organization, which helped in understanding the business context, existing workflows, and the needs related to tender document processing.

- External academic and scientific sources, including peer-reviewed papers and technical publications covering topics such as Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Named Entity Recognition (NER), and information extraction techniques.

This combination of sources made it possible to build a strong literature review and provided the necessary background for designing and implementing the proposed AI-based solution.

1.4.2 Survey Method

The survey method, carried out through structured and semi-structured interviews, involved direct discussions with project supervisors and relevant stakeholders within the organization.

These discussions were essential for understanding how things currently work on the ground and for collecting detailed information about user expectations and existing processes.

The main goals of these interviews were to:

- understand the current workflow for processing and reviewing tender documents
- identify the main limitations, challenges, and inefficiencies in the existing manual process
- gather functional and non-functional requirements from end users, especially managers
- follow up on the project's progress and ensure continuous feedback throughout the development process

1.5 Project Planning

Project planning represents a fundamental aspect of effective project management. It consists of decomposing the project into a set of well-defined tasks and organizing them in a logical and chronological sequence to ensure efficient execution. As illustrated in the image, the project schedule was established using a Gantt chart, providing a clear visual representation of task durations, dependencies, parallel executions, and key milestones.

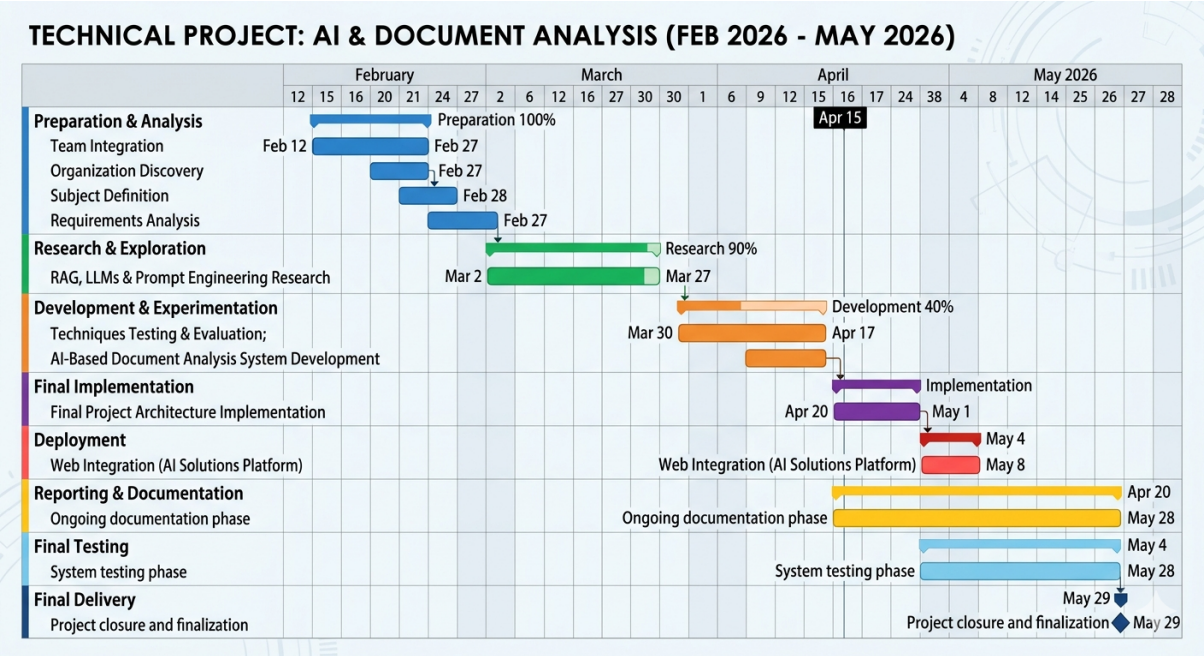

Figure 1.4: Gantt diagram for the project planning

The project is structured into the following key phases:

– Preparation & Analysis: This phase includes initial team integration and organizational discovery, followed by subject definition and requirements analysis.

– Research & Exploration: This phase focuses on an in-depth investigation of Retrieval-Augmented Generation (RAG), Large Language Models (LLMs), and prompt engineering techniques.

– Development & Experimentation: This phase involves the parallel execution of system development and techniques testing and evaluation to iteratively improve the solution.

– Final Implementation: This phase consists of consolidating the final project architecture based on the outcomes of experiments and validations.

– Deployment: This phase covers the final web integration of the AI solutions platform and system finalization.

– Reporting & Documentation: This is a continuous phase that spans both the final implementation and testing stages, ensuring comprehensive documentation and project delivery.

This planning framework enables continuous monitoring of project progress and ensures that all tasks are completed within the allocated timeframe.

1.6 Research Positioning

This research is situated within the broader domains of Artificial Intelligence (AI) and Natural Language Processing (NLP), with a particular focus on intelligent document analysis and automated information extraction systems. These fields have gained significant attention in recent years due to the increasing need to process and analyze large volumes of unstructured textual data in organizational and industrial contexts.

Recent advancements in AI, particularly the development of Large Language Models (LLMs), have significantly enhanced the capability of machines to understand, interpret, and generate human language with a high degree of accuracy. In parallel, retrieval-based architectures, and more specifically Retrieval-Augmented Generation (RAG) approaches, have further improved the reliability of AI systems by combining generative capabilities with external knowledge retrieval mechanisms. This hybrid approach enables more context-aware, precise, and explainable outputs when dealing with complex documents.

These technologies are increasingly applied across a wide range of domains, including knowledge management systems, legal and regulatory document analysis, automated report generation, customer support automation, and enterprise content management. Their adoption reflects a growing need for intelligent systems capable of transforming unstructured data into actionable knowledge.

Within this context, the present project is positioned at the intersection of intelligent document processing and applied AI systems for decision support. It specifically focuses on the application of advanced NLP techniques, including LLMs and RAG-based architectures, to the analysis of tender documents—an area characterized by high complexity, structured constraints, and critical decision-making requirements.

The contribution of this work lies in the design and implementation of an intelligent system that leverages these modern AI approaches to automate the extraction of

relevant information from tender documents. By addressing both technical and organizational challenges, this research aims to improve document processing efficiency, enhance information accessibility, and support more effective decision-making within an enterprise environment.

1.7 Conclusion

This chapter provided a comprehensive overview of the general framework of the project. It began with the presentation of the host organization, highlighting its main activities and its role within the context of engineering and digital transformation.

Subsequently, the problem statement was introduced, emphasizing the challenges associated with the manual analysis of complex tender documents and the need for more efficient and automated solutions. Based on this context, the research objectives and key research questions were defined, establishing a clear direction for the study.

The chapter then outlined the methodological framework adopted to conduct this research, combining documentary analysis with field-based investigation through stakeholder interactions. This approach ensured both a solid theoretical foundation and a practical understanding of organizational needs.

In addition, the project planning was presented, providing a structured view of the different phases involved in the realization of the system. Finally, the research was positioned within the broader domains of Artificial Intelligence and Natural Language Processing, with a particular emphasis on intelligent document analysis and modern AI techniques such as Large Language Models and Retrieval-Augmented Generation.

Overall, this chapter establishes the conceptual and methodological basis of the project. The following chapter will build upon this foundation by presenting the theoretical background and a detailed state of the art of the Artificial Intelligence techniques used in document analysis and information extraction.

Chapter 2

Theoretical Framework and Literature Review

2.1 Introduction

Following the presentation of the general context, problem statement, and methodological framework in the previous chapter, this chapter aims to establish the theoretical foundations that underpin the proposed solution. A solid understanding of these concepts is essential for the design and implementation of an intelligent system dedicated to the automated analysis of tender documents.

In recent years, significant advancements in Artificial Intelligence (AI) have transformed the way unstructured textual data is processed and exploited within organizations. In particular, the fields of Natural Language Processing (NLP) and Large Language Models (LLMs) have enabled the development of sophisticated systems capable of understanding, interpreting, and generating human language with a high level of accuracy. These technologies play a central role in modern intelligent document processing systems, where the ability to extract relevant information from large and complex documents is of critical importance.

Within this context, new paradigms such as Retrieval-Augmented Generation (RAG) have emerged, combining the strengths of information retrieval techniques and generative models to enhance the precision, reliability, and contextual relevance of extracted information. These approaches are particularly well-suited for applications involving complex and structured documents, such as tender documents, where both completeness and accuracy are essential.

The objective of this chapter is therefore to present the theoretical framework that supports the development of the proposed system. It begins with an overview of the fundamental concepts of Natural Language Processing and Large Language Models, followed by a detailed presentation of Retrieval-Augmented Generation and its underlying mechanisms. The chapter then introduces document processing pipelines and retrieval strategies, directly linking theoretical concepts to the technical implementation adopted in this project. Finally, it provides a review of existing research works and state-of-the-art approaches in intelligent document analysis and information extraction.

This chapter thus serves as a bridge between the conceptual foundations of Artificial Intelligence and their practical application within the scope of this project.

2.2 Artificial Intelligence and Deep Learning Foundations

2.2.1 Artificial Intelligence Overview

Artificial Intelligence (AI) refers to the development of computer systems capable of performing tasks that normally require human intelligence, such as reasoning, learning, problem solving, decision-making, and language understanding. Over the past few decades, AI has evolved from rule-based systems that relied on manually defined logic to more advanced data-driven approaches capable of learning from large volumes of information.

The rapid growth of digital data has significantly accelerated the adoption of AI across multiple sectors. Today, AI technologies are widely used in healthcare, finance, cybersecurity, recommendation systems, autonomous systems, and intelligent document processing.

In the context of this project, AI provides the foundation for automating the analysis of tender documents. Since these documents contain large amounts of unstructured textual information, intelligent techniques are required to automatically identify relevant content and support decision-making processes.

2.2.2 Machine Learning and Deep Learning

Machine Learning (ML) is a subfield of Artificial Intelligence that enables systems to learn patterns from data and improve their performance without being explicitly programmed for every task. Instead of relying entirely on predefined rules, machine learning models identify relationships within historical data to make predictions or classifications.

Traditional machine learning techniques include algorithms such as decision trees, logistic regression, and Support Vector Machines (SVMs), which have been widely used for classification and prediction tasks.

With the increasing availability of large datasets and computational power, Deep Learning emerged as a more advanced branch of machine learning. Deep learning relies on artificial neural networks composed of multiple layers that automatically learn complex representations of data.

These models have significantly improved performance in domains such as image recognition, speech processing, and natural language understanding.

In particular, deep learning has transformed Natural Language Processing through the development of Transformer architectures, which enabled major breakthroughs in language understanding and generation. Models such as BERT and GPT are direct results of these advancements and play an important role in modern document analysis systems.

2.3 Natural Language Processing (NLP)

2.3.1 Definition and Overview

Natural Language Processing (NLP) is a fundamental subfield of Artificial Intelligence (AI) and computational linguistics that focuses on enabling machines to process, interpret, and generate human language in a coherent, meaningful, and context-aware manner

[1]. It lies at the intersection of linguistics, computer science, and statistical learning, and plays a central role in transforming unstructured textual data into structured and actionable knowledge.

The rapid growth of digital information has led to an exponential increase in textual data generated by organizations. Consequently, NLP has become an essential component of modern intelligent systems, particularly for applications involving large-scale document analysis, information retrieval, and automated decision support.

Formally, let us consider a corpus of textual documents defined as:

$$D = \{d_1, d_2, \ldots, d_n\},$$

where each document $d_i \in D$ represents an unstructured sequence of tokens such as words, sentences, or paragraphs.

The primary objective of NLP is to learn a transformation function:

$$f : D \to I,$$

where I denotes a structured representation of the information extracted from the corpus. This representation may include named entities, semantic relationships, document classifications, summaries, or answers to user queries.

More precisely, for an individual document d_i, the transformation can be expressed as:

$$f(d_i) = I_i,$$

where I_i is the structured representation associated with document d_i.

From a structured information extraction perspective, this output can be further formalized as:

$$I_i = \{(k_1, v_1), (k_2, v_2), \ldots, (k_m, v_m)\},$$

where each k_j represents an extracted attribute (e.g., Objective, Budget, Duration, Payment Terms), and each v_j corresponds to the value identified in the document.

From a computational perspective, the function f can be implemented using various approaches, including:

- Statistical methods, which rely on probabilistic modeling of language.
- Rule-based systems, which use predefined linguistic patterns.
- Neural network architectures, particularly deep learning models such as Transformers.
- Hybrid approaches, which combine symbolic reasoning with data-driven learning.

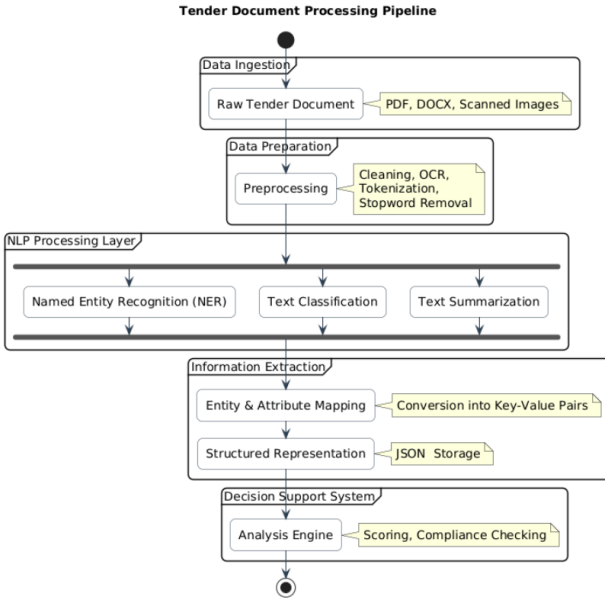

Figure 2.1: End-to-end pipeline for automated tender document processing using Natural Language Processing techniques.

In the context of this project, the function f corresponds to the transformation of raw tender documents into structured information fields such as budget, deadline, project scope, and contractual terms. This structured representation is directly exploitable for managerial decision-making.

NLP encompasses a wide range of fundamental tasks, each addressing a specific aspect of language understanding. These tasks can be formally modeled using mathematical frameworks and constitute essential building blocks in intelligent document processing systems.

In this project, several core NLP tasks—such as text classification, named entity recognition, summarization, and information retrieval—are integrated into a unified processing pipeline to enable the automated analysis of complex tender documents. Collectively, these tasks bridge the gap between unstructured textual content and structured, high-value information.

2.3.2 Text Classification

Text classification is a fundamental task in Natural Language Processing (NLP) that consists of assigning a predefined category to a given document based on its semantic content. It plays a crucial role in structuring unorganized textual data and enabling efficient downstream processing in intelligent information systems.

Formally, let $d \in D$ be a document and let $C = \{c_1, c_2, \ldots, c_k\}$ be a predefined set of classes. The objective of text classification is to assign to d the most probable class $y \in C$ such that:

$$y = \arg\max_{c \in C} P(c \mid d; \theta)$$

where $P(c \mid d; \theta)$ denotes the conditional probability that document d belongs to class c, parameterized by a model θ. This function is learned from labeled training data using supervised learning techniques.

Classical approaches for estimating this function include probabilistic models such as Naïve Bayes and discriminative models such as Support Vector Machines (SVMs) [17]. More recently, deep learning architectures based on Transformer models, such as BERT and RoBERTa, have achieved state-of-the-art performance by learning contextual representations of text.

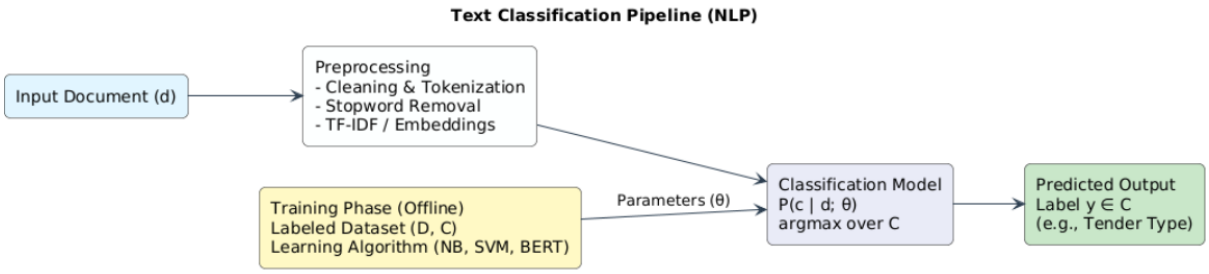

Figure 2.2: Architecture of the text classification process in NLP

In the context of this project, text classification is used to categorize tender documents into predefined classes such as document type, thematic domain, or level of complexity. This step helps organize documents before applying further Natural Language Processing tasks in the pipeline.

Formally, the classification model can be expressed as:

$$f_{\text{cls}}(d) = y, \quad y \in C$$

where f_{cls} maps each input document d to a single label representing its assigned category. In the case of multi-label classification, the output may instead be a subset of labels:

$$f_{\text{cls}}(d) = \{y_1, y_2, \ldots, y_m\}, \quad y_i \in C$$

This classification step improves document organization and supports downstream tasks such as information retrieval, filtering, and prioritization within the proposed intelligent document analysis system.

2.3.3 Named Entity Recognition (NER)

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing (NLP) that focuses on the automatic identification and classification of meaningful information units, referred to as entities, within unstructured text. These entities typically correspond to semantic categories such as dates, organizations, monetary values, locations, and domain-specific attributes.

Formally, given an input text sequence composed of tokens:

$$T = \{w_1, w_2, \ldots, w_n\},$$

the objective of NER is to assign to each token a label indicating whether it belongs to an entity and the type of that entity.

This can be modeled as a sequence labeling problem:

$$y_i = f(w_i, w_{1:n}; \theta),$$

where y_i is the predicted label for token w_i, and θ represents the parameters of the model.

A common representation of an extracted entity is a span-based tuple:

$$e_i = (s_i, e_i, \tau_i),$$

where s_i and e_i denote the start and end positions of the entity in the text, and τ_i represents its semantic type (e.g., *DATE*, *ORGANIZATION*, *MONEY*).

The set of extracted entities from a document is then defined as:

$$E = \{e_1, e_2, \ldots, e_m\}.$$

Modern NER systems are typically implemented using probabilistic models or deep learning architectures, particularly Transformer-based models such as BERT, which leverage contextual embeddings to improve entity recognition and disambiguation accuracy [18].

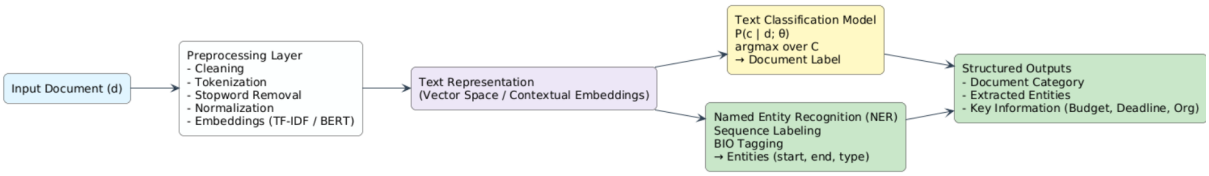

Figure 2.3: End-to-end Natural Language Processing pipeline combining text classification and Named Entity Recognition

In the context of this project, Named Entity Recognition is applied to extract structured information from tender documents, such as submission deadlines, contract amounts, issuing organizations, technical requirements, and certification constraints.

The system produces a structured representation of extracted entities:

$$I_d = \{(\text{Deadline}, v_1), (\text{Budget}, v_2), (\text{Organization}, v_3), \ldots\},$$

where each pair represents a normalized attribute-value structure derived from the recognized entities.

This transformation enables unstructured tender documents to be converted into machine-readable data, which can be stored, queried, and integrated into decision-support systems.

2.3.4 Text Summarization

Text summarization is a Natural Language Processing (NLP) task that aims to automatically generate a concise and informative representation of a document while preserving its essential meaning. It is particularly useful for reducing the cognitive load associated with large textual data.

Formally, summarization can be modeled as a conditional generation problem. Given a document d, the objective is to generate a summary S that maximizes:

$$S^* = \arg\max_S P(S \mid d)$$

where $P(S \mid d)$ denotes the probability of generating a coherent and relevant summary conditioned on the input document.

Text summarization methods are generally divided into two categories. *Extractive summarization* selects the most relevant sentences from the source text based on statistical or semantic criteria [3]. In contrast, *abstractive summarization* generates new sentences by rephrasing the original content using natural language generation models, typically based on Transformer architectures.

Recent advances in Large Language Models (LLMs) have significantly improved abstractive summarization by enhancing contextual understanding and long-range dependency modeling [4].

In this project, summarization is applied to tender documents to produce a condensed representation:

$$S_d = f_{\text{summ}}(d)$$

where $f_{\text{summ}}(\cdot)$ denotes the summarization function, and S_d highlights key elements such as objectives, deadlines, financial terms, and contractual constraints. This improves the efficiency of document analysis and decision-making.

2.3.5 Information Retrieval (IR)

Information Retrieval (IR) is a fundamental subfield of Natural Language Processing (NLP) concerned with the identification and ranking of relevant information from a large collection of documents in response to a user query. It constitutes a core component of modern intelligent document processing systems, particularly those operating on large-scale unstructured textual corpora.

Formally, given a corpus of documents $D = \{d_1, d_2, \ldots, d_n\}$ and a user query q, the objective of information retrieval is to identify the most relevant document d^* such that:

$$d^* = \arg\max_{d_i \in D} \text{Sim}(q, d_i)$$

where $\text{Sim}(q, d_i)$ denotes a similarity function that measures the semantic or lexical proximity between the query q and a document d_i. Traditional approaches rely on vector space models and statistical weighting schemes such as TF-IDF [5], while modern approaches leverage dense vector representations (embeddings) computed using deep neural networks.

In particular, similarity is often computed using cosine similarity between embedding vectors:

$$\text{Sim}(q, d_i) = \frac{\mathbf{v}_q \cdot \mathbf{v}_{d_i}}{\|\mathbf{v}_q\| \, \|\mathbf{v}_{d_i}\|}$$

where $\mathbf{v}_q$ and $\mathbf{v}_{d_i}$ represent the dense vector embeddings of the query and document respectively. These embeddings are typically generated using transformer-based models such as Sentence-BERT or other sentence embedding architectures [6].

In the context of modern intelligent systems, Information Retrieval is no longer an isolated process but is tightly integrated with generative models through Retrieval-Augmented Generation (RAG) architectures. In such systems, retrieved documents or passages are used as contextual grounding for Large Language Models (LLMs), enabling them to generate more accurate, context-aware, and factually consistent responses.

Within the scope of this project, Information Retrieval plays a central role in the processing of tender documents. Given a user query q (e.g., regarding deadlines, budget

constraints, or eligibility criteria), the system retrieves the most relevant sections of tender documents from a structured corpus. These retrieved segments are then passed to the generative module to produce precise and contextually enriched outputs.

This retrieval mechanism ensures that the system is not solely dependent on parametric knowledge embedded within the language model but is instead grounded in external, domain-specific data sources. As a result, the proposed architecture significantly improves reliability, traceability, and interpretability of generated responses in the context of tender document analysis.

2.3.6 NLP in the Context of Tender Document Analysis

Within the scope of this project, Natural Language Processing (NLP) is a fundamental enabling technology for the automated interpretation and structured analysis of tender documents. These documents are typically characterized by their length, domain-specific terminology, and semi-structured or unstructured nature, which makes manual processing both time-consuming and error-prone. Consequently, NLP techniques are employed to transform raw textual content into structured, machine-readable representations that can support efficient decision-making.

The proposed system leverages multiple NLP tasks in a unified processing pipeline to extract and organize relevant information from tender documents. More specifically, the system performs:

- **Document classification:** automatic categorization of tenders according to their domain, type, or sector, enabling efficient indexing and filtering within large document repositories.
- **Named Entity Recognition (NER):** extraction of critical entities such as submission deadlines, contract values, issuing authorities, eligibility requirements, and legal constraints.
- **Text summarization:** generation of concise representations of full documents, allowing stakeholders to quickly understand the core content without processing the entire text.
- **Information structuring:** transformation of unstructured textual data into structured formats suitable for downstream processing, storage, and analysis.

From a formal perspective, let d represent a tender document belonging to a corpus D. The NLP pipeline can be modeled as a composite transformation function:

$$I_d = f_{\text{NLP}}(d)$$

where f_{NLP} denotes a sequence of NLP operations including classification, entity extraction, and summarization, and I_d represents the resulting structured information extracted from the document.

This structured representation can be expressed as:

$$I_d = \{\text{Deadline: } t,\ \text{Budget: } b,\ \text{Agency: } a,\ \text{Requirements: } r, \dots\}$$

where each element corresponds to a semantically meaningful field extracted from the original document d. This transformation enables the conversion of unstructured

legal and administrative text into a normalized format that is directly exploitable by decision-support systems.

In this project, the output of the NLP module is integrated into a broader intelligent document processing pipeline based on Retrieval-Augmented Generation (RAG). The extracted structured data not only improves indexing and retrieval performance but also enhances the contextual grounding of generative models used for answering user queries over tender corpora.

As a result, NLP acts as a foundational layer in the proposed architecture, bridging the gap between raw textual data and high-level semantic understanding required for automated tender analysis and intelligent information extraction.

2.3.7 Importance in Modern AI Systems

The field of Natural Language Processing (NLP) has undergone a significant transformation with the advent of Large Language Models (LLMs), such as BERT and GPT-based architectures [18, 4]. These models have substantially improved the ability of machine learning systems to capture syntactic structures, semantic dependencies, and contextual relationships within natural language, thereby overcoming many limitations of traditional statistical and rule-based approaches.

From a methodological perspective, modern LLMs are typically based on Transformer architectures, which model long-range dependencies in text through self-attention mechanisms [7]. This enables a more comprehensive representation of linguistic context, allowing systems to achieve state-of-the-art performance across a wide range of tasks, including text classification, named entity recognition, information extraction, and question answering.

In parallel, the integration of NLP with Retrieval-Augmented Generation (RAG) frameworks has introduced a new paradigm in intelligent information systems. Rather than relying exclusively on parametric knowledge encoded within neural models, RAG-based architectures dynamically retrieve relevant external knowledge from large document repositories and incorporate it into the generation process. This hybrid approach significantly enhances factual accuracy, contextual relevance, and domain adaptability.

Formally, given a query q, a retrieval component first identifies a set of relevant documents $D_r \subset D$, which are then used to condition a generative model G, producing an output y:

$$y = G(q, D_r)$$

This formulation ensures that the generated response is grounded in external knowledge sources, reducing hallucination and improving interpretability.

In the context of this project, the combination of NLP techniques and RAG-based architectures is particularly well-suited for tender document analysis. Tender documents often contain dense, domain-specific, and legally constrained information that requires both precise extraction and context-aware interpretation. By leveraging LLMs for semantic understanding and RAG for knowledge retrieval, the proposed system is able to generate accurate, concise, and contextually grounded outputs from complex document collections.

Consequently, NLP, when integrated with modern LLMs and retrieval mechanisms, forms the backbone of intelligent document processing systems, enabling scalable and reliable automation of knowledge extraction tasks in enterprise environments.

2.4 Large Language Models (LLMs)

2.4.1 Definition

Large Language Models (LLMs) are a class of deep neural network architectures designed to model and generate natural language by learning probability distributions over large-scale textual corpora. These models are primarily based on the Transformer architecture, which leverages self-attention mechanisms to capture long-range dependencies and complex semantic relationships within sequential data.

The development of LLMs has significantly advanced the field of Natural Language Processing (NLP), enabling systems to perform a wide range of tasks such as text generation, summarization, question answering, information extraction, and document classification with strong contextual understanding [18, 4].

From a formal standpoint, let a textual input be represented as a sequence of tokens:

$$X = (x_1, x_2, \ldots, x_n)$$

An LLM models the joint probability of the sequence by factorizing it into conditional probabilities:

$$P_\theta(X) = \prod_{t=1}^{n} P_\theta(x_t \mid x_{<t})$$

where $x_{<t} = (x_1, \ldots, x_{t-1})$ and θ denotes the learnable parameters of the model. This autoregressive formulation enables next-token prediction conditioned on previously generated tokens, supporting both understanding and generation.

The training objective is typically defined as the minimization of the negative log-likelihood (cross-entropy loss):

$$\mathcal{L}(\theta) = -\frac{1}{n} \sum_{t=1}^{n} \log P_\theta(x_t \mid x_{<t})$$

This objective encourages the model to assign higher probabilities to correct token sequences, improving syntactic and semantic consistency in generated outputs.

In the context of this project, LLMs serve as the core generative component of the Retrieval-Augmented Generation (RAG) pipeline. While retrieval modules provide relevant contextual passages from a corpus of tender documents, the LLM generates responses grounded in this external information rather than relying solely on parametric knowledge encoded in its weights.

Consequently, LLMs play a central role in transforming retrieved textual evidence into coherent and meaningful outputs, particularly in tender document analysis, where accuracy, contextual grounding, and interpretability are essential.

2.4.2 Transformer Architecture and Representation Learning

The Transformer architecture constitutes a major breakthrough in deep learning for Natural Language Processing (NLP), as it replaces recurrent and convolutional structures with a fully attention-based mechanism. Introduced by Vaswani et al., it has become the foundational architecture for modern Large Language Models (LLMs) due to its ability to efficiently model long-range dependencies and scale to large textual corpora.

At a fundamental level, the Transformer processes a sequence of tokens $X = (x_1, x_2, \ldots, x_n)$ by computing contextual representations that capture dependencies between all elements in the sequence. This is achieved through the self-attention mechanism, which dynamically assigns importance weights to different tokens depending on their relevance within the context.

Self-Attention Mechanism

Self-attention is a core component of Transformer-based architectures that enables the model to compute contextual representations of input tokens by modeling dependencies between all positions in a sequence.

Given an input sequence represented as a matrix $X \in \mathbb{R}^{n \times d}$, each token is projected into three distinct representations: Query, Key, and Value:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_k}$ are learnable parameter matrices.

The attention mechanism computes a weighted aggregation of values based on similarity between queries and keys:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where d_k is the dimensionality of the key vectors. The scaling factor $\sqrt{d_k}$ prevents excessively large dot-product values, improving numerical stability and gradient behavior during training.

This mechanism allows the model to capture both local and global dependencies within the input sequence, making it particularly effective for processing long and complex documents such as tender specifications.

Figure 2.4 illustrates the Transformer architecture proposed by Vaswani et al. [7], which relies entirely on self-attention mechanisms and removes recurrent structures.

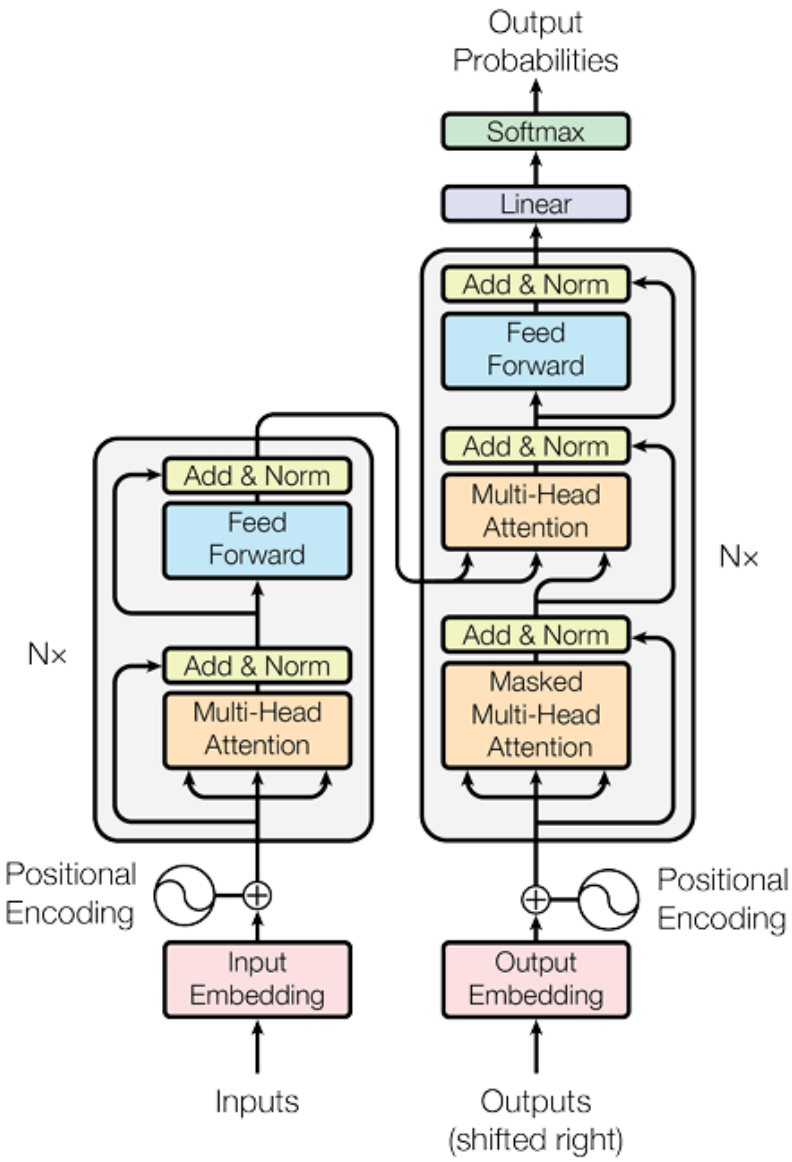

Figure 2.4: Transformer architecture based on self-attention mechanisms

Multi-Head Attention and Positional Encoding

To enhance representational capacity, the Transformer employs multi-head attention, where multiple attention heads are computed in parallel, allowing the model to capture diverse semantic and syntactic relationships.

The multi-head attention mechanism is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(H_1, H_2, \ldots, H_h)W^O$$

where h denotes the number of attention heads and W^O is a learnable output projection matrix.

Each attention head is computed independently as:

$$H_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

where W_i^Q, W_i^K, W_i^V are learnable projection matrices corresponding to the i-th head.

This design enables the model to jointly attend to information from different representation subspaces at different positions.

Since the Transformer architecture does not inherently model sequential order, positional encodings are added to the input embeddings to inject information about token positions. The final input representation is given by:

$$X' = X + PE$$

where PE denotes positional encoding, which can be either fixed (e.g., sinusoidal functions) or learned embeddings. This mechanism ensures that the model preserves the order and structural relationships within the input sequence.

Encoder–Decoder Architecture

The Transformer is structured as an encoder–decoder architecture, enabling sequence-to-sequence learning for tasks such as summarization and information extraction.

The encoder maps the input sequence X into a set of contextualized representations:

$$E = f_{\text{Encoder}}(X)$$

where E represents the encoded memory of the input sequence. The encoder is composed of stacked layers, each including multi-head self-attention, feedforward neural networks, residual connections, and layer normalization, which together ensure stable optimization and effective representation learning.

The decoder generates the output sequence in an autoregressive manner by modeling the conditional probability:

$$P(Y \mid X) = \prod_{t=1}^{T} P(y_t \mid y_{<t}, E)$$

where $y_{<t}$ denotes previously generated tokens and E is the encoder output.

Each decoder layer consists of masked multi-head self-attention (to prevent access to future tokens), encoder–decoder attention (to attend over encoder representations), and feedforward layers. This design ensures that generated outputs are both contextually coherent and grounded in the input sequence.

Embeddings and Semantic Representation

Embeddings are dense vector representations that encode semantic and syntactic properties of linguistic units, enabling discrete text to be mapped into continuous vector spaces where similarity can be measured mathematically.

Token Embeddings Token embeddings map discrete tokens into a continuous vector space using a learned embedding matrix:

$$e_i = W_e \cdot \text{onehot}(x_i)$$

where x_i is the input token and $W_e \in \mathbb{R}^{|V| \times d}$ is the embedding matrix. Classical methods such as Word2Vec and GloVe learn static embeddings, where each word has a fixed representation independent of context, limiting their ability to handle polysemy.

Contextual Embeddings In Transformer-based models, contextual embeddings are produced by encoding the full input sequence through multiple self-attention layers:

$$E = (e_1, e_2, \ldots, e_n) = f_{\text{Transformer}}(X)$$

where each e_i depends on all tokens in the sequence X through self-attention mechanisms. This enables dynamic representations that capture meaning based on context, as used in models such as BERT and GPT.

Document Embeddings To obtain a fixed-length representation of a document, token embeddings are aggregated into a single vector. A simple approach is mean pooling:

$$E_d = \frac{1}{n} \sum_{i=1}^{n} e_i$$

A more expressive approach uses attention-based pooling:

$$\alpha_i = \frac{\exp(w^\top e_i)}{\sum_{j=1}^{n} \exp(w^\top e_j)}, \quad E_d = \sum_{i=1}^{n} \alpha_i e_i$$

where w is a learnable parameter vector that assigns importance weights to tokens.

Embedding Visualization Although embeddings exist in high-dimensional spaces, dimensionality reduction techniques such as PCA and t-SNE can be used to project them into lower dimensions for visualization and interpretability, often revealing semantic clustering of related concepts.

2.4.3 Sentence Embedding Model: all-MiniLM-L6-v2

Definition

`all-MiniLM-L6-v2` is a lightweight sentence embedding model developed within the SentenceTransformers framework [6]. It is designed to transform textual data—ranging from words to full sentences and paragraphs—into dense vector representations called embeddings.

These embeddings encode the semantic meaning of the input text in a continuous high-dimensional space, enabling efficient comparison based on meaning rather than exact lexical matching.

The model is based on the MiniLM architecture, a distilled variant of Transformer-based models such as BERT [8]. Through knowledge distillation, MiniLM preserves strong semantic representation capabilities while significantly reducing computational cost.

Key Characteristics

The main characteristics of `all-MiniLM-L6-v2` are:

- **Model type:** Sentence Transformer for semantic embeddings
- **Architecture:** MiniLM (distilled Transformer)
- **Depth:** 6 Transformer layers

– **Embedding dimension:** 384

– **Performance:** Competitive semantic representation quality

– **Efficiency:** Low latency and reduced computational cost

– **Hardware:** Runs efficiently on CPUs (no GPU required)

– **Availability:** Open-source

Operational Principle

Sentence embedding models map input text into a fixed-dimensional vector space:

$$f_\theta : \mathcal{T} \to \mathbb{R}^{384}$$

where $\mathcal{T}$ represents the space of input texts and $\mathbb{R}^{384}$ denotes the embedding space learned by the model.

Given an input text, the model produces an embedding $e \in \mathbb{R}^{384}$ using a Transformer encoder followed by a pooling strategy:

$$e = \text{Pool}(f_{\text{Transformer}}(X))$$

To measure semantic similarity between two embeddings e_1 and e_2, cosine similarity is used:

$$\text{Sim}(e_1, e_2) = \frac{e_1 \cdot e_2}{\|e_1\| \cdot \|e_2\|}$$

This enables comparison of texts based on semantic meaning rather than lexical overlap.

– "I love programming" $\to e_1$

– "Coding is my passion" $\to e_2$

Even though the wording differs, the embeddings e_1 and e_2 are close in vector space, reflecting semantic similarity.

Applications

`all-MiniLM-L6-v2` is widely used in NLP tasks:

– **Semantic Search:** Retrieving documents based on meaning

– **Question Answering:** Improving retrieval in conversational systems

– **Text Similarity:** Comparing sentences or documents

– **Clustering:** Grouping similar documents

– **RAG Systems:** Retrieving relevant documents for LLMs

– **Document Classification:** Feature representation for ML models

Rationale for Model Selection

The choice of `all-MiniLM-L6-v2` is motivated by its balance between performance and efficiency:

- Fast inference suitable for real-time applications
- Low memory usage
- No need for GPU acceleration
- Easy integration with vector databases and RAG pipelines

This makes it particularly suitable for embedding-based components in tender document analysis systems, especially within Retrieval-Augmented Generation (RAG) architectures.

2.4.4 Fine-Tuning and Domain Adaptation

Fine-tuning improves accuracy [11] for:

- Named Entity Recognition (NER)
- Contract clause extraction
- Document classification

Prompt tuning adjusts model behavior without changing parameters.

2.4.5 Evaluation Metrics

Evaluation metrics are essential for assessing the performance of NLP models, including Large Language Models (LLMs) such as `gemini-2.5-flash`. In the context of tender document analysis, these metrics ensure that generated summaries, classifications, and extracted entities are accurate, reliable, and useful for decision-making.

Evaluation metrics are grouped according to task type.

1. Classification Metrics In classification tasks, the model assigns each document or segment to predefined categories such as tender type, sector, or priority level. The most commonly used metrics are accuracy, precision, recall, and F1-score.

Accuracy measures the proportion of correctly classified instances:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively.

For imbalanced datasets, precision, recall, and F1-score provide more informative evaluation:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2. Summarization Metrics Summarization systems are evaluated by comparing generated summaries with reference summaries.

ROUGE [9] measures n-gram overlap between generated and reference summaries:

$$\text{ROUGE-N} = \frac{\sum_{g \in \text{Ref}} \min(\text{Count}_{\text{cand}}(g), \text{Count}_{\text{ref}}(g))}{\sum_{g \in \text{Ref}} \text{Count}_{\text{ref}}(g)}$$

BLEU [10] measures n-gram precision with a brevity penalty:

$$\text{BLEU} = BP \cdot \exp\Big(\sum_{n=1}^{N} w_n \log p_n\Big)$$

where p_n is the n-gram precision, w_n are weights, and BP is the brevity penalty.

3. Entity Extraction Metrics Named Entity Recognition (NER) evaluates the system's ability to extract structured information from text at the token level, such as organizations, deadlines, budgets, and contractual constraints.

The evaluation uses standard classification metrics:

$$\text{Precision} = \frac{\text{Correctly Extracted Entities}}{\text{Total Extracted Entities}}$$

$$\text{Recall} = \frac{\text{Correctly Extracted Entities}}{\text{Total Relevant Entities}}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Example:
If `gemini-2.5-flash` extracts 50 entities and 45 are correct:

$$\text{Precision} = 0.90$$

If the document contains 60 true entities:

$$\text{Recall} = 0.75$$

$$\text{F1-score} \approx 0.81$$

4. Importance in Tender Document Analysis

– Ensures accurate extraction of critical information such as budgets, deadlines, and contracting entities.

– Evaluates completeness and readability of generated summaries.

– Provides quantitative measures for validating the NLP pipeline performance.

2.4.6 Limitations and Challenges

Despite their strong capabilities, Large Language Models (LLMs) such as `Gemini 2.5 Flash` present several inherent limitations when applied to domain-specific tasks such as tender document analysis. These limitations arise from their probabilistic nature and reliance on large-scale pretraining rather than explicit domain knowledge.

Hallucinations and Factual Inconsistencies

A major limitation of LLMs is hallucination, where the model generates fluent but factually incorrect or unsupported information. This is particularly critical in tender analysis, where errors may involve incorrect deadlines, financial figures, or contractual conditions.

Mitigation Strategy: This issue is addressed using Retrieval-Augmented Generation (RAG), which grounds the model in external knowledge sources. Given a query q, relevant document segments $D_r \subset D$ are retrieved and used as contextual input. The generation process can be formalized as:

$$y = \arg\max_{y} P(y \mid q, D_r)$$

Additionally, prompt constraints enforce the use of retrieved context only, reducing hallucination and improving factual consistency.

Sensitivity to Prompt Engineering

LLM performance is highly sensitive to prompt design. Poorly structured prompts may lead to irrelevant or incomplete outputs, especially in structured tasks such as entity extraction or classification.

Mitigation Strategy: This limitation is addressed through structured prompt engineering, which:

- Clearly defines the task (classification, extraction, or summarization)
- Specifies expected output formats (e.g., JSON structures)
- Restricts the model to retrieved context from the RAG pipeline

This improves consistency and reduces ambiguity in generated responses.

Computational Complexity and Resource Constraints

LLMs are computationally expensive and may introduce latency, particularly when processing long documents or multiple queries. Another key limitation is the restricted context window, which limits the amount of text that can be processed simultaneously.

Mitigation Strategy: To address these constraints, the system uses:

- Document chunking to reduce input size
- Embedding-based retrieval to limit context to relevant segments
- Lightweight sentence embedding models (e.g., Sentence-BERT variants)
- Selective LLM invocation on retrieved content only

This hybrid design improves scalability and reduces computational cost.

Domain Adaptation Limitations

Although LLMs are trained on large and diverse corpora, they may underperform in specialized domains such as procurement and tender documentation, which contain structured legal and administrative language.

Mitigation Strategy: This limitation is mitigated through:

- Retrieval-Augmented Generation (RAG) for domain grounding
- Task-specific prompt engineering
- Use of robust general-purpose embedding models for semantic retrieval

Future improvements may include domain-specific fine-tuning or continued pretraining on procurement datasets.

Summary

Overall, while LLMs provide powerful capabilities for natural language understanding and generation, their limitations require hybrid solutions. In this project, the combination of retrieval mechanisms, structured prompting, and optimized inference ensures robustness, factual accuracy, and efficiency in tender document analysis.

2.4.7 Integration with Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) [12] enhances Large Language Model (LLM) outputs by incorporating external knowledge retrieved from a document corpus. In this project, retrieval is performed at the chunk level, where each document is segmented into smaller textual units to improve retrieval precision and contextual relevance.

- **Chunk-Based Document Retrieval:** Given a query q, the system retrieves the top-k most relevant text chunks $C_r = \{c_1, \ldots, c_k\}$ from the document corpus C using dense embeddings and cosine similarity.
- **Retrieval Process:** Each chunk is encoded into a dense vector representation using a sentence embedding model. The retrieved set is defined as:

$$C_r = \underset{c \in C}{\text{TopK}} \ \text{sim}(e_q, e_c)$$

  where e_q and e_c are embeddings of the query and document chunks.

- **Generation with Context:** The LLM generates an output conditioned on both the query and the retrieved chunks:

$$y \sim P(y \mid q, C_r)$$

  This formulation ensures that the generated response is grounded in retrieved external knowledge rather than relying solely on parametric memory.

- **Applications in Tender Analysis:** This approach enables accurate extraction of structured information such as budgets, deadlines, and legal clauses, while also supporting coherent summarization for decision-making.

- **Advantages:**
  - Improves factual accuracy through grounding in retrieved evidence
  - Reduces hallucinations in LLM outputs
  - Scales efficiently to large document collections
  - Enables fine-grained chunk-level retrieval for higher relevance

2.4.8 Related Concepts

- **Tokenization:** Splits text into tokens (words, subwords, or characters) for efficient processing by LLMs.
- **Contextual Embeddings:** Represent tokens based on surrounding context, allowing the model to differentiate multiple meanings of the same word. *Example:* "Bank" in finance vs. riverbank.
- **Multi-Head Attention:** Allows the model to attend to different parts of the input simultaneously, capturing diverse relationships between tokens:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{d_k}\right)V$$

- **Prompting and Fine-Tuning:** Prompting adapts model behavior via input queries, while fine-tuning updates model weights on domain-specific corpora for improved performance.
- **RAG (Retrieval-Augmented Generation):** Combines external document retrieval with LLM generation to reduce errors and hallucinations.
- **Other LLMs:** OpenAI GPT series, Google PaLM, Meta LLaMA, Anthropic Claude.

2.5 Retrieval-Augmented Generation (RAG)

2.5.1 Concept of RAG

Retrieval-Augmented Generation (RAG) is an advanced natural language processing paradigm that combines Information Retrieval (IR) techniques with Large Language Models (LLMs) to produce outputs that are both contextually relevant and grounded in external knowledge [12]. Unlike conventional LLMs, which rely solely on parametric knowledge learned during pretraining, RAG dynamically incorporates external information sources at inference time.

The central idea behind RAG is to enhance generative models with an explicit retrieval mechanism that selects the most relevant information from a large external corpus. These retrieved text segments are then provided as additional context to the language model, enabling more accurate and domain-aware responses.

This architecture is particularly effective for domain-specific applications such as tender document analysis, where documents are long, structured, and contain critical constraints such as financial conditions, deadlines, and legal requirements. In such contexts,

relying solely on parametric memory is insufficient, making external retrieval essential for ensuring reliability and precision.

This process is illustrated in Figure 2.5, which presents the general architecture of a Retrieval-Augmented Generation system.

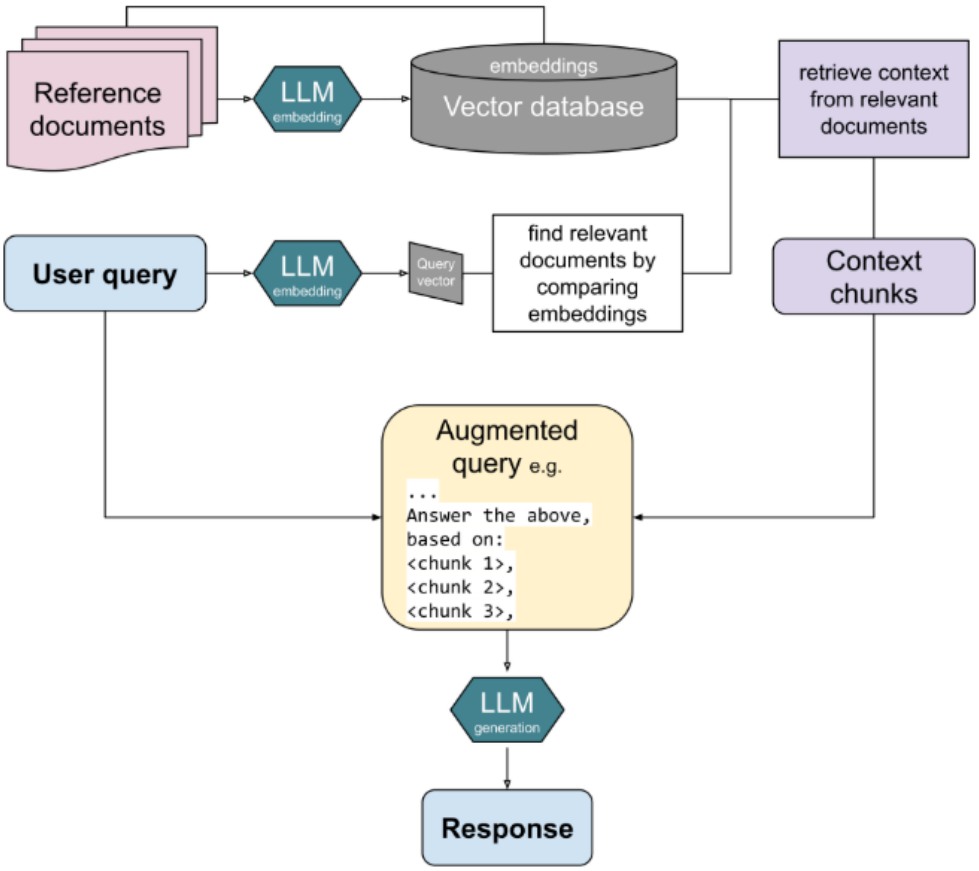

Figure 2.5: Retrieval-Augmented Generation (RAG) architecture (Source: Wikimedia Commons)

Formal Definition

Let q denote a user query and let $D = \{d_1, d_2, \ldots, d_N\}$ represent a corpus of documents.

The retriever function $R(q)$ returns a ranked list of relevant documents based on semantic similarity:

$$R(q) = \text{Top-}k\big(\text{Sim}(q, d_i)\big), \quad d_i \in D$$

The retrieved set is denoted as:

$$D_r = \{d_{r1}, d_{r2}, \ldots, d_{rk}\} \subset D$$

The final output is generated by marginalizing over the retrieved documents:

$$P(y \mid q) \approx \sum_{z \in D_r} P(z \mid q)\, P(y \mid q, z)$$

where:

- q is the input query,
- D_r is the set of retrieved document chunks,
- $P(z \mid q)$ represents the retrieval relevance score,
- $P(y \mid q, z)$ is the generation probability conditioned on both query and retrieved context,
- k is the number of retrieved documents.

Encoder–Decoder Perspective

RAG can also be described using an encoder–decoder framework.

The encoder maps documents into dense vector representations, while the decoder generates the output conditioned on both the query and retrieved context.

Let:

$$e_i = f_{\text{Encoder}}(d_i), \quad e_q = f_{\text{Encoder}}(q)$$

The generation process can be expressed as:

$$y = f_{\text{Decoder}}\big(q, \{e_{d_1}, e_{d_2}, \ldots, e_{d_k}\}\big)$$

This formulation highlights that the decoder conditions its output on both the query and the embeddings of retrieved documents.

Integration with the Proposed System

In this project, RAG plays a central role in ensuring accurate and grounded analysis of tender documents. Each document is segmented into semantic chunks and embedded into a vector space using transformer-based embedding models [18]. During inference, a user query is also embedded and compared against stored vectors to retrieve the most relevant content using FAISS-based similarity search [19].

The retrieved chunks are then provided as input to a Large Language Model, which generates structured and coherent responses conditioned on this context. This design helps reduce hallucinations by constraining generation to retrieved document evidence, improving consistency with the source material.

Consequently, the proposed system leverages RAG as a core architectural component that bridges semantic search and generative reasoning, enabling robust and domain-adapted document intelligence.

2.5.2 Vector Databases and FAISS

Retrieval-Augmented Generation systems rely fundamentally on efficient similarity search over high-dimensional vector spaces. To enable scalable and semantically meaningful retrieval, documents are represented using vector databases, where textual content is transformed into dense embeddings.

Each document chunk d_i is mapped into a continuous vector representation $e_i \in \mathbb{R}^d$ using an encoder function:

$$e_i = f_{\text{Encoder}}(d_i)$$

Similarly, a user query q is encoded into a query embedding $e_q \in \mathbb{R}^d$:

$$e_q = f_{\text{Encoder}}(q)$$

Retrieval is performed by computing similarity scores between the query embedding and document embeddings. The most commonly used metric is cosine similarity:

$$\text{Sim}(e_q, e_i) = \frac{e_q \cdot e_i}{\|e_q\| \, \|e_i\|}$$

In large-scale retrieval systems, computing exact similarity between a query and all document embeddings becomes computationally expensive. To address this limitation, this work employs FAISS (Facebook AI Similarity Search), a high-performance library for efficient similarity search and clustering of dense vectors [19].

FAISS supports multiple indexing strategies such as Flat (exact search), IVF (Inverted File Index), HNSW, and Product Quantization (PQ), enabling a trade-off between accuracy and efficiency.

FAISS relies on Approximate Nearest Neighbor (ANN) search, which significantly reduces computational cost while maintaining high retrieval quality. This makes it suitable for large document collections such as tender repositories.

Relevance to Tender Document Analysis

The integration of FAISS-based retrieval is particularly important in the context of tender document analysis due to the volume and complexity of procurement documents.

This approach provides several advantages:

- **Semantic Retrieval:** captures conceptual similarity beyond exact lexical matching.
- **Scalability:** supports efficient indexing and retrieval over large document corpora.
- **Low-Latency Search:** enables near real-time query processing through ANN search.
- **Improved Decision Support:** ensures that the generative model operates on the most relevant contextual information.

2.5.3 Types of RAG Approaches

This section presents a taxonomy of Retrieval-Augmented Generation (RAG) approaches, highlighting the evolution from basic retrieval pipelines to structured and graph-based frameworks.

Simple Retrieval-Augmented Generation (RAG)

Simple Retrieval-Augmented Generation (RAG) is the foundational form of retrieval-augmented systems, originally introduced by Lewis et al. [12]. It follows a two-stage architecture that combines information retrieval with natural language generation. The system first retrieves relevant information from an external knowledge source and then uses a Large Language Model (LLM) to generate a response based on the retrieved content.

In a typical Simple RAG pipeline, source documents are divided into smaller text chunks and stored in a vector database. Each chunk is converted into a dense vector representation using embedding models such as Sentence-BERT or other transformer-based encoders. When a user submits a query, the query is embedded into the same vector space, and similarity measures such as cosine similarity are used to retrieve the top-k most relevant chunks.

The retrieved chunks are then provided to the LLM as additional context. Based on both the user query and the retrieved information, the model generates a grounded response. This approach is widely used in applications such as question answering, summarization, and document retrieval.

The overall workflow can be represented as:

$$q \rightarrow e_q \rightarrow \text{Top-}k(\text{Sim}(e_q, e_d)) \rightarrow R_k \rightarrow \text{LLM} \rightarrow y$$

where:

- q represents the user query
- e_q represents the query embedding
- e_d represents document chunk embeddings
- R_k represents the retrieved top-k chunks
- y represents the final generated output

Figure 2.6 illustrates the overall Simple RAG pipeline.

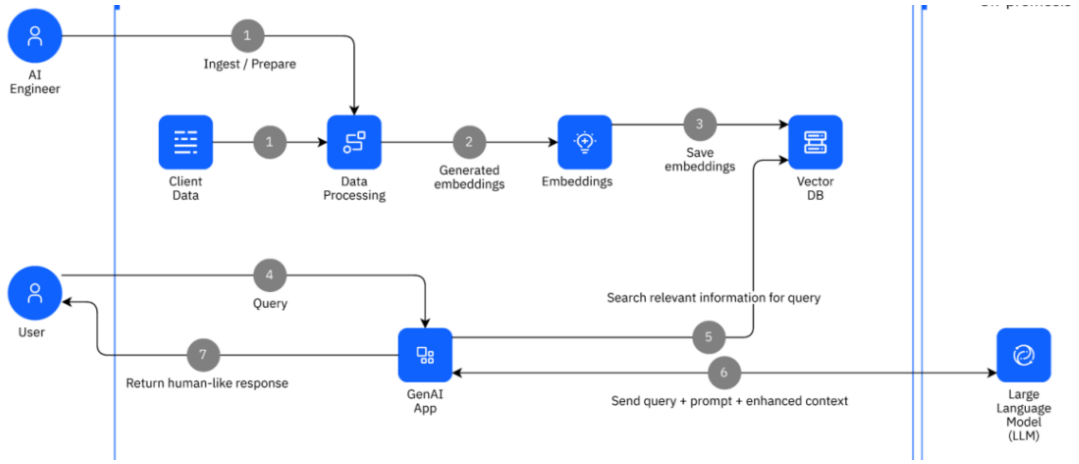

Figure 2.6: Simple RAG pipeline architecture

Although Simple RAG is efficient and easy to implement, it has several limitations. Since retrieval is performed only once, it may struggle with multi-hop reasoning tasks that require combining information from multiple sources. It also does not explicitly model relationships between retrieved chunks or preserve document structure. These limitations motivated the development of more advanced approaches such as GraphRAG, RAPTOR, Structured RAG, and Metadata-Driven RAG.

Graph RAG (Graph-based Retrieval-Augmented Generation)

GraphRAG, a retrieval framework that combines knowledge graphs with RAG [13].Traditional Retrieval-Augmented Generation systems typically retrieve document chunks based primarily on semantic similarity. While this approach performs well for fact-based queries, it often struggles when answering questions that require understanding relationships across multiple documents or identifying broader patterns within large datasets.

To address this limitation, Edge et al. introduced **GraphRAG**, a retrieval framework that combines knowledge graphs with Retrieval-Augmented Generation to improve reasoning over large document collections [13].

The main idea behind GraphRAG is to transform unstructured documents into a structured graph representation. First, documents are divided into smaller chunks that can be efficiently processed by a language model. From each chunk, important entities and relationships are extracted. These extracted elements are then used to construct a knowledge graph defined as:

$$G = (V, E)$$

where:

- V represents entities such as people, organizations, locations, or concepts
- E represents relationships between these entities

Unlike traditional RAG systems that retrieve isolated chunks independently, GraphRAG captures connections between related pieces of information. This enables better handling of complex queries that require multi-hop reasoning or global understanding of the dataset.

After graph construction, GraphRAG applies community detection algorithms (such as Leiden clustering) to group related entities into communities. Each community represents a broader topic or theme within the dataset. The system then generates summaries for these communities, creating hierarchical representations of the underlying information.

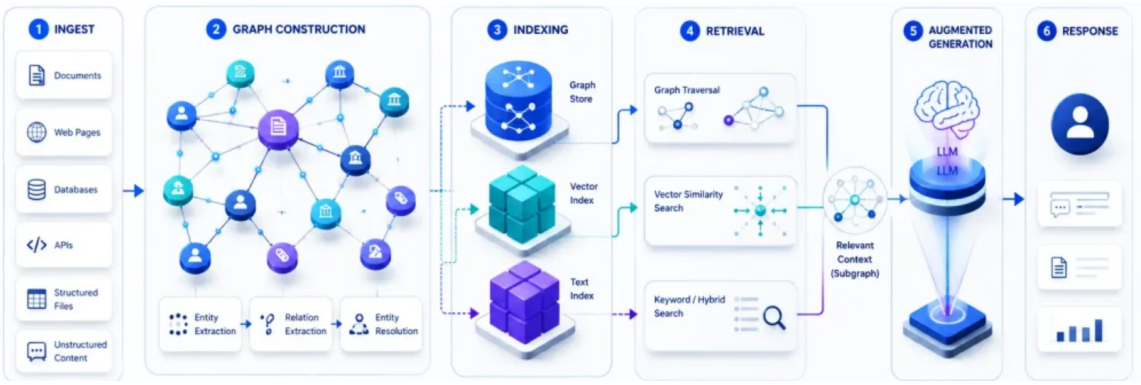

Figure 2.7: Overview of the GraphRAG pipeline

During query processing, the system identifies relevant communities, retrieves associated summaries, and generates partial responses. These responses are then combined to produce the final answer.

Advantages of GraphRAG

- Supports multi-hop reasoning across related entities
- Captures relationships between distributed information
- Improves retrieval for complex analytical queries
- Provides better interpretability through graph structures

Limitations of GraphRAG

- Requires additional preprocessing for graph construction
- Depends on accurate entity and relationship extraction
- More computationally expensive than traditional RAG systems

Summary GraphRAG extends traditional RAG by introducing structured knowledge representation through entity graphs and community-based retrieval. This makes it more effective for complex multi-document reasoning tasks where relationships between information sources are important.

RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval)

Traditional Retrieval-Augmented Generation systems typically retrieve document chunks independently, which limits their ability to capture global context in long and complex documents. This limitation becomes more significant when relevant information is distributed across multiple sections of a document.

To address this issue, Sarthi et al. introduced **RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval)** [14], a hierarchical retrieval framework that organizes documents into a tree structure composed of multiple levels of abstraction.

In RAPTOR, documents are first divided into smaller text chunks, and each chunk is converted into vector embeddings. Semantically similar chunks are then grouped using

clustering techniques, and each cluster is summarized using a language model. This summarization process is applied recursively, producing a hierarchical tree where:

- leaf nodes represent original document chunks,
- intermediate nodes represent summaries of related chunks,
- root nodes represent high-level summaries of larger document sections.

The hierarchical structure can be represented as:

$$T = (N, E)$$

where N represents nodes at different abstraction levels and E represents parent-child relationships between nodes.

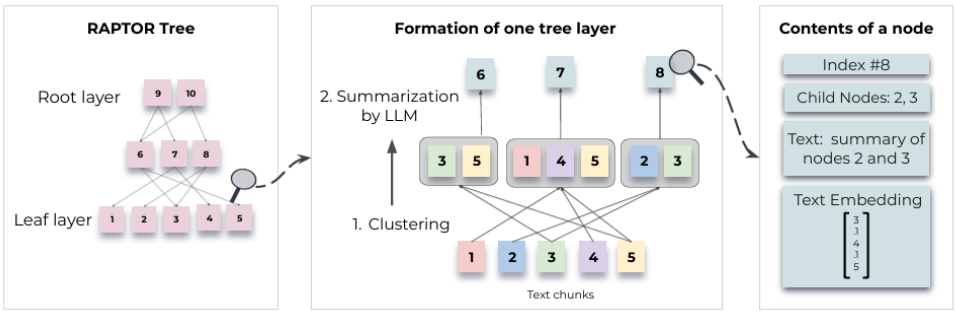

Figure 2.8: Hierarchical tree construction in RAPTOR

During retrieval, RAPTOR supports two different mechanisms.

Hierarchical Tree Traversal: In this approach, retrieval begins at higher-level summary nodes and progressively moves downward toward more detailed child nodes. This allows the system to efficiently narrow the search space by first identifying relevant high-level topics before retrieving fine-grained information.

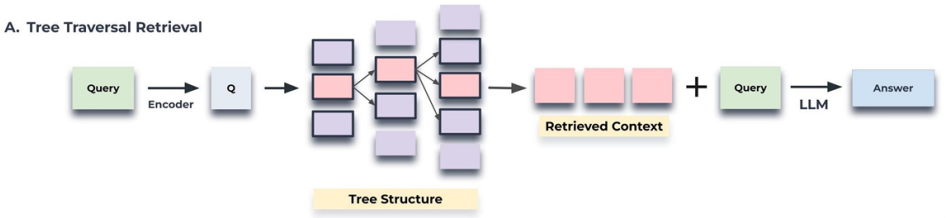

Figure 2.9: Hierarchical tree traversal in RAPTOR

This method improves efficiency and interpretability, but it may miss relevant information located in unrelated branches of the tree.

Collapsed Tree Retrieval: In this approach, all nodes in the hierarchy are flattened into a single retrieval pool. The query is compared against every node regardless of its level, and the most relevant nodes are retrieved directly.

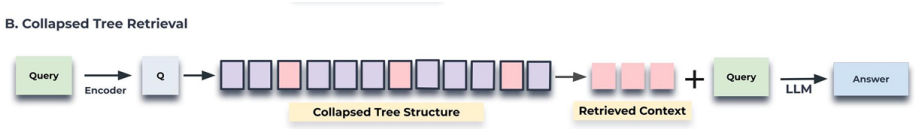

Figure 2.10: Collapsed tree retrieval in RAPTOR

Collapsed retrieval improves flexibility and recall because relevant information can be retrieved from any level of the hierarchy. However, it requires more computation compared to hierarchical traversal.

Compared to traditional RAG, RAPTOR improves long-document understanding by capturing both local details and global context. However, it introduces additional preprocessing overhead, and its performance depends heavily on the quality of clustering and summarization.

Overall, RAPTOR extends traditional RAG by introducing hierarchical retrieval mechanisms that improve performance on long and complex documents.

Structured Retrieval-Augmented Generation (SRAG)

Traditional Retrieval-Augmented Generation (RAG) systems retrieve relevant document chunks and directly provide them to a Large Language Model (LLM) for response generation. While this approach performs well for simple question-answering tasks, it becomes less effective when information is distributed across multiple documents and requires complex reasoning.

Retrieved chunks may contain redundant, incomplete, or noisy information, making it difficult for the LLM to identify relationships between entities and generate accurate responses. To address this limitation, Lin et al. introduced **Structured Retrieval-Augmented Generation (SRAG)**, which introduces an intermediate structuring phase between retrieval and generation [15].

Instead of directly passing raw retrieved text to the LLM, SRAG transforms the retrieved information into structured representations such as relational tables. This allows the model to reason over organized data rather than fragmented text passages.

The SRAG framework consists of three main stages:

- **Retrieval:** Relevant documents or entities are retrieved using semantic search techniques.
- **Structuring:** Important attributes and relationships are extracted from the retrieved content and organized into structured formats such as tables.
- **Generation:** The LLM performs reasoning over the structured representation and generates the final response.

Figure 2.11 illustrates the SRAG framework proposed by Lin et al. [15]. Unlike traditional RAG systems that directly generate responses from raw retrieved text, SRAG introduces a structured reasoning layer that improves answer accuracy for multi-entity queries.

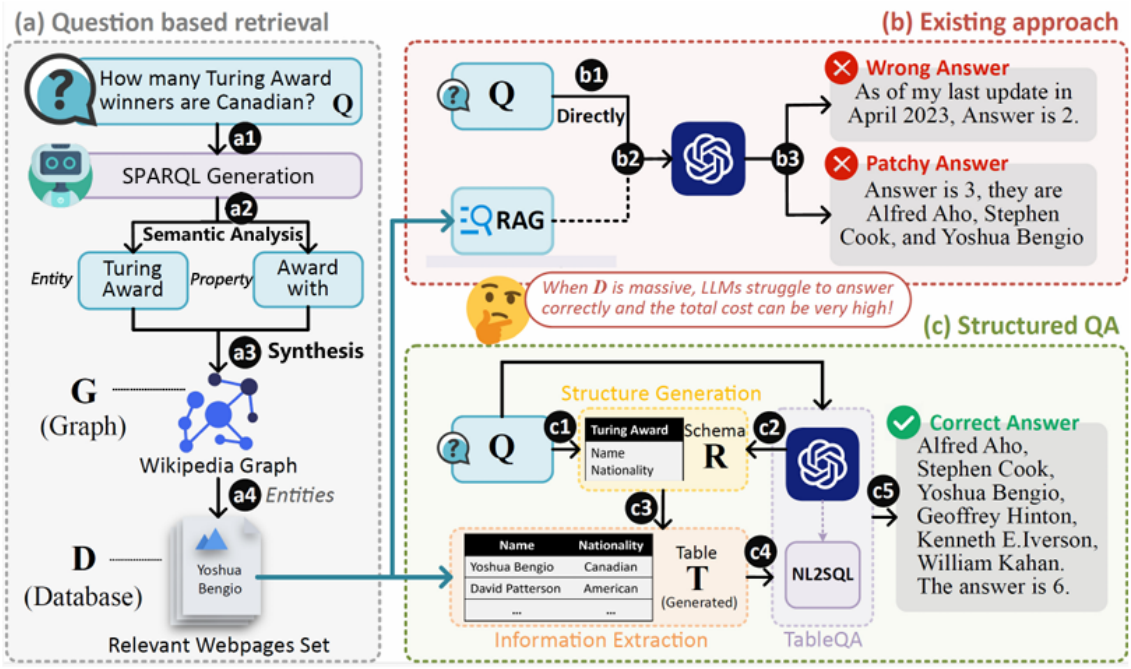

Figure 2.11: Overview of the Structured RAG framework proposed by Lin et al. [15]

For example, instead of retrieving multiple unstructured paragraphs related to suppliers, deadlines, and pricing requirements, SRAG can organize the retrieved information into a structured format such as:

Supplier	Price	Deadline	Requirements
A	\$5000	$Jan\ 20$	$ISOCertification$

This structured representation reduces retrieval noise and improves reasoning efficiency, particularly for analytical queries that require comparing multiple entities.

Advantages

- Improves multi-document reasoning
- Reduces irrelevant retrieved context
- Enhances analytical question answering
- Facilitates structured information processing

Limitations

- Requires additional preprocessing steps
- Schema generation can be challenging
- Increases implementation complexity compared to traditional RAG systems

Summary SRAG extends traditional RAG by introducing a structured reasoning layer between retrieval and generation. This approach is particularly relevant for tender document analysis, where important information such as deadlines, financial requirements, technical specifications, and eligibility criteria is often distributed across multiple sections of a document.

Metadata-Driven Retrieval-Augmented Generation

Traditional RAG systems typically retrieve document chunks based primarily on semantic similarity or keyword matching. While this approach works well for simple datasets, it becomes less effective when dealing with large and highly structured documents such as financial reports, legal contracts, and tender documents.

These documents often contain multiple sections, tables, deadlines, technical requirements, eligibility criteria, and administrative information. When retrieval is performed without considering document structure, the system may return irrelevant chunks or fail to capture important contextual information.

To address this limitation, recent research introduced **Metadata-Driven Retrieval-Augmented Generation**, where additional metadata is generated and incorporated into the retrieval process to improve retrieval precision [16].

Instead of storing only raw text embeddings, each document chunk is enriched with metadata such as:

- document title
- section name
- document category
- publication date
- extracted entities
- keywords
- document summary

This additional metadata enables the retrieval system to better understand document structure and contextual relevance.

1. Document Parsing The system first extracts content from raw documents such as PDFs, Word files, or scanned files while preserving structural elements such as headings, tables, lists, and section hierarchy.

2. Metadata Generation Metadata is generated at both document and chunk levels. Document-level metadata may include:

- project name
- issuing organization
- publication date

- submission deadline
- tender category

Chunk-level metadata may include:

- technical requirements
- financial constraints
- qualification criteria
- extracted entities

3. Metadata-Enriched Indexing Each document chunk d_i is converted into an embedding vector:

$$v_i = f(d_i)$$

Similarly, the query q is represented as:

$$v_q = f(q)$$

Similarity is commonly computed using cosine similarity:

$$sim(v_q, v_i) = \frac{v_q \cdot v_i}{\|v_q\|\|v_i\|}$$

Unlike traditional RAG, metadata filters can also be applied before retrieval, such as filtering by document type, date, category, or specific document sections.

4. Retrieval and Generation The system combines semantic similarity search with metadata filtering to retrieve the most relevant document chunks. The retrieved content is then provided to the LLM to generate grounded responses.

Advantages

- Improves retrieval precision
- Reduces irrelevant results
- Preserves document structure
- Improves performance on long documents
- Supports domain-specific retrieval

Limitations

- Requires additional preprocessing
- Metadata generation can be computationally expensive
- Performance depends on metadata quality

Summary Metadata-Driven RAG extends traditional retrieval by incorporating structural information into the indexing and retrieval process. This approach is particularly relevant for tender document analysis, where accurate retrieval depends on identifying specific sections such as deadlines, financial requirements, and technical specifications.

2.5.4 Comparative Analysis of RAG Approaches

The Retrieval-Augmented Generation approaches discussed in this chapter address different limitations of traditional retrieval systems. While Simple RAG focuses on efficient semantic retrieval, more advanced variants introduce hierarchical reasoning, structured retrieval, graph-based reasoning, and metadata-aware optimization.

Approach	**Main Idea**	**Strengths**	**Limitations**
Simple RAG	Retrieves top-k document chunks using vector similarity search	Simple implementation, fast retrieval, computationally efficient	Limited reasoning ability, weak handling of document structure and relationships
GraphRAG	Builds knowledge graphs from entities and relationships for graph-based retrieval	Supports multi-hop reasoning, captures relationships, improves explainability	High preprocessing cost, requires accurate graph construction
RAPTOR	Organizes documents into hierarchical trees using recursive summarization	Captures both global and local context, effective for long documents	Computationally expensive preprocessing, depends on summary quality
Structured RAG (SRAG)	Uses structured representations such as tables, schemas, or relational formats to organize retrieved information before generation	Reduces retrieval noise, improves analytical and multi-entity reasoning	Requires schema generation and additional processing
Metadata-Driven RAG	Enhances retrieval using metadata such as section labels, dates, categories, and extracted entities	Improves retrieval precision and contextual relevance for structured documents	Metadata generation can be computationally expensive and domain-dependent

Table 2.1: Comparison of different RAG approaches

2.5.5 Synthesis and Positioning of the Proposed Approach

The analysis of existing RAG variants shows that no single approach fully addresses all challenges of structured document understanding. Simple RAG lacks reasoning capabilities, Graph RAG requires complex knowledge graph construction, and RAPTOR

introduces additional preprocessing overhead. Advanced RAG, while highly effective for structured extraction tasks, remains dependent on manually designed queries.

These limitations motivate the development of task-oriented retrieval strategies tailored to domain-specific applications such as tender document analysis. The proposed system builds upon these existing approaches by combining structured query decomposition with efficient semantic retrieval to improve both precision and adaptability.

2.6 Literature Review

Recent advances in Natural Language Processing (NLP) and deep learning have significantly improved the capabilities of document understanding and information extraction systems. Traditional approaches based on rule-based systems and keyword matching have gradually been replaced by neural architectures that leverage contextual embeddings and transformer-based models.

A major research direction focuses on Retrieval-Augmented Generation (RAG), which combines information retrieval with Large Language Models (LLMs). Studies have shown that integrating external knowledge sources during inference improves factual accuracy and reduces hallucination, especially in knowledge-intensive tasks. This has led to the development of several enhanced RAG architectures such as Graph RAG, RAPTOR, and field-oriented retrieval strategies, each addressing specific limitations of standard retrieval pipelines.

Graph-based approaches introduce structured reasoning by modeling documents as knowledge graphs, enabling multi-hop reasoning over entities and relationships. Hierarchical methods such as RAPTOR improve performance on long documents by organizing information into tree-like structures that capture both local and global context. Meanwhile, task-oriented and field-driven retrieval strategies improve precision in structured information extraction by decomposing queries into multiple targeted retrieval tasks.

In parallel, classical information extraction techniques such as Named Entity Recognition (NER), relation extraction, and document-level reasoning models continue to play an important role. However, these methods alone are often insufficient when dealing with large, unstructured, or heterogeneous documents such as calls for tenders.

Overall, the literature highlights that modern document intelligence systems increasingly rely on hybrid architectures that combine retrieval, structured representation, and generative reasoning. This combination is particularly effective for complex domains such as legal and procurement documents, where accuracy, traceability, and completeness are critical requirements.

Chapter 3

System Design and Modeling

3.1 Introduction

This chapter presents the design and modeling phase of the proposed intelligent system for automated tender document analysis. Building upon the theoretical foundations introduced in the previous chapter—particularly Natural Language Processing (NLP), Large Language Models (LLMs), and Retrieval-Augmented Generation (RAG)—this chapter focuses on transforming these concepts into a practical architecture capable of addressing the specific constraints associated with procurement document processing.

Tender documents represent a particularly complex category of unstructured documents due to their large size, heterogeneous formatting, legal terminology, and the distribution of critical information across multiple sections. Important elements such as budgets, submission deadlines, technical requirements, contractual conditions, and evaluation criteria are often embedded within lengthy documents, making manual analysis both time-consuming and error-prone. Additionally, many procurement documents are distributed in scanned PDF formats, which introduces further complexities related to text extraction and document readability.

To address these challenges, several retrieval architectures were explored during the design phase of this project. Traditional RAG architectures were initially evaluated due to their simplicity and strong performance in semantic retrieval tasks. However, they frequently retrieved semantically relevant chunks that lacked the precision required for extracting highly specific procurement attributes such as team composition requirements, financial conditions, and submission deadlines.

More advanced architectures, including GraphRAG, RAPTOR, and Structured Retrieval-Augmented Generation (SRAG), were subsequently studied to assess their suitability for processing long and complex procurement documents. Although these architectures introduced significant improvements in retrieval quality, they also presented limitations when applied to tender analysis. GraphRAG introduced high computational complexity through graph construction processes, while RAPTOR risked losing critical fine-grained details during hierarchical summarization. While SRAG introduced useful concepts for structured retrieval, its complete architecture was not directly adopted. Instead, selected retrieval optimization mechanisms were integrated into the proposed solution.

Based on these observations, a specialized architecture named **Advanced RAG** was designed as the final solution of this project. This architecture remains fundamentally based on a traditional RAG pipeline while incorporating retrieval optimization mechanisms inspired by metadata-driven retrieval and SRAG approaches. The resulting Advanced RAG architecture was specifically adapted for procurement document analysis through field-oriented semantic retrieval, OCR-enhanced document processing, and structured extraction generation. Unlike traditional RAG systems that retrieve generic docu-

ment chunks, the proposed approach performs targeted retrieval for specific procurement fields such as budget, project duration, payment conditions, service-level agreements, technical references, and submission deadlines.

The chapter begins with a detailed analysis of system requirements. It then presents the evaluation of the different retrieval architectures explored during the design phase before introducing the proposed Advanced RAG architecture. Finally, the chapter presents the system modeling process and the final architecture adopted for implementation.

3.2 Evaluation of Existing Retrieval Architectures

After identifying the system requirements, multiple retrieval architectures were evaluated to determine the most suitable approach for automated tender document analysis. Although these architectures were introduced theoretically in Chapter 2, this section focuses on their practical suitability for the specific constraints of procurement documents.

The evaluation was conducted based on several criteria:

- Retrieval precision
- Ability to process long documents
- Preservation of fine-grained information
- Computational complexity
- Adaptability to procurement-specific extraction tasks

3.2.1 Traditional RAG

The first architecture implemented during the experimentation phase was a traditional Retrieval-Augmented Generation (RAG) pipeline designed for structured extraction from procurement documents. This approach was based on a straightforward semantic retrieval mechanism combining document chunking, dense embeddings, and vector similarity search.

The implementation workflow consisted of the following steps:

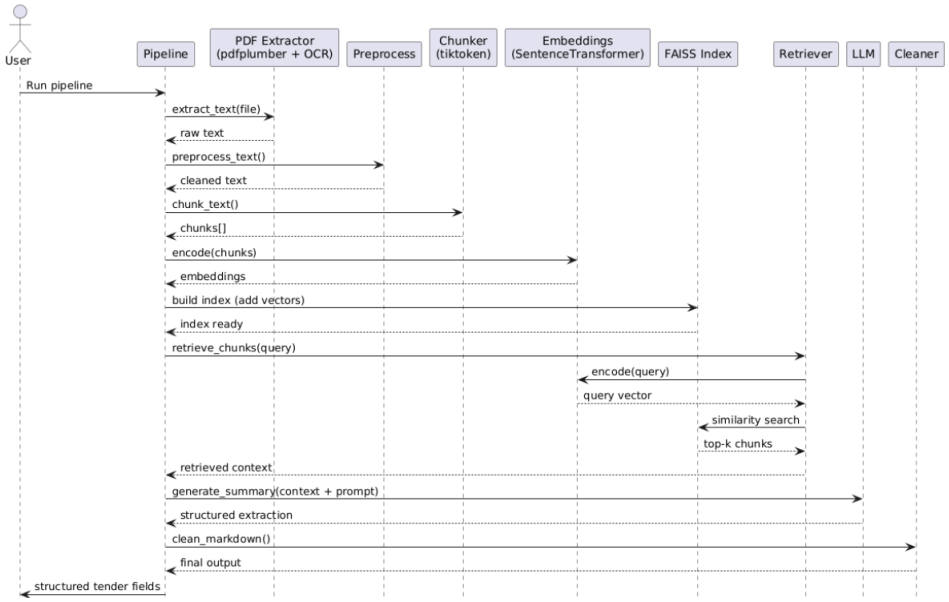

Figure 3.1: Graph RAG results

This architecture provided a strong and efficient baseline due to its simplicity and fast retrieval pipeline. The use of FAISS enabled scalable similarity search over document embeddings, while dense vector representations ensured reasonable semantic matching between queries and document content.


```
{
  "objet": "Acquisition d'une nouvelle baie de stockage pour l'Institut Paris Region",
  "budget": "Non spécifié dans le document",
  "duree": "La livraison du matériel retenu et son installation physique devraient être effectuées dans la seconde quinzaine du mois de mars 2024.",
  "modalites_de_paiement": "Non spécifié dans le document",
  "equipe_projet": "L'équipe technique de l'Institut Paris Region sera impliquée dans le transfert de compétences et la formation.",
  "qualification": "Les candidats devront proposer une solution de stockage qui répond aux besoins de l'Institut Paris Region,
                    notamment en termes de performances, de capacités et de sécurité.",
  "penalites": "Les candidats devront respecter les délais de livraison et de mise en service, sous peine de pénalités.",
  "SLA": "Les candidats devront proposer un engagement de prix de la maintenance sur 10 ans et une garantie \"Constructeur\"
          pour l'ensemble des matériels fournis.",
  "references": "Les candidats devront fournir des références de clients satisfaits.",
  "cautionnement": "Non spécifié dans le document",
  "grille_de_notation": "La valeur technique de l'offre (35%), le prix (35%), la garantie, l'évolutivité et le contrat de support/maintenance (15%),
                         les conditions d'exécution du projet (10%) et les livrables (5%) seront pris en compte pour évaluer les offres.",
  "modele_de_cv": "Non spécifié dans le document",
  "offre_financiere": "Les candidats devront proposer une offre financière détaillée, incluant les coûts des matériels, des logiciels,
                       de la maintenance et des prestations de support.",
  "visite_des_lieux": "Non spécifié dans le document",
  "date_de_remise": "Les documents contenant les candidatures et les offres devront être envoyés par messagerie électronique
                     le vendredi 19 janvier 2024 à 17h00 au plus tard."
}
```


Figure 3.2: Graph RAG results

The system achieved **good overall extraction quality**, as demonstrated in the experimental results. It was able to correctly identify most procurement fields such as *object of the contract, duration, penalties, SLA conditions, evaluation criteria, references, and financial offer structure*. The final outputs were coherent and well-structured, confirming the effectiveness of semantic retrieval combined with LLM-based extraction.

However, performance analysis revealed several important limitations that affected alignment with the project objectives, particularly for high-precision and structured fields:

- **Budget extraction remained incomplete or vague**, often returning “Non spécifié dans le document” even when financial indicators were implicitly present in the text,

- **Team-related information was inconsistently captured**„
- retrieval was still purely semantic and context-independent, leading to occasional selection of thematically similar but irrelevant chunks,
- important dispersed information was sometimes missed due to the lack of global document reasoning.

Overall, while traditional RAG provided a solid baseline and demonstrated the feasibility of automated procurement extraction, its limitations in handling implicit financial information and distributed organizational details (such as team composition and structured budget fields) motivated the need for more advanced retrieval strategies and structure-aware enhancements in later iterations.

3.2.2 GraphRAG

In this work, a Graph-based Retrieval-Augmented Generation (GraphRAG-inspired) approach was implemented to improve document understanding by explicitly modeling semantic relationships between text chunks.

Unlike classical GraphRAG architectures that rely on hierarchical clustering, dimensionality reduction, and recursive summarization, the implemented system adopts a lighter and more practical graph construction strategy, designed to remain computationally feasible for real-world tender document processing.

The pipeline includes the following steps:

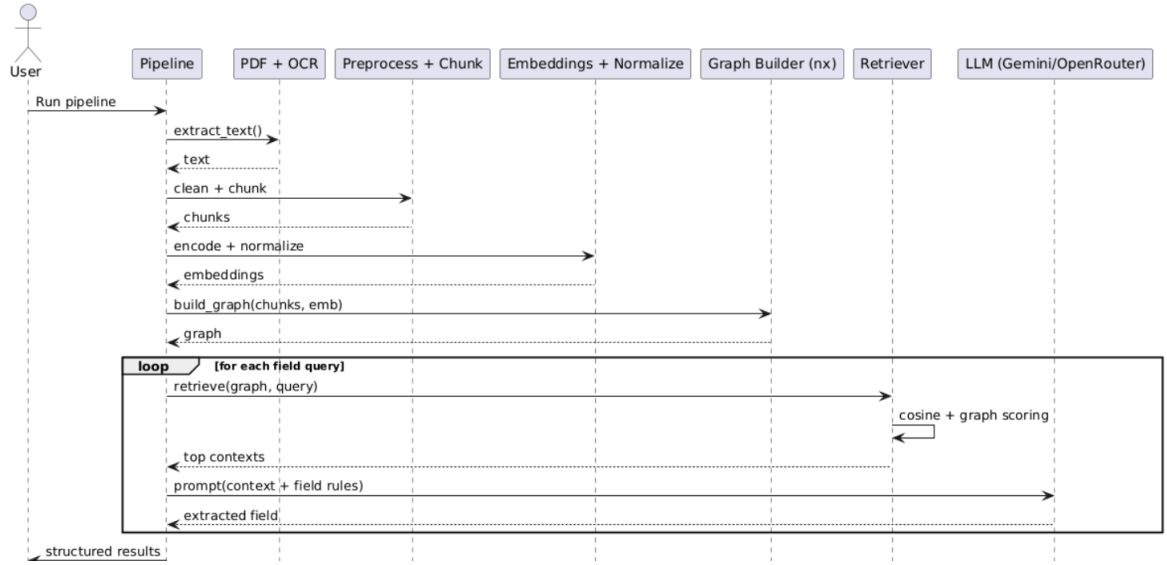

Figure 3.3: Graph RAG Pipeline

This design enables the system to leverage both semantic similarity and local graph connectivity, leading to improved retrieval quality compared to a pure vector-based search approach.

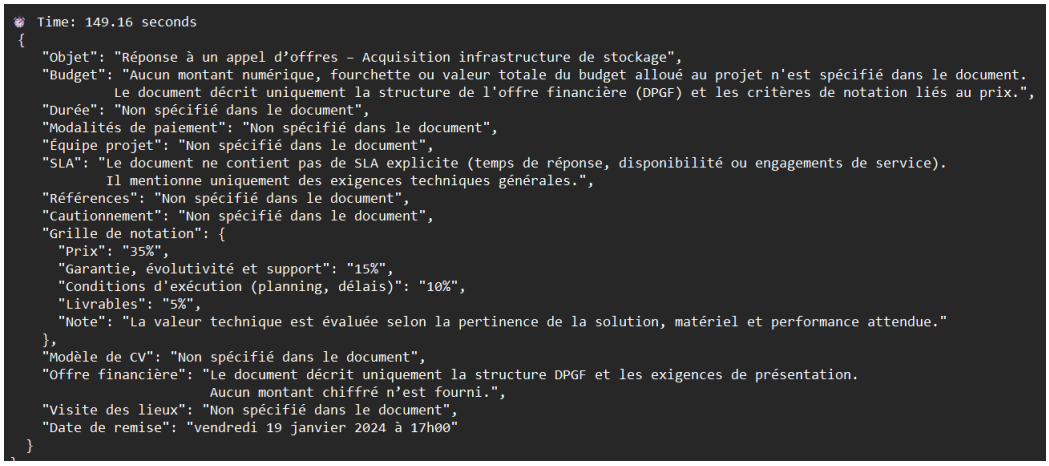

Figure 3.4: Graph RAG results

The system is able to generate structured outputs covering key contractual information. However, during experimentation, it was observed that some important fields were not consistently or correctly extracted. In particular, missing or incomplete extractions included:

- Timeline, payement, team, SLA, financial offer (which was often missing or not properly structured).

The *Budget* field was partially extracted but frequently remained descriptive rather than strictly numerical or structured.

From the executed experiment , the system produced a structured JSON output in approximately 149.16 seconds. This relatively high execution time is mainly due to multiple LLM calls, since each field is processed independently, as well as repeated retrieval operations for each query.

Additional limitations observed during testing include:

- sensitivity to context sparsity, where relevant information is scattered or not sufficiently represented in retrieved chunks,
- high latency caused by multiple sequential LLM calls (one per field),
- incomplete global document reasoning despite graph-based enhancements.

Although this Graph-based retrieval approach improves contextual matching compared to standard chunk-based RAG systems, it remains a simplified GraphRAG variant. It does not include hierarchical clustering (UMAP/GMM) or recursive summarization mechanisms typically associated with full GraphRAG pipelines.

As a result, while the method enhances semantic retrieval and structural awareness, it still suffers from high latency, partial extraction of key contractual elements (including *offre financière*), and limited global reasoning capabilities. Consequently, despite its improvements in retrieval quality, this approach remains computationally expensive and not fully suitable for real-time tender analysis systems.

3.2.3 RAPTOR with Hierarchical Traversal

The first RAPTOR variant I implemented in this project was based on hierarchical tree traversal. The main idea of this approach was to recursively organize document chunks into semantic clusters using embedding similarity combined with Gaussian Mixture Models (GMM). After clustering, each group of chunks was summarized using a Large Language Model (LLM), which created several abstraction levels inside a hierarchical tree structure. During the retrieval phase, the system traversed the tree from parent nodes to the most relevant child nodes in order to retrieve contextual information related to the query.

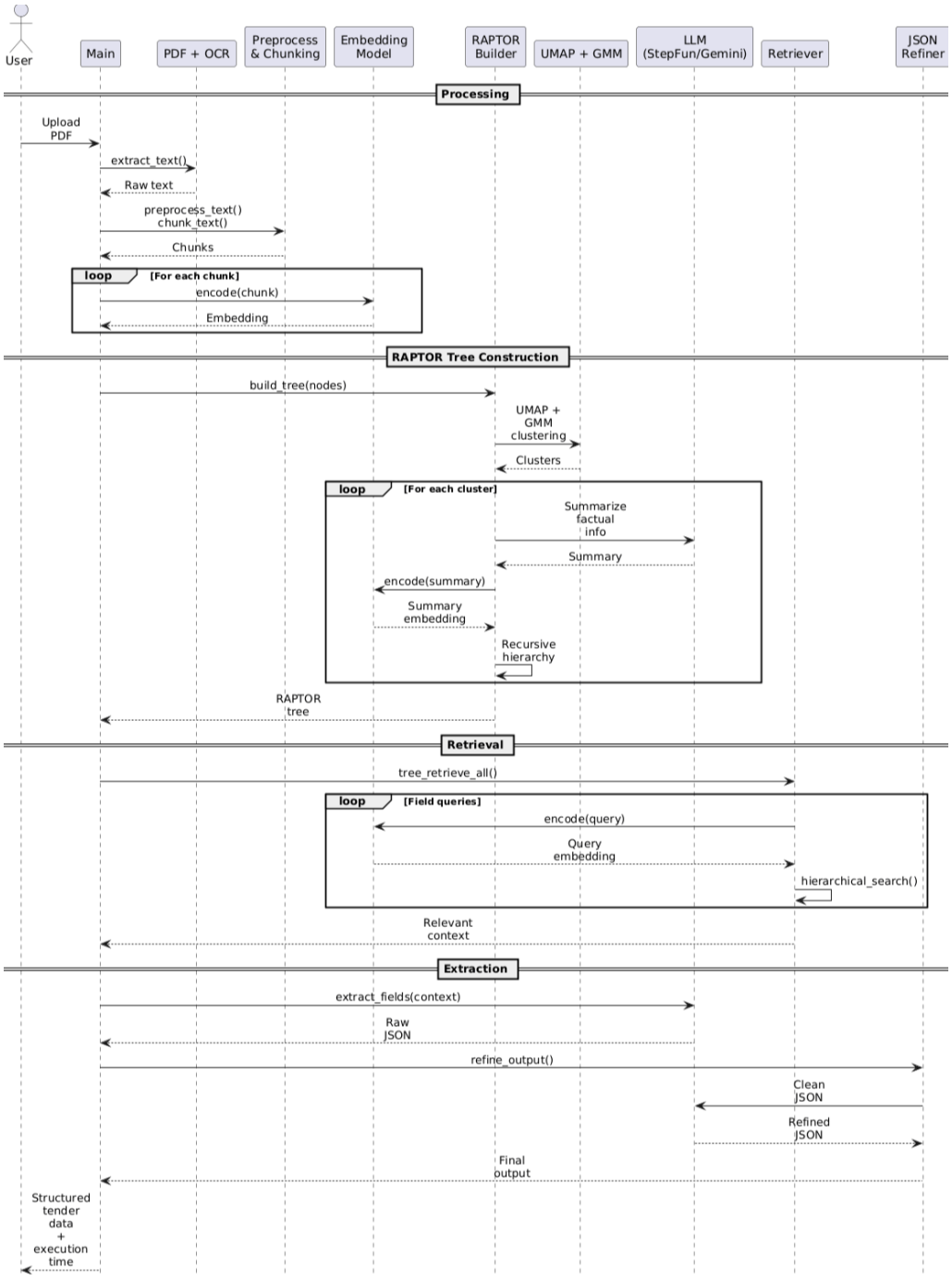

Figure 3.5: RAPTOR Tree Traversal Retrieval RAG Pipeline

Compared to traditional flat RAG systems, this architecture improved retrieval precision because irrelevant information was gradually filtered during the traversal process. In addition, the combination of semantic clustering and recursive summarization helped the system identify relationships between different contractual sections, including SLA conditions, financial requirements, evaluation criteria, and maintenance clauses.

To evaluate the effectiveness of this approach, I experimented with several LLMs for the summarization stages. Among all the tested models, StepFun StepFun and Google Gemini generated the most coherent and well-structured outputs. Other models produced less accurate extractions and weaker contextual summaries, especially when processing technical and contractual information.

The system was able to generate structured JSON outputs containing:


```
⏱ Time: 139s
{
  "objet": "Acquisition d'une baie de stockage pour datacenter",
  "budget": "Non spécifié dans le document. Le DPGF (Décomposition du Prix Global et Forfaitaire)
             est à compléter par les candidats avec leurs prix.",
  "duree": {
    "livraison": "Seconde quinzaine de mars 2024",
    "mise_en_service_tests": "Début du second trimestre 2024",
    "demenagement": "Fin 2024"
  },
  "paiement": "Marché sous forme de bon de commande. Prestations complémentaires (maintenance évolutive, assistance)
               feront l'objet de bons de commande spécifiques.",
  "equipe": {
    "contact_technique": {
      "nom": "Jean-Frederic LASSARA",
      "service": "Département Systèmes Information, L'Institut Paris Region",
      "telephone": "01.77.49.76.24",
      "mobile": "07.63.95.62.74",
      "email": "jean-frederic.lassara@institutparisregion.fr"
    },
    "contact_administratif": {
      "nom": "Stéphane MAIN",
      "service": "Service Achats, L'Institut Paris Region",
      "telephone": "01.77.49.75.58",
      "email": "stephane.main@institutparisregion.fr"
    },
    "equipe_interne": "Équipe technique IPR à former (2 à 3 personnes). DSI de l'IPR comme maitre d'ouvrage."
  },
  "SLA": {
    "disponibilite": "99,999% annuel en heures ouvrées",
    "latence": "< 2ms avec réduction de données (déduplication/compression)",
    "support_telephonique": "8h30-18h30, 5 jours sur 7 (jours ouvrés)",
    "delai_dossier_recette": "Une semaine maximum après installation"
  },
  "references": "Non spécifié dans le document",
  "cautionnement": "Non spécifié dans le document",
  "notation": "Critères d'attribution : 1. Valeur technique (35%) 2. Prix (35%) 3. Garantie/évolutivité/contrat support (15%)
               4. Conditions d'exécution (planning/délais) (10%) 5. Livrables (5%). Clarté et complétude de l'offre financière importante.
               Droit de refuser l'offre la moins chère et d'accepter défauts non substantiels.",
  "structure_DPGF": [
    "Solution de stockage (Matériels, Matériels - Site de PRA, Licences Logiciels)",
    "Maintenance 5 ans sur site pour l'ensemble des matériels",
    "Maintenance 1 an supplémentaire sur site pour l'ensemble des matériels",
    "Support 60 mois sur site 5j/7",
    "Prestation de service (intégration, installation, transfert de compétences, rédactionnel)",
    "Prestation de service complémentaire - coût journalier",
    "Coût total du projet"
  ],
  "visite": "Non spécifié dans le document",
  "date_limite": "Vendredi 19 janvier 2024 à 17h00"
}
```


Figure 3.6: RAPTOR Tree Traversal Retrieval RAG results

The extraction quality was generally satisfactory, especially for complex tender documents containing heterogeneous sections.

However, despite the improvement in retrieval quality, several important limitations were identified during experimentation:

- high preprocessing time caused by recursive clustering and embedding generation,
- significant overhead from repeated LLM summarization at each hierarchy level,
- increased response latency during traversal and refinement,

- occasional loss of fine-grained contractual details due to aggressive summarization.

Performance evaluation showed that the complete pipeline required more than 130 seconds to process a single tender document in some experiments. The execution illustrated in Figure that contains the results, obtained using the StepFun model, required 139 seconds, while Gemini produced similar latency and comparable extraction quality. Since one of the primary objectives of the solution was to provide fast and practical processing of tender documents, this execution time was considered too high for real-world deployment.

Although RAPTOR with hierarchical traversal improved contextual relevance compared to traditional RAG approaches, its computational cost and response latency made it unsuitable as the final architecture of the proposed system. For this reason, a lighter and faster retrieval strategy was later adopted in order to achieve a better balance between extraction quality and execution efficiency.

3.2.4 RAPTOR with Collapsed Retrieval

The second RAPTOR variant I implemented was based on collapsed retrieval. Unlike the hierarchical traversal approach, this method flattened all nodes of the RAPTOR tree into a single retrieval space. This allowed the system to retrieve information from all hierarchy levels at the same time, including leaf nodes, intermediate summaries, and root summaries.

The main objective of this approach was to improve retrieval flexibility and increase recall by enabling access to information from different abstraction levels without following a strict hierarchical traversal. To perform retrieval, I used Meta FAISS vector indexing combined with semantic embeddings generated using the all-MiniLM-L6-v2 model. After retrieval, the contexts were expanded using parent-node information and reranked using the BGE reranker model before being passed to the LLM for structured information extraction.

I conducted several experiments using different LLMs during the extraction phase. Among the tested models, StepFun StepFun and Google Gemini produced the best results. However, despite their better performance compared to the other evaluated models, the overall extraction quality remained unsatisfactory. In many cases, the generated JSON outputs contained missing fields, incomplete contractual details, and repeated “Non spécifié” values, even when the required information was present in the document. This indicated that the retrieval process introduced too much irrelevant or duplicated context, which negatively affected the extraction quality.

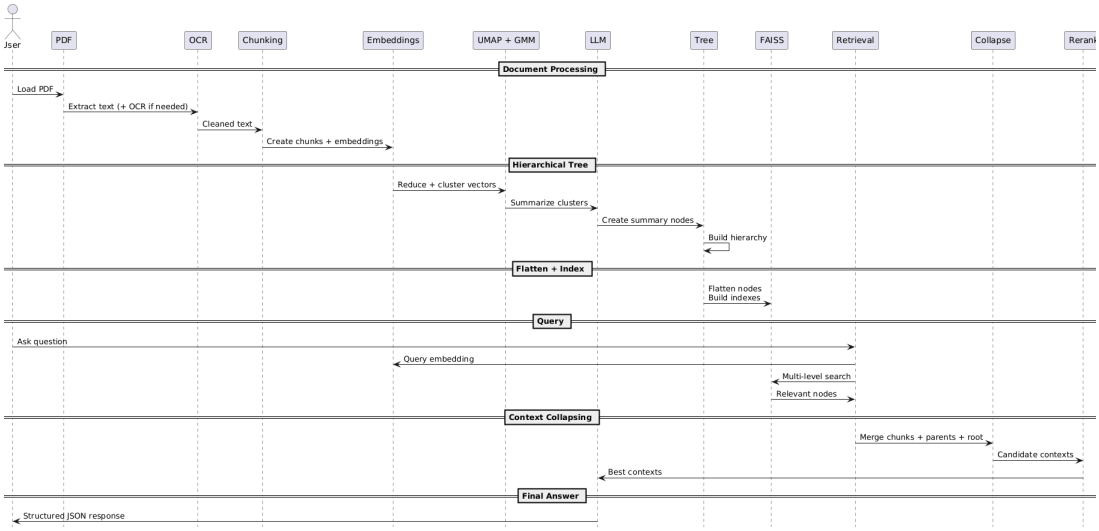

Figure 3.7: RAPTOR Collapsed Tree Retrieval RAG Pipeline

The experiments also highlighted several limitations of this approach:


```
Time: 39.2 seconds

{
  "objet": "Acquisition d'une baie de stockage pour plan de reprise informatique (PRI)",
  "budget": "Non spécifié",
  "duree": "Non spécifié",
  "paiement": "Non spécifié",
  "equipe": "Non spécifié",
  "SLA": "Disponibilité minimale 99,99% en heures ouvrées",
  "references": "Non spécifié",
  "cautionnement": "Non spécifié",
  "notation": "Valeur technique (35%), prix (35%), garantie/évolutivité/support (15%), conditions d'exécution (10%)",
  "offre_financiere": "DPGF détaillée avec tarifs journaliers pour prestations complémentaires",
  "visite": "Non spécifié",
  "date_de_remise": "Non spécifié",
  "type_projet": "Infrastructure de stockage pour plan de reprise informatique"
}
```


Figure 3.8: RAPTOR Collapsed Tree Retrieval RAG results

- retrieval introduced a large amount of noisy information,
- redundant contexts were frequently retrieved from multiple hierarchy levels,
- preprocessing remained computationally expensive due to recursive clustering and summarization,
- response time was significantly higher than traditional RAG systems,
- extraction performance for technical and contractual information was inconsistent.

Although collapsed retrieval improved recall by allowing the system to access information from all hierarchy levels, it also reduced retrieval precision because many irrelevant contexts were retrieved simultaneously. In addition, the complete pipeline required around 40 seconds to process a single tender document, even when using StepFun or Gemini. The other tested LLMs produced even weaker extraction results and lower output quality.

Since one of the main objectives of this project was to develop a fast and reliable tender analysis solution, I did not select this RAPTOR variant as the final architecture.

Despite its ability to retrieve broader contextual information, its high computational cost, slow response time, and poor extraction performance made it unsuitable for practical deployment.

3.2.5 Structured Retrieval-Augmented Generation (SRAG)

Structured Retrieval-Augmented Generation was studied as an alternative approach for improving retrieval precision.

Instead of directly passing retrieved chunks to the language model, SRAG introduces an intermediate structuring phase that organizes retrieved information before generation.

During experimentation, several SRAG concepts were integrated into prototype implementations:

- field-specific retrieval
- retrieval filtering
- structured output generation

These mechanisms improved extraction precision for procurement fields.

However, fully implementing SRAG introduced additional complexity related to schema design and intermediate structuring processes.

Rather than adopting the complete SRAG architecture, selected retrieval optimization concepts were reused in the final system design.

3.2.6 Comparative Analysis and Final Architectural Decision

The experimentation phase demonstrated that no existing architecture fully satisfied the requirements of tender document analysis.

Traditional RAG lacked precision, GraphRAG introduced excessive complexity, and RAPTOR sometimes lost important fine-grained information. SRAG and metadata-driven retrieval provided the most relevant ideas but required adaptation.

Based on these findings, a new architecture named **Advanced RAG** was designed. This architecture remains fundamentally based on a traditional RAG pipeline while incorporating retrieval optimization mechanisms inspired by metadata-driven retrieval and SRAG approaches. The resulting Advanced RAG architecture was specifically adapted for procurement document analysis through field-oriented semantic retrieval, OCR-enhanced document processing, and structured extraction generation.

The next section presents the design of this final architecture.

3.3 Requirements Analysis

As discussed in Chapter 1, the manual analysis of tender documents presents several challenges for organizations. These documents often contain large amounts of information related to administrative requirements, technical specifications, financial constraints, contractual obligations, and submission procedures. Since this information is usually distributed across multiple pages and sections, identifying the most relevant elements manually requires significant time and effort.

Another major challenge is the diversity of document formats. Some tenders are provided as structured digital PDFs, while others are scanned documents that require OCR before any processing can be performed. In addition, procurement documents often contain complex terminology and lengthy descriptions that make traditional keyword-based search insufficient.

Based on these challenges, the proposed system must be able to automatically process tender documents and extract relevant information in a structured manner. The main requirements identified during the design phase are presented below.

3.3.1 Functional Requirements

The system must provide the following functionalities:

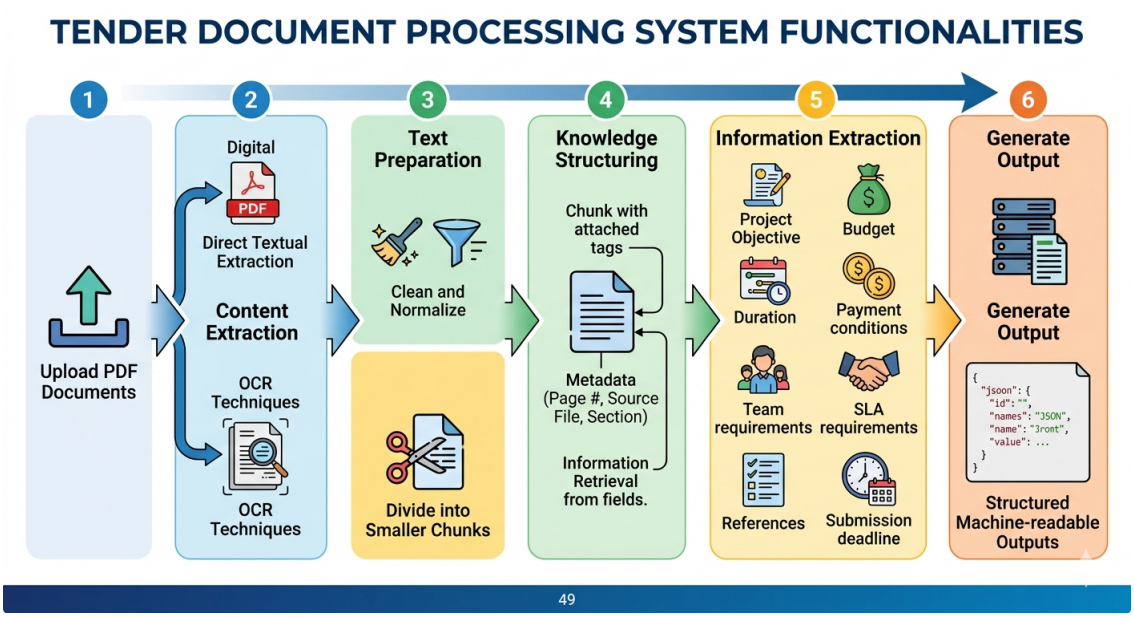

Figure 3.9: Overview of the proposed Advanced RAG architecture

3.3.2 Non-Functional Requirements

Beyond functional requirements, the system must also satisfy several performance constraints:

- **Accuracy:** The extracted information should be as precise as possible.
- **Speed:** The system should significantly reduce the time required for document analysis compared to manual review.
- **Scalability:** The system should be capable of processing long procurement documents efficiently.
- **Reliability:** The system should produce consistent results across different document types.
- **Maintainability:** The architecture should remain modular and easy to update or extend.

- **Flexibility:** The system should adapt to different procurement document structures and formats.

3.3.3 Design Constraints

During experimentation, several constraints were identified that influenced the final architecture design:

- Some documents contained scanned pages requiring OCR.
- Important information was often distributed across multiple sections of the same document.
- Traditional RAG approaches sometimes retrieved irrelevant chunks that reduced extraction precision.
- Advanced retrieval architectures such as GraphRAG introduced significant computational overhead during graph construction and traversal.
- RAPTOR-based summarization approaches sometimes caused the loss of fine-grained details such as exact deadlines or financial values.
- The final solution needed to remain computationally efficient while improving retrieval precision.

These functional requirements, non-functional requirements, and architectural constraints directly influenced the design decisions presented in the following sections and ultimately led to the development of the proposed Advanced RAG architecture.

3.4 Proposed Advanced RAG Architecture

3.4.1 Overview of the Proposed Architecture

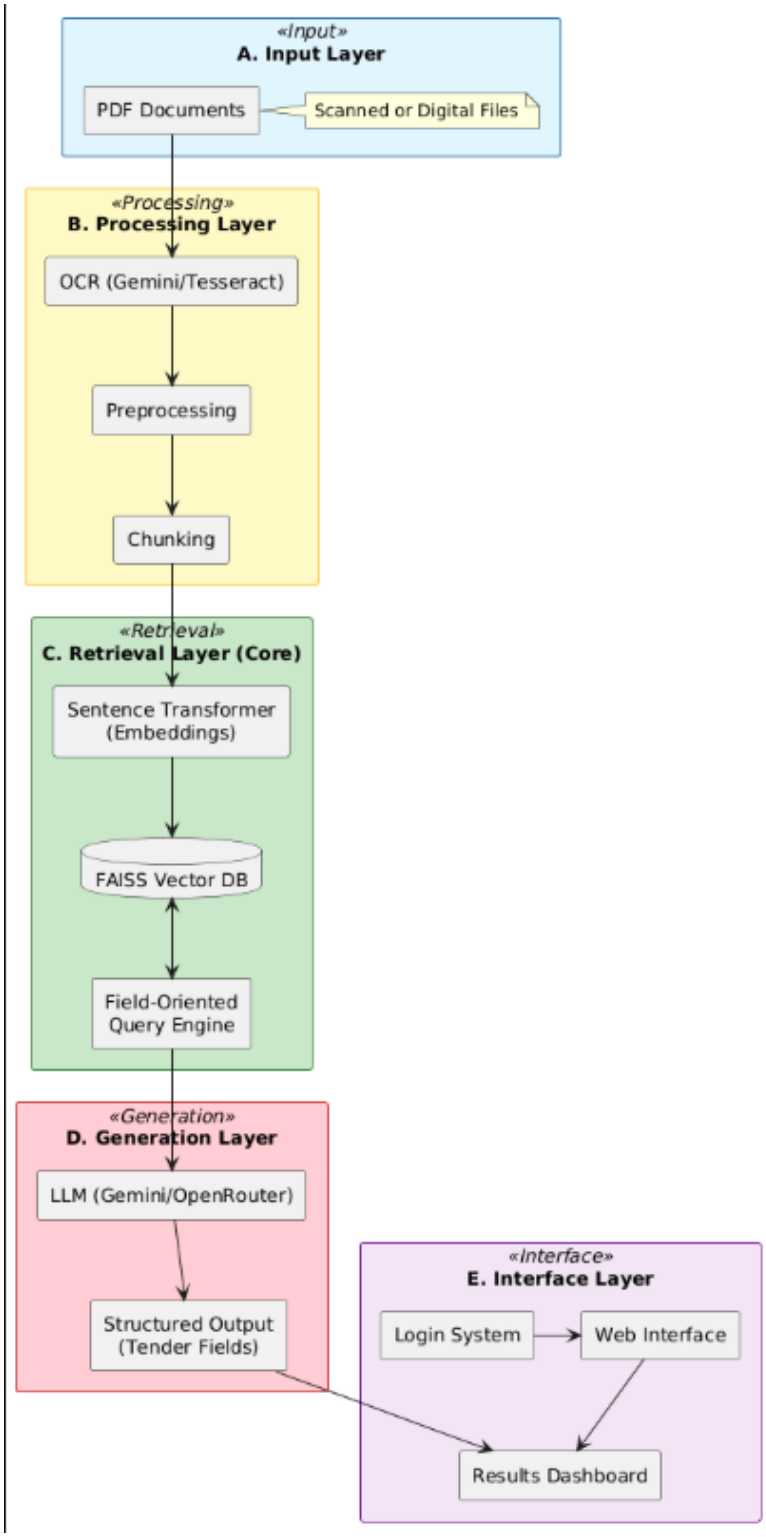

Figure 3.10: The proposed Advanced RAG architecture

The proposed Field-Driven Retrieval-Augmented Generation (RAG) architecture has been designed to address the limitations identified in existing retrieval approaches when applied to the analysis of complex procurement documents. In such documents, relevant

information is often dispersed across multiple sections and expressed in heterogeneous formats, which reduces the effectiveness of conventional semantic retrieval techniques.

To overcome these limitations, the proposed architecture introduces a field-oriented retrieval strategy. Rather than relying on a single generic query, the document is analyzed according to a predefined set of procurement-related fields, including budget, duration, technical specifications, service-level agreements, and submission deadlines. This design choice enables more targeted retrieval and improves the precision of the extracted information.

The overall pipeline integrates several stages, namely OCR-based document processing, text preprocessing and chunking, embedding-based vector representation, semantic retrieval using similarity search, and structured generation using a Large Language Model.

3.4.2 OCR-Based Document Processing

Tender documents are commonly provided in different formats, including digitally generated PDFs and scanned documents. This heterogeneity necessitates the integration of an Optical Character Recognition (OCR) step within the processing pipeline.

Initially, textual content is extracted directly from digital PDF files using standard parsing techniques. In cases where the document consists of scanned images or when the extracted text is incomplete, OCR is applied to convert the visual content into machine-readable text. This step ensures that all relevant information is captured regardless of the document's original format.

3.4.3 Text Preprocessing and Chunking

Once the textual content has been extracted, it undergoes a preprocessing phase aimed at improving its quality and consistency. This phase includes text normalization, removal of unnecessary whitespace, and elimination of structural artifacts such as page identifiers.

The cleaned text is then segmented into smaller overlapping chunks. This chunking strategy is essential to preserve semantic continuity while enabling efficient retrieval. The use of overlap between consecutive chunks helps mitigate the risk of losing relevant contextual information at segment boundaries, which is particularly important in long and structured documents such as tenders.

3.4.4 Metadata Enrichment

To enhance the effectiveness of retrieval, each text chunk is enriched with additional contextual metadata. This metadata typically includes information such as the source document, page number, and, when available, structural indicators derived from the document layout.

Although not used as a strict filtering mechanism in the final implementation, this enrichment process is inspired by metadata-driven retrieval approaches. It contributes to improving traceability and contextual relevance during the retrieval phase.

3.4.5 Embedding Generation and Vector Indexing

Each text chunk is transformed into a dense vector representation using a pre-trained Sentence Transformer model. Formally, a document chunk d_i is mapped into an embedding vector as:

$$\mathbf{v}_i = f(d_i)$$

Similarly, a user query q is transformed into:

$$\mathbf{v}_q = f(q)$$

These embeddings capture the semantic meaning of the text and enable similarity-based comparison between queries and document segments.

The resulting vectors are stored in a FAISS index, which provides efficient nearest-neighbor search capabilities based on cosine similarity. The similarity between a query and a document chunk is computed as:

$$\text{sim}(\mathbf{v}_q, \mathbf{v}_i) = \frac{\mathbf{v}_q \cdot \mathbf{v}_i}{\|\mathbf{v}_q\| \, \|\mathbf{v}_i\|}$$

This vector-based indexing constitutes the core retrieval mechanism of the system, enabling fast and scalable access to relevant information.

3.4.6 Field-Oriented Semantic Retrieval

Unlike traditional RAG systems that rely on a single global query, the proposed architecture adopts a field-oriented retrieval strategy. A predefined set of procurement-related fields is used to guide the retrieval process.

For each field, a specific semantic query is formulated in order to retrieve the most relevant document chunks from the vector database. This multi-query approach ensures a more comprehensive coverage of the document content and reduces the risk of omitting critical information that may be distributed across different sections.

3.4.7 Structured Extraction and Response Generation

Following retrieval, the selected document chunks are aggregated and provided as input to a Large Language Model. The model is guided through a carefully designed prompt that enforces a structured extraction of relevant information.

The generated output is organized according to predefined procurement categories, such as budget, duration, payment conditions, and evaluation criteria. The model is explicitly constrained to rely solely on the provided context, thereby reducing the risk of hallucination and ensuring factual consistency.

This stage transforms unstructured textual content into a structured representation that is more suitable for analysis and decision-making purposes.

3.4.8 Advantages of the Proposed Architecture

The proposed Advanced RAG architecture presents several advantages over conventional retrieval approaches.

First, it enhances retrieval precision by introducing field-specific semantic queries instead of relying on a single generic query. Second, it improves robustness through the integration of OCR, enabling the processing of both scanned and digital documents. Third, it reduces information loss by employing overlapping chunking and structured extraction mechanisms.

Furthermore, the architecture improves interpretability by producing structured outputs aligned with procurement requirements. Finally, it remains computationally efficient compared to more complex retrieval frameworks such as GraphRAG or hierarchical RAPTOR models, making it suitable for practical deployment in real-world scenarios.

3.5 Web Platform Architecture

In addition to the proposed Field-Driven Retrieval-Augmented Generation (RAG) pipeline, a complete web platform was developed to operationalize the system and provide an accessible interface for end users. This platform constitutes the main entry point of the application, enabling users to interact with the AI pipeline through document upload, authentication, and visualization of extracted structured results.

The web application is implemented using the Django framework, following the Model–View–Template (MVT) architectural pattern. The frontend layer is developed using standard HTML and CSS technologies, ensuring a lightweight, responsive, and maintainable user interface. The backend is responsible for handling user requests, orchestrating the execution of the AI processing pipeline, and returning structured outputs to the presentation layer.

Overall, the platform is designed as a multi-layered system that integrates user interaction, backend logic, and external AI services into a unified and coherent workflow. Figure 3.11 illustrates the complete operational flow of the proposed system, starting from document upload to final structured output generation. The diagram highlights the interaction between the user interface, backend services, OCR processing, the RAG pipeline, and the language model.

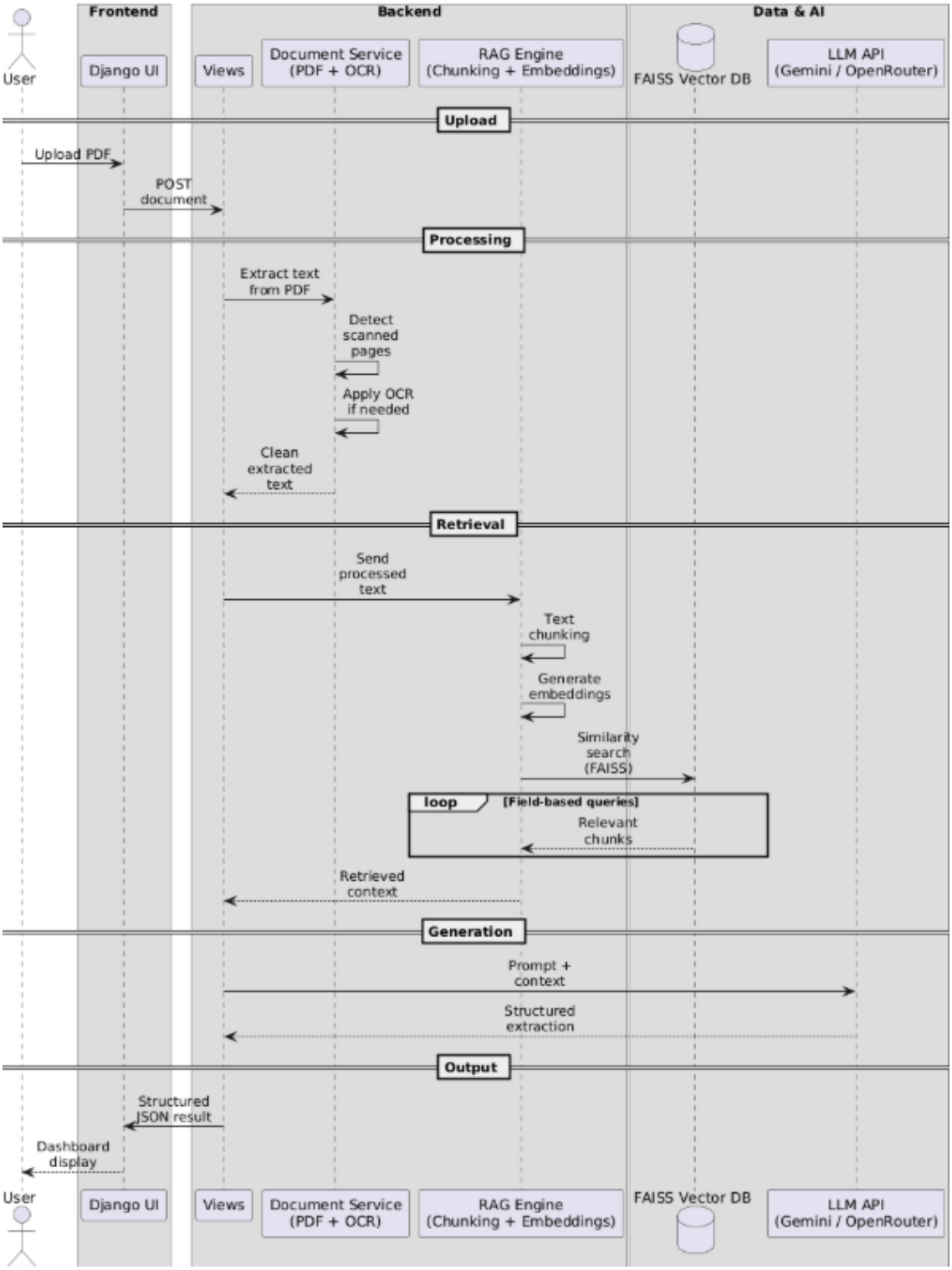

Figure 3.11: Sequence diagram of the Advanced RAG system

3.5.1 Frontend Architecture

The frontend layer represents the user-facing component of the system. It is implemented using Django templates rendered in HTML and styled with CSS, ensuring seamless integration with the backend while maintaining simplicity and usability.

The interface provides three main functionalities. First, a secure authentication mod-

ule allows users to access the system. Second, a document upload module enables users to submit tender documents in PDF format for processing. Third, a results dashboard displays the structured information extracted by the AI pipeline in a clear and organized format.

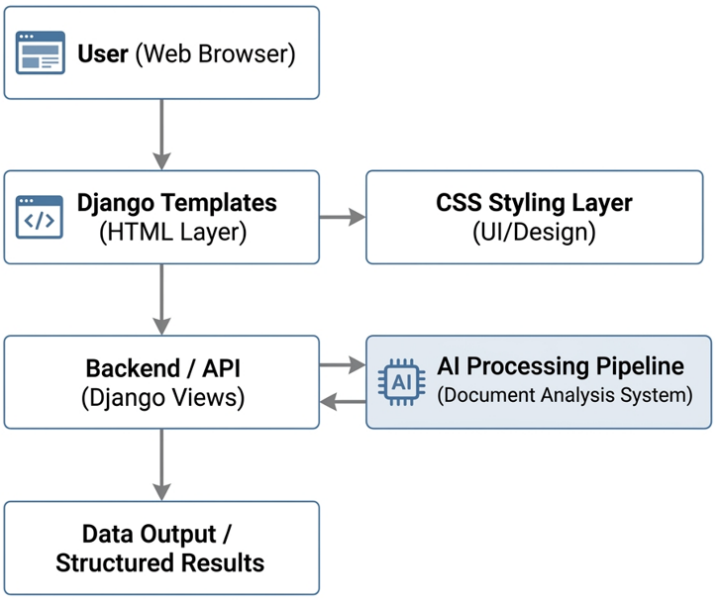

Figure 3.12: Frontend Architecture of the System

As illustrated in Figure 3.12, the frontend is designed as a layered structure that separates presentation, interaction, and communication with the backend. Django templates are responsible for rendering dynamic HTML pages, while CSS ensures a consistent and minimalistic user interface.

The frontend design prioritizes usability and clarity, as the system's primary objective is functional document analysis rather than visual complexity. Therefore, the interface is intentionally minimalistic, focusing on efficient interaction with the underlying processing pipeline.

3.5.2 Backend Architecture

The backend is implemented using Django and follows the Model–View–Template (MVT) architectural pattern. It represents the core processing layer of the system and acts as the intermediary between the user interface, the AI processing pipeline, and external services.

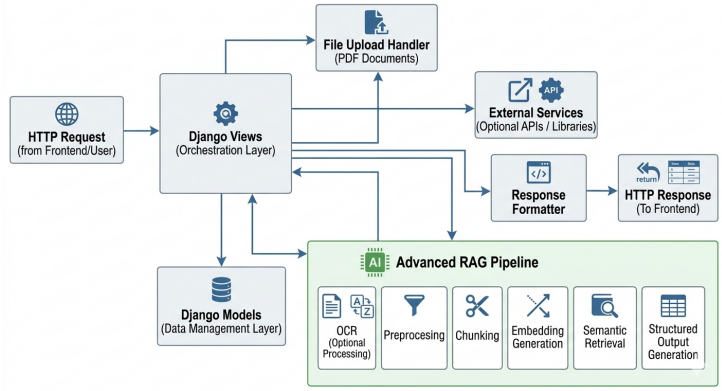

Figure 3.13: Backend Architecture of the System

As illustrated in Figure 3.13, the backend is responsible for handling HTTP requests, managing file uploads, and orchestrating the execution of the Advanced Retrieval-Augmented Generation (RAG) pipeline. When a document is uploaded, the backend triggers a sequence of processing stages, including text extraction, Optical Character Recognition (OCR) when required, preprocessing, chunking, embedding generation, semantic retrieval, and structured output generation.

Django views act as the central orchestration layer, ensuring that each component of the pipeline is executed in the correct order. The backend also handles response formatting before delivering the final structured results to the frontend in a structured and consistent format.

3.5.3 Authentication and User Access Management

To ensure secure access to the system, an authentication module is implemented using Django's built-in user management framework. This module supports user registration, login, and session management.

Authentication plays a critical role in the system, as it processes potentially sensitive procurement documents. Restricting access to authenticated users ensures data confidentiality and controlled usage of the platform.

Once authenticated, users gain access to the document upload functionality and the full processing capabilities of the system.

3.5.4 Integration with the AI Processing Pipeline

A key aspect of the architecture is its tight integration with the Advanced RAG pipeline. The web platform acts as the entry point for data ingestion, while the AI pipeline performs the core information extraction and reasoning tasks.

When a document is uploaded, the backend triggers a sequential processing workflow consisting of PDF parsing, OCR-based text extraction (if necessary), text preprocessing, chunking, embedding generation, and semantic retrieval using a vector database. The retrieved content is then passed to a Large Language Model, which generates structured outputs aligned with predefined procurement fields.

This integration ensures a seamless end-to-end workflow, connecting user interaction with advanced AI-based document understanding.

3.5.5 Communication with External APIs

The system relies on external Large Language Model APIs for the generation of structured outputs. In this project, models such as Gemini and OpenRouter are used to process retrieved contextual information and produce structured responses.

The backend communicates with these APIs by constructing carefully designed prompts that include the retrieved document chunks. The external model then returns a generated response, which is processed and formatted by the backend before being displayed to the user.

This design choice allows the system to leverage powerful pre-trained language models without requiring local deployment, thereby improving scalability, maintainability, and computational efficiency.

3.5.6 Summary

The proposed web platform provides a complete operational interface for the Advanced RAG system. By combining a Django-based backend, a lightweight HTML/CSS frontend, and external AI services, the platform ensures a smooth workflow from document upload to structured information extraction.

This architecture effectively bridges the gap between complex AI-based document processing and practical user interaction, making the system suitable for real-world deployment in automated procurement document analysis.

3.6 System Modeling

This section presents the modeling phase of the proposed system using standard software engineering diagrams. The objective of this phase is to formally describe the system's functional behavior, structural architecture, and deployment environment.

Given the complexity of the proposed Advanced RAG system, which integrates a web-based platform, an AI processing pipeline, and external language model APIs, multiple UML diagrams are used to represent different perspectives of the system.

These include use case diagrams to describe system functionalities from a user perspective, sequence diagrams to illustrate dynamic interactions, activity diagrams to model workflow execution, component diagrams to describe system architecture, and deployment diagrams to present the physical deployment of the system.

3.6.1 Use Case Diagram

The use case diagram presents a high-level functional perspective of the proposed system by modelling the interactions between external actors and the Advanced RAG platform. It identifies the principal services provided by the system, including user authentication, document upload, automated document processing, and the presentation of structured extraction results. It also incorporates administrative functionalities such as user management and system monitoring. Overall, this diagram contributes to a clearer understanding of the system boundaries, its operational scope, and the roles involved in its usage.

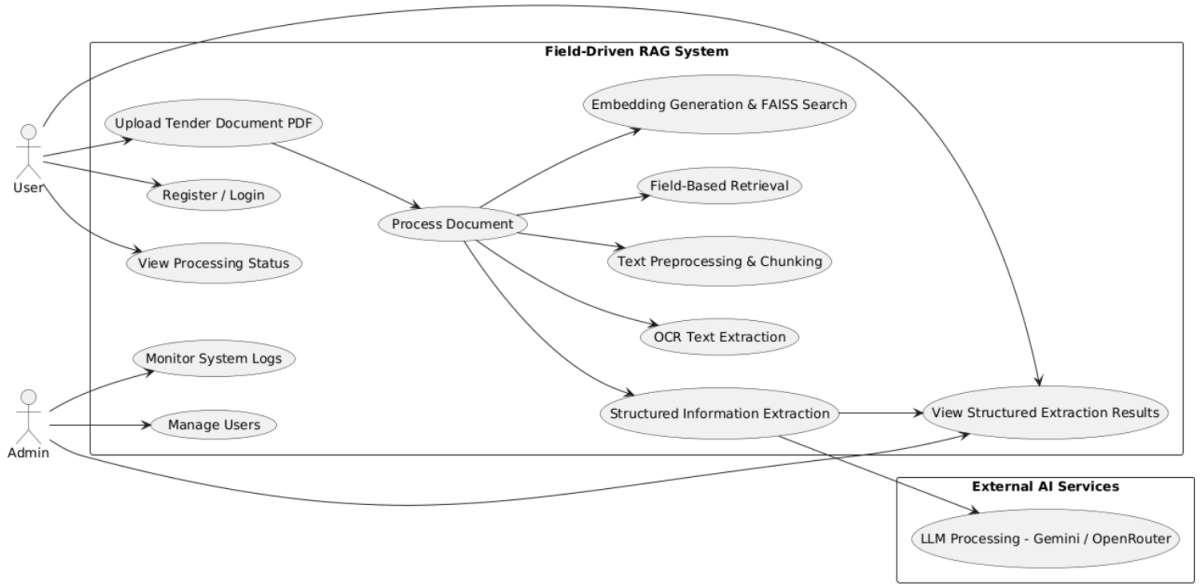

Figure 3.14: Use Case Diagram of the Advanced RAG System

3.6.2 Sequence Diagram

The sequence diagram describes the dynamic behavior of the system by modeling the chronological interactions between the user, the web platform, the backend services, and the external AI components. It illustrates the end-to-end processing of a user request, beginning with authentication and document upload and culminating in the generation and display of structured extraction results.

More specifically, it emphasizes the role of the Django backend in orchestrating the overall workflow. This includes handling incoming requests, coordinating the execution of the RAG pipeline, and managing communication with external Large Language Model APIs. Overall, the diagram provides a detailed representation of the system execution flow and highlights the interaction and coordination between the different architectural layers of the proposed solution.

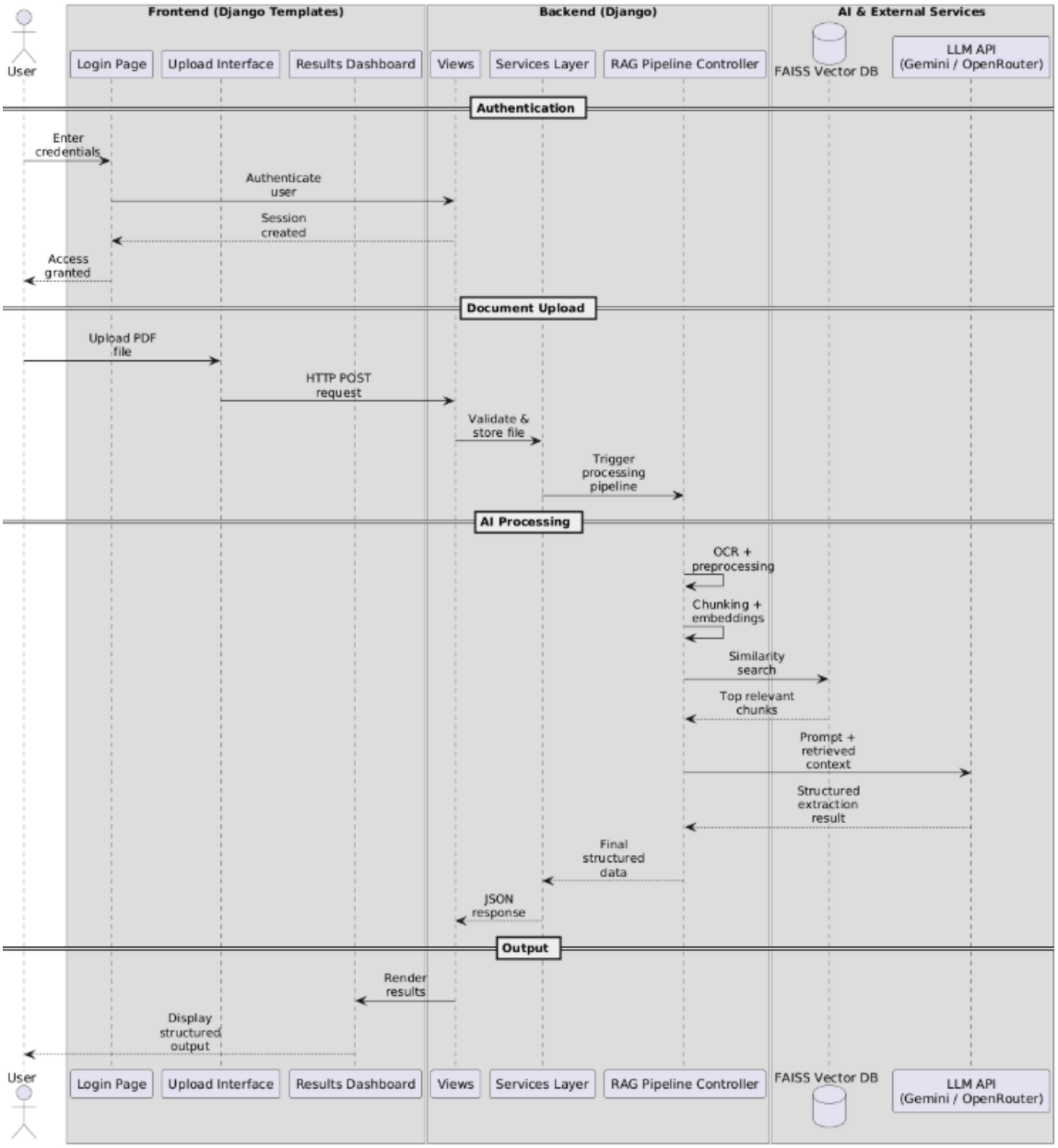

Figure 3.15: Sequence Diagram - Web Platform Interaction with Advanced RAG System

3.6.3 Activity Diagram

The activity diagram represents the operational workflow of the proposed Advanced RAG system. It provides a behavioral view of the system by illustrating the sequence of activities performed from the moment a user submits a document until the final structured results are generated and displayed.

Unlike the sequence diagram, which focuses on interactions between system components, the activity diagram emphasizes the internal processing flow and decision points within the system. It highlights the different stages of the pipeline, including document upload, preprocessing, OCR processing when necessary, text chunking, embedding generation, semantic retrieval, and structured output generation using a Large Language Model.

This diagram also captures the conditional nature of the processing workflow, partic-

ularly the decision to apply OCR depending on the type of input document. Overall, it provides a clear and structured representation of the end-to-end processing logic of the system.

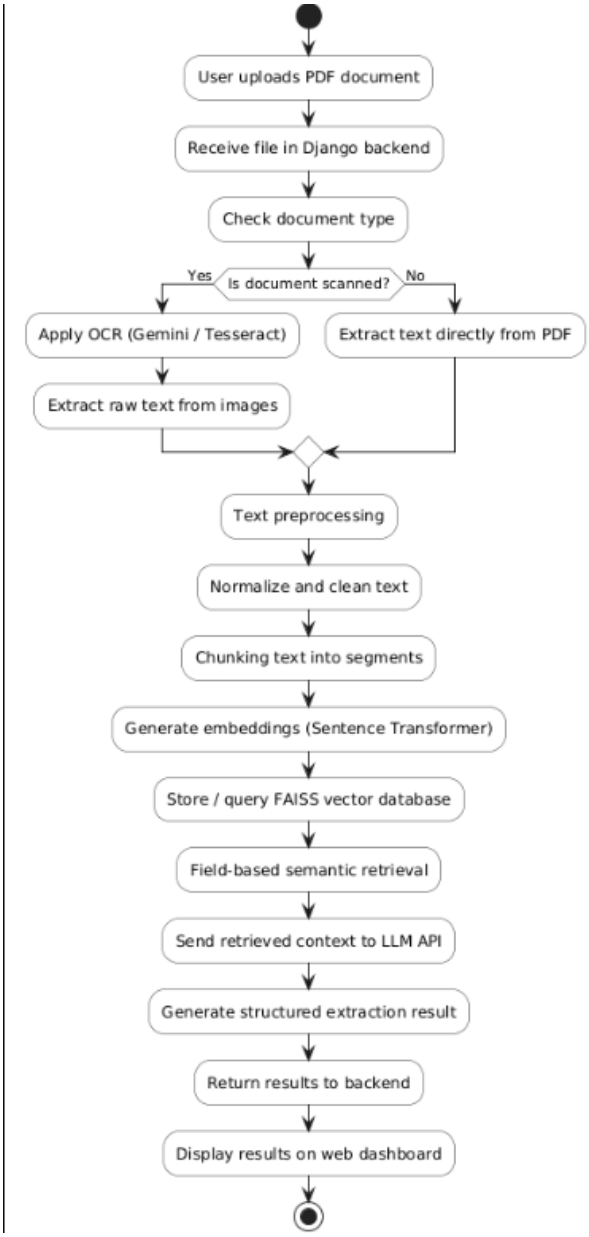

Figure 3.16: Activity Diagram of the Advanced RAG System

3.6.4 Component Diagram

The component diagram provides a static representation of the architecture of the proposed Advanced RAG system by identifying its principal software components and the relationships between them. It offers a modular view of the system, emphasizing the separation of concerns across the different layers that compose the overall solution.

The architecture is structured into several interconnected layers, including the frontend layer implemented using Django templates, which is responsible for user interaction, and the backend layer, which handles request processing and system orchestration. The

backend further coordinates the AI processing pipeline, which encompasses key tasks such as Optical Character Recognition (OCR), text preprocessing, document chunking, embedding generation, and semantic retrieval.

In addition, the system integrates a vector storage component based on FAISS to support efficient similarity search over high-dimensional embeddings. It also relies on external Large Language Model APIs to perform structured information extraction and response generation based on the retrieved contextual data.

This modular architecture promotes a clear separation of responsibilities between components, thereby enhancing the system's scalability, maintainability, and extensibility in the context of complex procurement document analysis.

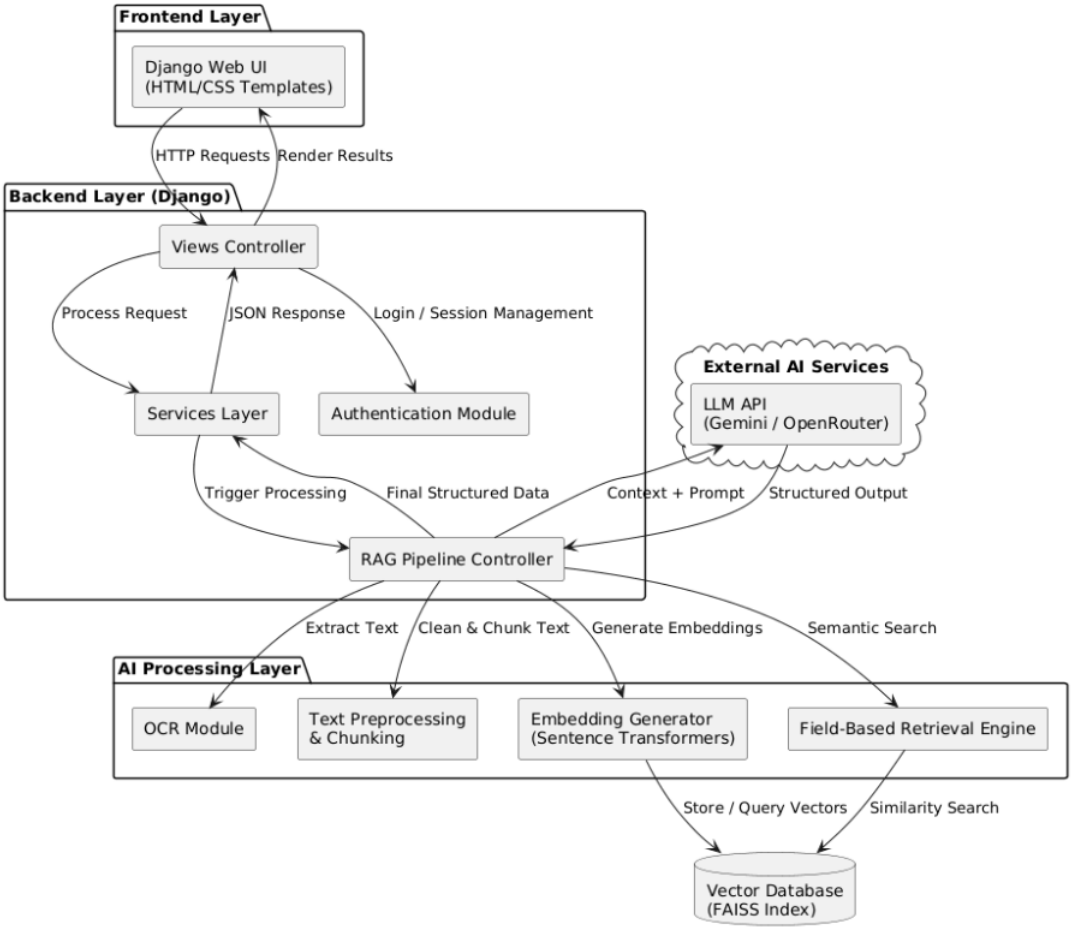

Figure 3.17: Component Diagram of the Advanced RAG System

3.6.5 Deployment Diagram

The deployment diagram represents the physical architecture of the proposed Advanced RAG system. It describes how the system components are distributed across different nodes, including the user device, web server, AI processing environment, and external AI services.

This diagram highlights the runtime environment of the application, showing how the Django-based web platform interacts with the OCR engine, RAG pipeline, vector database, and external Large Language Model APIs. It also illustrates the separation between client-side interaction, backend processing, and AI-driven services, ensuring a

clear understanding of the system deployment strategy.

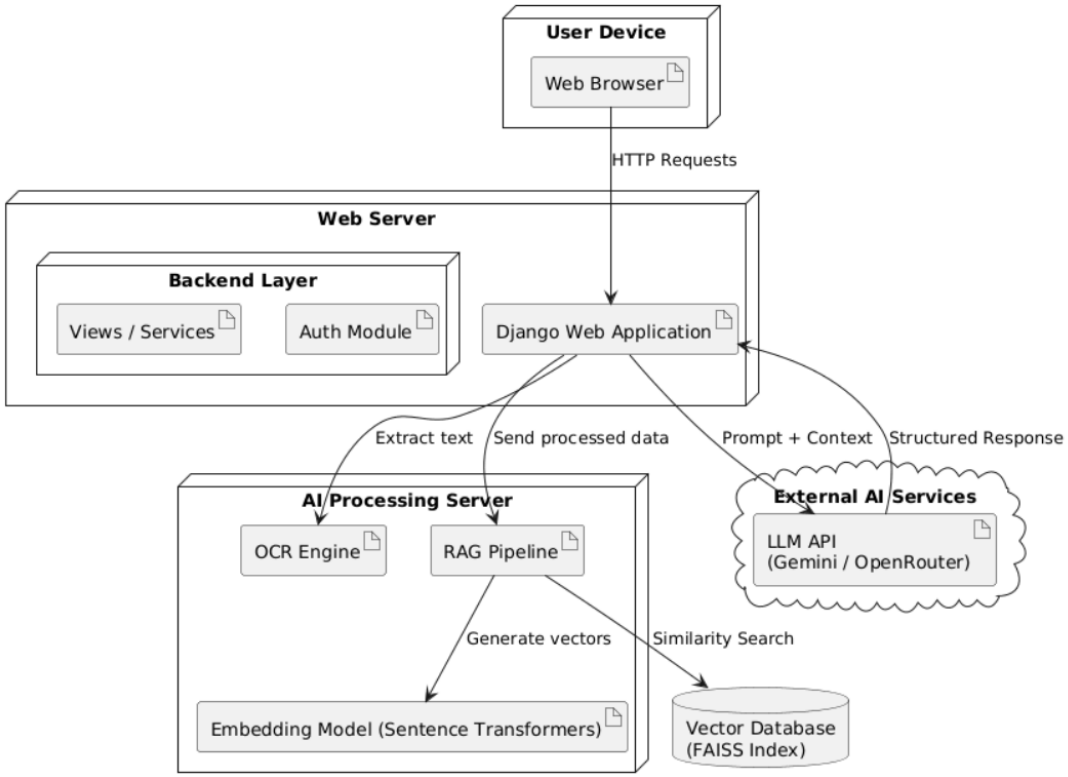

Figure 3.18: Deployment Diagram of the Advanced RAG System

3.7 Data Processing Workflow

The data processing workflow of the proposed Advanced RAG system represents a unified end-to-end pipeline that transforms raw procurement documents into structured, machine-readable outputs. The workflow begins with the upload of a PDF document through the Django-based web interface, after which the backend performs initial validation and routing to the processing pipeline. Depending on the nature of the document, Optical Character Recognition (OCR) is applied when necessary to extract textual content from scanned pages. The extracted text then undergoes preprocessing and segmentation into overlapping chunks, followed by embedding generation and semantic indexing using a vector database (FAISS). A field-oriented retrieval mechanism is subsequently applied to extract relevant contextual information for each procurement attribute. Finally, a Large Language Model is used to generate structured outputs based strictly on the retrieved context, ensuring consistency and factual accuracy in the final results.

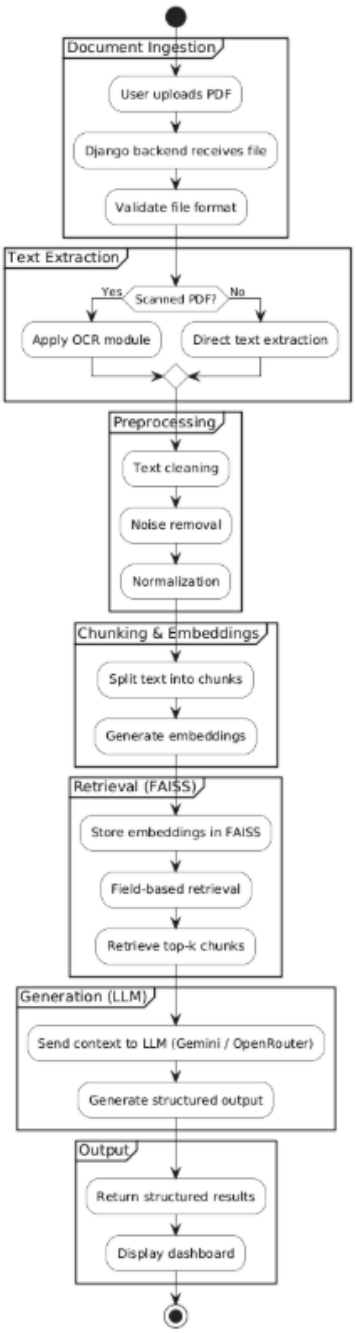

Figure 3.19: End-to-End Data Processing Workflow of the Field-Driven RAG System

3.8 Summary

This chapter presented the system design and modeling of the proposed solution. It compared existing retrieval architectures and motivated the choice of an advanced RAG-based approach.

The proposed architecture integrates OCR processing, document preprocessing, embedding generation, and semantic retrieval with structured response generation. The web platform architecture, system modeling diagrams, and data processing workflow were also introduced to describe the system's structure and end-to-end operation. Overall, this chapter provides the architectural foundation for the implementation phase.

Chapter 4

Implementation and Deployment

4.1 Introduction

After presenting the study and evaluation of different Retrieval-Augmented Generation (RAG) architectures in the previous chapters, this chapter focuses on the implementation and deployment of the proposed solution. Several retrieval approaches, including traditional RAG, GraphRAG, RAPTOR-based architectures, and Structured Retrieval-Augmented Generation (SRAG), were analyzed and evaluated according to the specific requirements, constraints, and objectives of the system. The final architecture was selected based on its ability to provide accurate information extraction, efficient semantic retrieval, scalability, and structured response generation for tender document analysis.

The main objective of this phase is to develop an intelligent platform capable of automatically analyzing tender documents in PDF format, extracting the most relevant information, and generating structured summaries using advanced artificial intelligence and semantic retrieval techniques.

The implemented solution is based on an advanced RAG architecture integrating multiple components, including text extraction from both native and scanned PDF documents, OCR processing, text preprocessing, semantic chunking, vector embedding generation, FAISS-based indexing, and structured response generation using a Large Language Model (LLM).

As part of the implementation process, the solution was integrated into a web platform developed using the Django framework, allowing users to upload tender documents and automatically obtain detailed analyses through a simple and interactive interface.

This chapter presents the development environment, the technologies and libraries used, the implementation of the different modules of the intelligent processing pipeline, and the integration of the solution into the web platform.

4.2 Development Environment and Technologies

This section presents the development environment, tools, and technologies adopted for the implementation of the proposed intelligent tender analysis system. The selection of the technological stack was guided by the requirements of the system, particularly the need to support artificial intelligence workflows, semantic information retrieval, and modern web application development. The overall architecture relies on a combination of machine learning libraries, NLP tools, and a web development framework to ensure scalability, modularity, and performance.

4.2.1 Programming Languages and Frameworks

The implementation of the proposed system is primarily based on Python, which was selected as the core programming language due to its extensive ecosystem dedicated to artificial intelligence, natural language processing, and data engineering. Python was used for the development of the entire AI processing pipeline, including document parsing, semantic chunking, embedding generation, vector indexing, and structured response generation.

The web application layer was developed using the Django framework (version 6.0.3), which provides a robust and scalable backend architecture. Django was responsible for managing HTTP requests, handling file uploads, interacting with the database, and integrating the AI pipeline into the web platform.

The frontend interface was implemented using HTML, CSS, and JavaScript, enabling the development of an interactive and user-friendly experience for uploading documents and visualizing extracted results.

To ensure reproducibility, dependency isolation, and environment portability, the development environment was configured using a dedicated virtual environment (`venv`). In addition, Visual Studio Code (VS Code) was used as the primary development environment due to its advanced debugging capabilities, extensibility, and strong support for Python-based projects. Source code management and version control were handled using Git, ensuring structured tracking of project evolution and collaborative development support.

Table 4.1: Programming Languages, Frameworks, and Development Tools

Technology / Tool	**Role**
Python 3.14	Core language used for implementing the AI pipeline and system logic
Django 6.0.3	Backend framework for web application development, routing, and system integration
HTML	Markup language for structuring frontend pages
CSS	Styling and responsive user interface design
JavaScript	Client-side interactivity and dynamic user experience
VS Code	Integrated development environment used for implementation and debugging
Git	Version control system for source code management and tracking changes
venv	Virtual environment used for dependency isolation and reproducibility
.gitignore	Configuration file used to exclude unnecessary files from version control (cache, logs, media, environments)

4.2.2 Libraries and Dependencies

The system integrates several specialized libraries and frameworks to support document processing, OCR, semantic retrieval, vector search, and AI-driven text generation. These libraries form the core computational layer of the proposed solution.

Table 4.2: Main Technologies and Libraries Used

Technology / Library	Role
Python 3.14	Core programming language for system implementation
Django 6.0.3	Backend framework for system integration and web services
pdfplumber	Extraction of structured text from native PDF documents
pdf2image	Conversion of PDF pages into images for OCR processing
Pillow (PIL)	Image processing and preprocessing operations
Google Generative AI	OCR processing and large language model-based text generation
SentenceTransformers	Generation of dense semantic embeddings for text chunks
FAISS	High-performance similarity search and vector indexing engine
NumPy	Numerical computations and vector normalization
tiktoken	Token-based text segmentation and chunking strategy
requests	HTTP communication with external APIs
unicodedata	Text normalization and character standardization
re	Regular expression-based text cleaning and pattern extraction
os	Operating system interactions and environment variable handling

The project dependencies were managed through a `requirements.txt` file, ensuring consistent installation of all required packages across different environments and facilitating deployment reproducibility.

4.2.3 Additional Models and Configurations Used

In addition to the core libraries, several pre-trained models and configuration components were integrated into the system to enhance performance in semantic understanding, token management, and generative capabilities.

all-MiniLM-L6-v2: This is a lightweight sentence embedding model from the SentenceTransformers library. It is used to convert textual chunks into dense vector representations that capture semantic meaning, enabling efficient similarity-based retrieval between user queries and document content.

cl100k_base: This tokenizer encoding, used through the `tiktoken` library, is designed for transformer-based models. It is employed to perform token-level segmentation of text, ensuring that chunking respects model input constraints while maintaining consistency with modern LLM tokenization schemes.

gemini-2.5-flash: This generative model from Google AI is used for two critical tasks: optical character recognition (OCR) of scanned PDF documents and structured text generation. It was selected due to its balance between computational efficiency and output quality, making it suitable for near real-time document analysis applications.

4.3 Implementation of the AI Processing Pipeline

This section presents the implementation of the proposed Retrieval-Augmented Generation (RAG) pipeline for automated tender document analysis. The system has been designed as a modular and extensible workflow, integrating document parsing, OCR-based extraction, semantic chunking, vector-based retrieval, and structured response generation. Each module of the pipeline was implemented to ensure robustness, scalability, and reliable extraction of structured information from heterogeneous PDF documents.

4.3.1 Implementation Environment and Core Dependencies

The implementation of the proposed system is based on a set of Python libraries and frameworks supporting document processing, artificial intelligence, semantic retrieval, and large language model integration. These dependencies constitute the foundational layer of the system and enable the execution of all processing stages.

The main imported libraries and modules used in the system are illustrated in Figure 4.1.

```python
import os
import re
import pdfplumber
import pytesseract
import unicodedata
import numpy as np
import faiss
import tiktoken
import requests
import google.generativeai as genai
from io import BytesIO
from PIL import Image
from pdf2image import convert_from_path
from sentence_transformers import SentenceTransformer
```

Figure 4.1: Core Python libraries and dependencies used in the proposed system

These libraries provide the following functionalities:

- Document parsing and PDF text extraction (`pdfplumber`)
- OCR-based text extraction from images (`pytesseract`, `Google Generative AI`)
- Image conversion and processing (`pdf2image`, `Pillow`)
- Text normalization and preprocessing (`unicodedata`, `re`)
- Tokenization and chunking (`tiktoken`)

- Semantic embedding generation (`SentenceTransformers`)
- Vector similarity search and indexing (`FAISS`)
- External API communication (`requests`, `google.generativeai`)

4.3.2 System Configuration and External API Setup

The proposed system integrates multiple external AI services and is designed to support flexible configuration of both OCR and Large Language Model (LLM) providers. This modular configuration allows the system to switch between different AI backends depending on performance requirements or deployment constraints.

Two main configuration parameters are defined at the system level:

- OCR provider selection (Gemini or alternative OCR engine)
- LLM provider selection (Gemini or OpenRouter-based models)

```
OCR_PROVIDER = "gemini"          # tesseract | gemini
LLM_PROVIDER = "gemini"      # openrouter | gemini
```

Figure 4.2: Initializing providers

To ensure secure and flexible deployment, API keys are retrieved from environment variables rather than hard-coded in the source code:

```
OPENROUTER_API_KEY = os.environ.get("OPENROUTER_API_KEY")
GEMINI_API_KEY = os.environ.get("GEMINI_API_KEY")
```

Figure 4.3: Configuring API keys

This approach improves security by avoiding direct exposure of sensitive credentials in the codebase.

Validation mechanisms are also implemented to ensure proper system initialization:

```
if LLM_PROVIDER == "openrouter" and not OPENROUTER_API_KEY:
    raise ValueError("OPENROUTER_API_KEY not set")

if not GEMINI_API_KEY:
    print("Warning: GEMINI_API_KEY not set (Gemini OCR/LLM won't work)")

genai.configure(api_key=GEMINI_API_KEY)
```

Figure 4.4: Error fallback

API-Based Model Execution Functions

The system supports two different inference backends for text generation:

OpenRouter API-based generation:

```python
def openrouter_call(prompt):
    response = requests.post(
        "https://openrouter.ai/api/v1/chat/completions",
        headers={
            "Authorization": f"Bearer {OPENROUTER_API_KEY}",
            "Content-Type": "application/json"
        },
        json={
            "model": "stepfun/step-3.5-flash",
            "messages": [
                {"role": "user", "content": prompt}
            ],
            "temperature": 0
        }
    )

    response.raise_for_status()
    return response.json()["choices"][0]["message"]["content"]
```

Figure 4.5: OpenRouter API-based generation

Gemini-based generation:

```python
def gemini_call(prompt):
    model = genai.GenerativeModel("gemini-2.5-flash")

    response = model.generate_content(prompt)

    return response.text
```

Figure 4.6: Gemini-based generation:

This dual-provider design increases the flexibility of the system, allowing it to adapt to different operational environments while maintaining consistent output quality.

4.3.3 PDF Document Processing

The first stage of the pipeline consists of extracting raw textual content from input PDF documents. The system initially attempts text extraction using `pdfplumber`, which is effective for digitally generated documents.

```python
def extract_text(file_path):
    text = ""

    with pdfplumber.open(file_path) as pdf:
        for i, page in enumerate[Page](pdf.pages, start=1):
            print(f"Processing page {i}")

            page_text = page.extract_text() or ""

            if len(page_text.strip()) < 50:
                print(f"Scanned page detected -> OCR page {i}")

                img = convert_pdf_page_to_image(file_path, i)

                page_text = gemini_ocr_image(img)

            text += page_text + f"\n(Page {i})\n"

    return text
```

Figure 4.7: PDF text extraction and OCR fallback mechanism

To handle scanned or low-quality documents, an adaptive detection mechanism is introduced. Specifically, pages for which the extracted text contains fewer than 50 characters are considered non-extractable. These pages are then forwarded to the OCR pipeline after conversion into images using `pdf2image`.

This hybrid strategy ensures that both native and scanned PDFs are processed uniformly within a single pipeline.

4.3.4 OCR-Based Text Extraction

For scanned documents, Optical Character Recognition is performed using the `gemini-2.5-flash` multimodal model. Unlike traditional OCR systems, this approach benefits from contextual understanding of document structure.

```python
def gemini_ocr_image(pil_image):
    model = genai.GenerativeModel("gemini-2.5-flash")

    prompt = """
    Extract ALL text from this document page exactly as written.

    Rules:
    - Preserve headings
    - Preserve bullet points
    - Preserve tables if possible
    - Do not summarize
    - Return raw text only
    """

    response = model.generate_content([
        prompt,
        pil_image
    ])

    return response.text
```

Figure 4.8: OCR-based text extraction using the Gemini multimodal model

A carefully engineered prompt is used to preserve structural elements such as headings, bullet points, and tables. The objective is to ensure faithful reconstruction of the

document content without summarization or information loss.

This approach significantly improves robustness when processing real-world tender documents, which often contain complex formatting and heterogeneous layouts.

4.3.5 Text Preprocessing

After extraction, the textual data is normalized and cleaned in order to improve consistency and reduce noise before further processing.

```
def preprocess_text(text):
    text = unicodedata.normalize("NFKC", text)
    text = re.sub(r'\bPage\s*\d+(\s*/\s*\d+)?\b', '', text)
    text = re.sub(r'[ \t]+', ' ', text)
    text = re.sub(r'\n\s*\n+', '\n\n', text)
    return text.strip()
```

Figure 4.9: Text preprocessing and normalization operations

Line-by-line implementation analysis:

- **Unicode normalization**

  `text = unicodedata.normalize("NFKC", text)`

  Standardizes Unicode characters to ensure consistent representation of equivalent symbols.

- **Removal of page markers**

  `text = re.sub(r'\bPage\s*\d+(\s*/\s*\d+)?\b', '', text)`

  Eliminates page identifiers that are typically introduced during PDF extraction.

- **Whitespace normalization**

  `text = re.sub(r'[ \t]+', ' ', text)`

  Reduces multiple spaces and tabulations into a single uniform space.

- **Paragraph structuring**

  `text = re.sub(r'\n\s*\n+', '\n\n', text)`

  Ensures consistent paragraph separation using standardized line breaks.

- **Final trimming**

  `return text.strip()`

  Removes leading and trailing whitespace from the processed text.

This stage ensures that subsequent modules operate on clean and semantically stable input data.

4.3.6 Text Chunking

The cleaned text is segmented into overlapping token-based chunks to facilitate semantic retrieval. Tokenization is performed using the `cl100k_base` encoder from the `tiktoken` library.

```python
encoding = tiktoken.get_encoding("cl100k_base")
MAX_TOKENS = 30000


def chunk_text(text, chunk_size=400, overlap=80):
    tokens = encoding.encode(text)

    chunks = []
    start = 0

    while start < len(tokens):
        end = start + chunk_size
        chunk = encoding.decode(tokens[start:end])
        chunks.append(chunk)
        start += chunk_size - overlap

    return chunks
```

Figure 4.10: Token-based text segmentation with overlap strategy

Each chunk is defined with a fixed size of 400 tokens and an overlap of 80 tokens. The overlap mechanism is critical to preserve contextual continuity between adjacent segments and reduce semantic fragmentation during retrieval.

4.3.7 Embedding Generation

Each text chunk is transformed into a dense vector representation using the pre-trained `all-MiniLM-L6-v2` model from SentenceTransformers.

```python
embedding_model = SentenceTransformer("all-MiniLM-L6-v2")


def normalize(vectors):
    norms = np.linalg.norm(vectors, axis=1, keepdims=True)
    return vectors / (norms + 1e-10)
```

Figure 4.11: Semantic embedding generation using SentenceTransformers

To ensure consistent similarity computation, all embeddings are normalized using L2 normalization. This enables cosine similarity-based retrieval while improving numerical stability during vector comparison.

4.3.8 Vector Indexing with FAISS

The generated embeddings are indexed using FAISS (Facebook AI Similarity Search), specifically through an `IndexFlatIP` structure.

```python
chunk_embeddings = embedding_model.encode(chunks)
chunk_embeddings = normalize(chunk_embeddings)

dimension = chunk_embeddings.shape[1]

index = faiss.IndexFlatIP(dimension)
index.add(np.array(chunk_embeddings).astype("float32"))
```

Figure 4.12: Vector indexing and similarity search using FAISS

This configuration enables efficient inner-product similarity search, which becomes equivalent to cosine similarity when embeddings are normalized. FAISS provides high-performance nearest-neighbor search, making the system scalable for large document collections.

4.3.9 Field-Oriented Semantic Retrieval

A key contribution of this work is the use of a field-oriented retrieval strategy. Instead of relying on a single global query, the system defines structured semantic queries corresponding to specific tender categories such as Budget, Duration, SLA, Team Composition, and Financial Offer.

```python
fields_queries = {
    "Type ": " mode de réalisation, forfait, régie, contrat à prix fixe,  mode de facturation",
    "Objet": "Objet du marché, description du projet, descriptif des besoins, finalité, infrastructure ou solution à mettre en place",
    "Budget": "Montant estimé, décomposition des prix, budget total, coût global, DPGF, offre financière, prix, estimation",
    "Durée": "Durée du contrat, délai d'exécution, période d'exécution, date de début et fin, calendrier, planning",
    "Modalités de paiement": "Conditions et modalités de paiement, bon de commande, échéancier, facturation",
    "Équipe projet": "Groupe de personnes,Equipe technique,Nombre de personnes dans l'équipe, postes requis, rôles, compétences techniques de l'équipe du candidat",
    "SLA": "Exigence de support,Niveaux de service, disponibilité, support technique, performance, temps de réponse, latence,Niveau de disponibilité des équipements",
    "Références": "Références, projets similaires, expériences, fournisseurs antérieurs, références techniques",
    "Cautionnement": "Cautionnement, garantie financière, dépôt de garantie, caution bancaire",
    "Grille de notation": "Critères d'évaluation, pondération, valeur technique, prix, garanties, livrables, Critères d'attribution",
    "Modèle de CV": "Format du CV exigé, informations à inclure, modèle proposé,Pièces à produire ",
    "Offre financière": "Structure de l'offre financière, DPGF, prix unitaire, quantité, coût total",
    "Visite des lieux": "Visite du site obligatoire ou facultative, inspection préalable",
    "Date de remise": " Date limite de réception des offres,Date limite de remise,échéance, soumission, deadline"
}
```

Figure 4.13: Definition of structured field-based semantic queries

```python
def retrieve_chunks(query_dict, top_k=10):
    all_chunks = []

    for field, query in query_dict.items():
        query_embedding = embedding_model.encode([query])
        query_embedding = normalize(query_embedding)

        distances, indices = index.search(
            np.array(query_embedding).astype("float32"),
            top_k
        )

        for i in indices[0]:
            chunk = chunks[i]

            if chunk not in all_chunks:
                all_chunks.append(chunk)

    return all_chunks
```

Figure 4.14: Field-oriented semantic retrieval using vector similarity search

Each query is independently executed over the vector index to retrieve the most relevant document chunks. The retrieved results are then merged while removing duplicates to form a unified and comprehensive context representation.

This structured retrieval strategy significantly improves precision and ensures systematic coverage of all relevant document aspects.

4.3.10 Structured Response Generation

The final stage of the pipeline consists of generating a structured and coherent summary using the `gemini-2.5-flash` large language model.

```
def generate_summary(context):
    prompt = f"""
Tu es un expert senior en analyse d'appels d'offres publics.

EXTRACTION STRUCTURÉE :

Règles strictes :
1. Utilise uniquement les informations présentes dans le CONTEXTE.
2. N'invente aucune information.
3. Résume sous forme de points courts.
4. Si l'information n'existe pas écrire "Non spécifié dans le document".
5. Pour **Durée**, indique la durée totale du projet, les phases principales, la maintenance et le support si spécifiés.
6. Pour **Équipe projet**, indique le nombre de personnes, leurs postes, rôles, responsabilités, côté client et côté fournisseur.
7. Pour **Offre financière**, déduplique les informations répétées et rends-les lisibles.

Format STRICT (Markdown) :

RAG STRUCTURED TENDER EXTRACTION (Markdown)

**Objet :**
**Budget :**
**Durée :**
**Modalités de paiement :**
**Équipe projet :**
**Qualification :**
**Pénalités :**
**SLA :**
**Références :**
**Cautionnement :**
**Grille de notation :**
**Modèle de CV :**
**Offre financière :**
**Visite des lieux :**
**Date de remise :**

CONTEXTE DOCUMENT :

{context}
"""

    if LLM_PROVIDER == "openrouter":
        return openrouter_call(prompt)

    elif LLM_PROVIDER == "gemini":
        return gemini_call(prompt)

    else:
        raise ValueError("Invalid LLM provider")
```

Figure 4.15: Structured response generation using the Gemini 2.5 Flash model

A strict prompt engineering strategy is applied to enforce the following constraints:

- The model must rely exclusively on retrieved context
- Hallucinated or external information is strictly prohibited
- Missing information must be explicitly indicated
- Output must follow a predefined structured Markdown format

The final output is generated in a structured Markdown format, ensuring both readability and seamless integration into the web-based user interface.

4.4 Integration into the Django Web Platform

After the implementation of the AI processing pipeline, the system was integrated into a web-based platform developed using the Django framework. This integration transforms

the standalone AI pipeline into a complete end-to-end application accessible through a user-friendly interface. The platform follows the MVT (Model–View–Template) architectural pattern, ensuring a clear separation between data management, business logic, and presentation layers.

4.4.1 Project Structure

The system is organized following a modular Django application architecture, ensuring separation of concerns between data management, business logic, and presentation layers. The main application structure is presented as follows:

```
    Directory: C:\Users\HP\django-silog-app\appel_offre


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/8/2026   6:20 PM                migrations
d-----         5/8/2026   6:34 PM                __pycache__
-a----         5/8/2026   3:17 PM             66 admin.py
-a----         5/8/2026   3:17 PM            101 apps.py
-a----         5/8/2026   6:34 PM            686 models.py
-a----         5/8/2026   6:34 PM           1767 queries.py
-a----         5/8/2026   6:34 PM           4034 services.py
-a----         5/8/2026   3:17 PM             63 tests.py
-a----         5/8/2026   6:34 PM            197 urls.py
-a----         5/8/2026   6:34 PM           1430 views.py
-a----         5/8/2026   3:17 PM              0 __init__.py
```

Figure 4.16: Backend structure

In addition, the frontend interface is implemented in the template directory:

```
    Directory: C:\Users\HP\django-silog-app\templates


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/8/2026   3:17 PM                appel_offre
```

Figure 4.17: Frontend directory

This structure promotes maintainability, scalability, and clear separation between the AI processing logic and the web interface layer.

4.4.2 Backend Integration

The backend of the system is implemented within a dedicated Django application named `appel_offre`. This module is responsible for handling file uploads, managing database storage, and orchestrating the execution of the AI processing pipeline.

The data model is defined using the `AppelOffre` class, which stores uploaded PDF files, generated results, processing status, and potential error messages.

```python
from django.db import models


class AppelOffre(models.Model):
    file = models.FileField(upload_to="uploads/")
    result = models.TextField(blank=True, null=True)

    status = models.CharField(
        max_length=20,
        choices=[
            ('pending', 'Pending'),
            ('processing', 'Processing'),
            ('done', 'Done'),
            ('failed', 'Failed')
        ],
        default='pending'
    )

    error_message = models.TextField(blank=True, null=True)

    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-created_at']

    def __str__(self):
        return f"Appel d'offre #{self.id}"
```

Figure 4.18: Data Model Definition

The main backend logic is implemented in the `views.py` file. When a user uploads a PDF document, the system performs the following operations:

- Validation of the uploaded file format
- Storage of the file in the media directory
- Execution of the AI processing pipeline
- Storage of the generated structured result in the database

```python
from django.shortcuts import render
from .models import AppelOffre
from .services import run_pipeline
from .queries import fields_queries


def upload_ao(request):

    result = None
    error = None
    if request.method == "POST":

        file = request.FILES.get("fichier")
        if not file:
            return render(request, "appel_offre/upload.html", {
                "error": "No file uploaded"
            })
        if not file.name.lower().endswith(".pdf"):
            return render(request, "appel_offre/upload.html", {
                "error": "Only PDF files are allowed"
            })
        try:
            print("📥 FILE RECEIVED")

            obj = AppelOffre.objects.create(file=file)
            file_path = obj.file.path

            print("🚀 STARTING PIPELINE")

            result = run_pipeline(file_path, fields_queries)

            print("✅ PIPELINE DONE")

            obj.result = result or "No result generated"
            obj.save()

        except Exception as e:
            print("❌ ERROR:", e)
            error = str(e)

    return render(request, "appel_offre/upload.html", {
        "result": result,
        "error": error
    })


def liste_ao(request):
    aos = AppelOffre.objects.all().order_by('-created_at')
    return render(request, "appel_offre/liste.html", {"aos": aos})
```

Figure 4.19: Main backend logic

This workflow ensures that each uploaded document is processed asynchronously at the application level and persistently stored for later consultation.

4.4.3 Frontend Interface

The frontend layer of the system is implemented using HTML, CSS, and JavaScript. It provides an interactive interface that allows users to upload tender documents and visualize structured AI-generated results.

The upload interface supports multiple usability features, including drag-and-drop file selection, real-time file preview, and a loading indicator during processing.

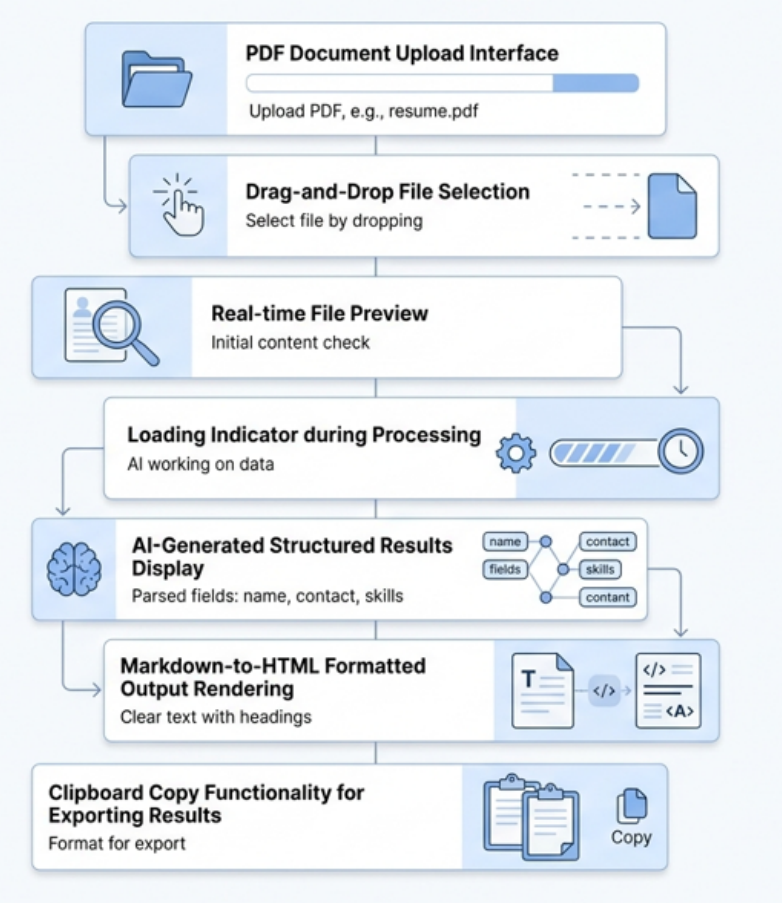

Figure 4.20: Frontend logic pipeline

Once processing is completed, the extracted results are displayed in a structured and formatted layout. A JavaScript-based formatting mechanism is applied to convert Markdown-style output into readable HTML components.

Additionally, a clipboard copy feature is integrated to allow users to easily export the generated analysis for external use.

These interface optimizations significantly improve user experience by enabling fast and intuitive access to AI-generated insights.

4.4.4 Service Layer Architecture

To ensure modularity and maintainability, the system adopts a service-oriented structure within the Django application. The core AI logic is separated from the web layer and implemented in a dedicated `services.py` module.

This design choice ensures a clear separation of concerns between:

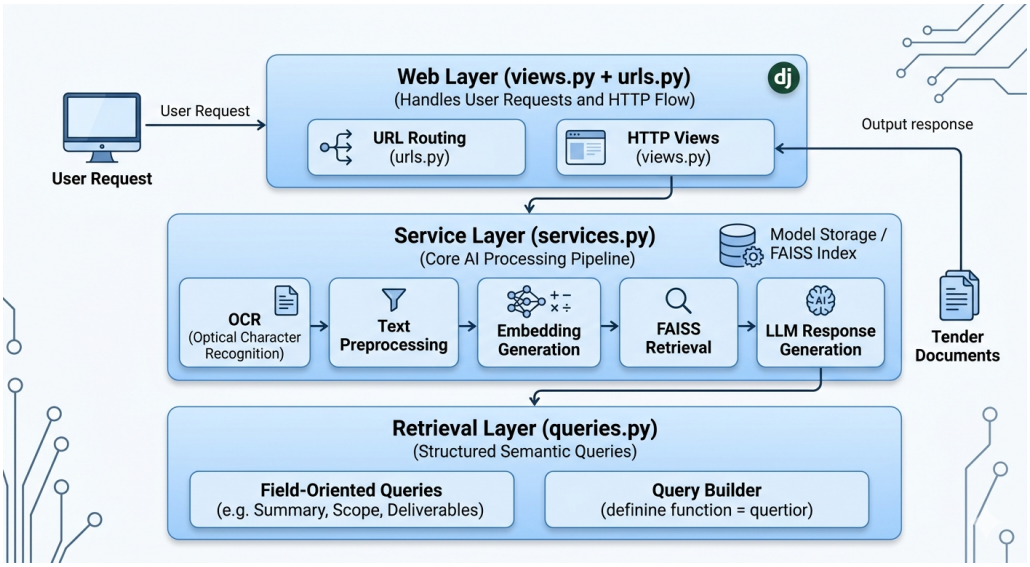

Figure 4.21: Service Layer Architecture nd Workflow

Web request handling and URL routing are managed through `urls.py`, which maps incoming user requests to the appropriate application components. The core business logic and execution of the AI-powered RAG pipeline are implemented in `services.py`, allowing processing tasks to remain independent from the presentation layer. In addition, the retrieval strategy is defined through structured semantic queries stored in `queries.py`, which guide the field-oriented retrieval mechanism used during information extraction. This layered organization improves system modularity, enhances code maintainability, and facilitates future scalability and extension of the platform.

4.5 System Results and User Interface

This section presents the operational results of the developed intelligent tender analysis system through its web-based user interface. It illustrates the complete user workflow, from document upload to the visualization of structured AI-generated outputs. The objective is to demonstrate the practical usability of the system and validate its end-to-end functionality within a real-world deployment scenario.

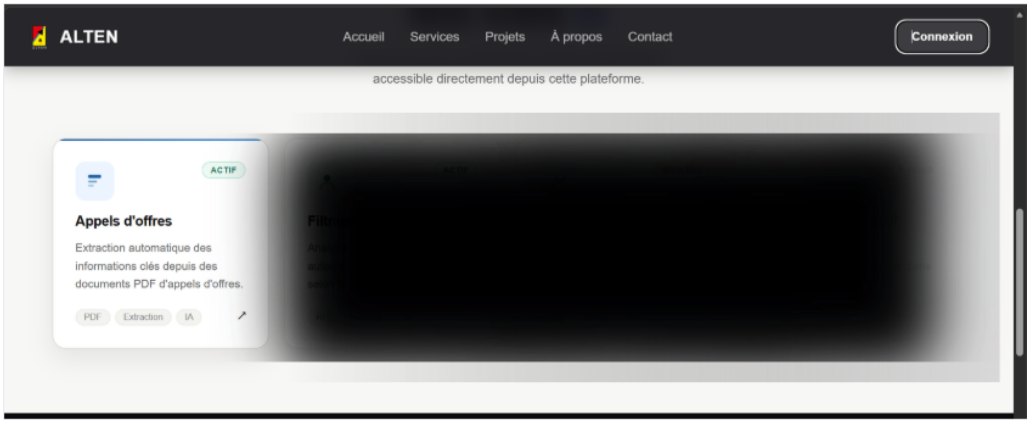

Figure 4.22: Home Page of the AI Website

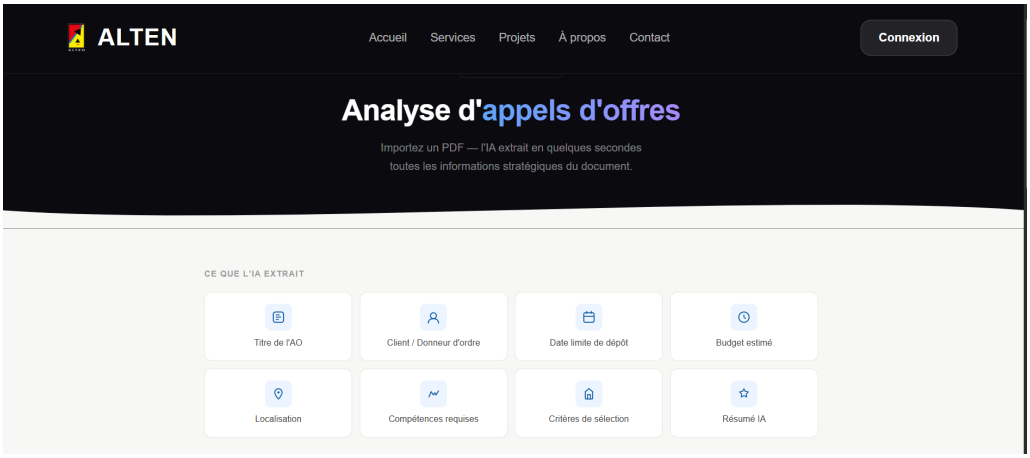

Figure 4.23: Tender Analysis Page

4.5.1 Document Upload Interface

The system provides a user-friendly upload interface that allows users to submit tender documents in PDF format. The interface supports drag-and-drop functionality as well as traditional file selection. This design ensures ease of use and improves accessibility for end users.

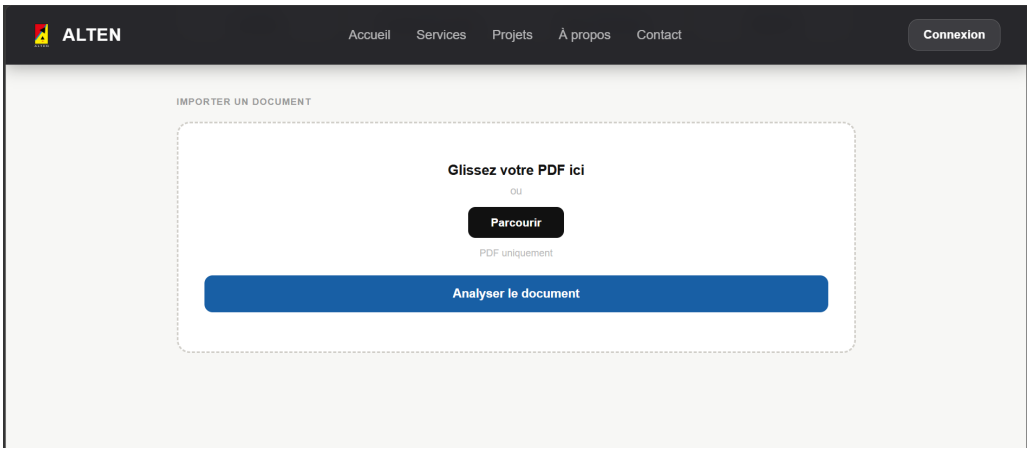

Figure 4.24: Document upload interface with drag-and-drop functionality for tender submission

4.5.2 Processing and Loading Indicator

Once a document is submitted, the system triggers the AI processing pipeline. During this stage, a real-time loading indicator is displayed to inform the user that the document is being analyzed. This improves user experience by providing feedback on system activity.

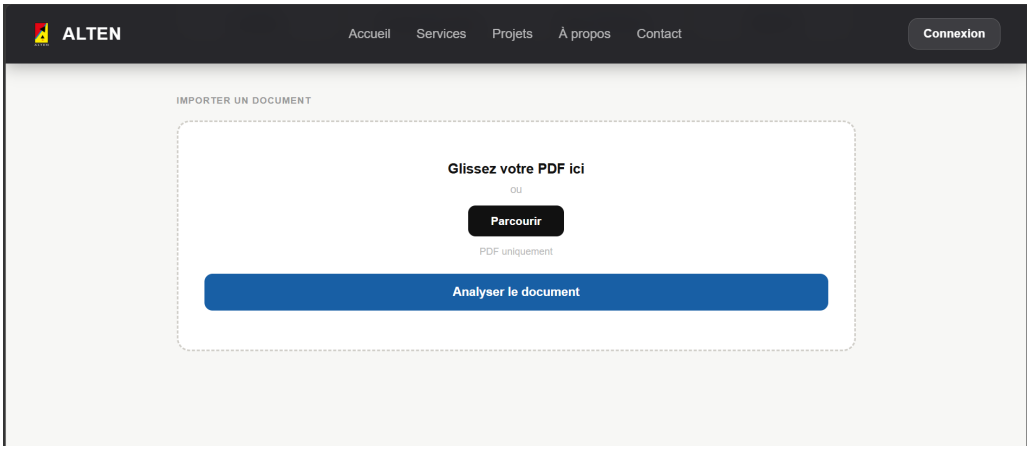

Figure 4.25: Processing state with loading indicator during AI pipeline execution

4.5.3 Structured Extraction Results

After processing is completed, the extracted information is displayed in a structured format. The results are organized into predefined fields such as budget, duration, SLA, and financial requirements. This structured visualization allows users to quickly interpret and analyze the extracted content.

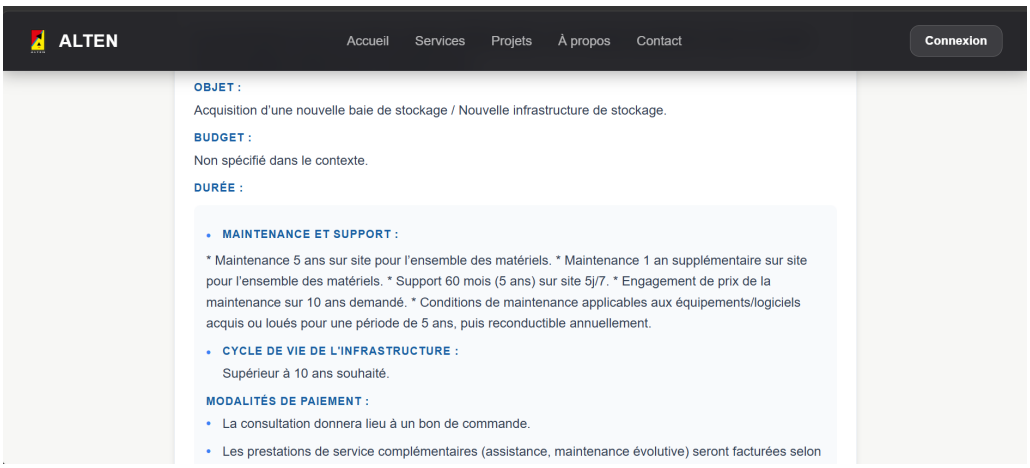

Figure 4.26: Structured AI-generated extraction results displayed in the web interface

4.5.4 Export and Copy Functionality

To enhance usability, the interface includes a clipboard copy feature that allows users to easily export the generated results for external use. This functionality is particularly useful for reporting and further document processing.

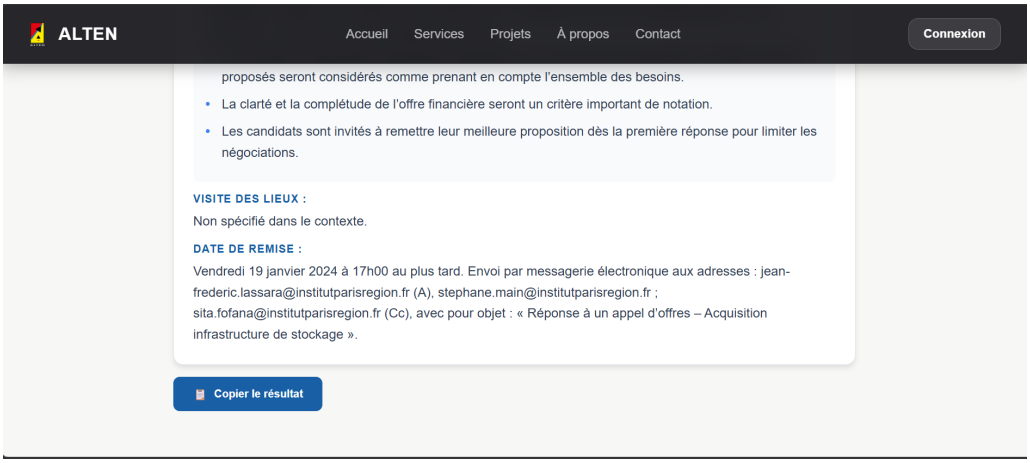

Figure 4.27: Clipboard copy functionality enabling export of extracted analysis results

This section demonstrates that the proposed system is not only effective in terms of backend processing and AI-driven extraction, but also provides a practical and intuitive user interface suitable for real-world deployment in enterprise environments.

4.6 Testing and Validation

This section presents the testing and validation process carried out to evaluate the performance and reliability of the proposed intelligent tender analysis system. The objective of this phase is to verify the effectiveness of the implemented Retrieval-Augmented Generation (RAG) pipeline in extracting relevant information from different types of tender documents.

The evaluation was conducted using real-world tender documents containing heterogeneous structures, scanned pages, tables, technical specifications, and administrative information. The system was tested under multiple scenarios in order to assess the robustness of the extraction pipeline, the reliability of the OCR module, and the relevance of the semantic retrieval mechanism.

4.6.1 Test Dataset and Evaluation Setup

The testing dataset consists of multiple tender documents collected from real procurement and call-for-tender environments, including **digitally generated PDF documents, scanned PDF files, semi-structured documents containing tables and complex formatting, as well as large multi-page tender specifications** The diversity of these documents enables the evaluation of the proposed system under realistic operational conditions and ensures that the pipeline is capable of handling heterogeneous document structures and varying formatting styles.

To objectively assess the quality of the generated outputs, a ground truth dataset was constructed using validated structured extractions corresponding to each tender document. This reference dataset serves as a benchmark for evaluating the accuracy and reliability of the information extraction process during experimentation.

4.6.2 Evaluation Methodology

The validation process is based on a comparison between the structured outputs generated by the AI system and the manually prepared ground truth data.

The evaluation focuses on several important aspects:

- **Extraction Accuracy:** Verification of the correctness of extracted information such as budget, duration, SLA, financial offer, and team composition.
- **OCR Reliability:** Evaluation of the OCR module performance on scanned and low-quality documents.
- **Completeness of Extraction:** Validation that the system successfully extracts the majority of required tender fields without significant information loss.

The evaluation was not limited to exact textual matching only. Semantic equivalence between the generated outputs and the expected results was also considered during validation.

4.6.3 Ground Truth Comparison

To ensure a reliable and objective evaluation process, the outputs generated by the system were compared against a manually validated ground truth dataset stored in JSON format.

The ground truth file contains the expected structured extraction results for each tender document and serves as the reference benchmark used during testing.

Figure 4.29 presents an example of the ground truth JSON structure used during the evaluation phase. for a tender that has 30 page and contains aslo tables as visible in the following image:

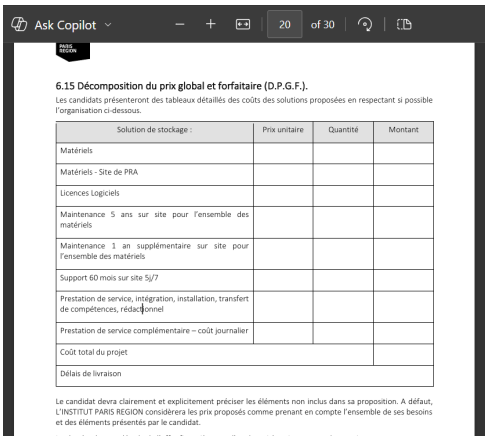

Figure 4.28: Example of the pdf used for the comparison

```
{
  "Type :": "** Appel d'offres restreint pour acquisition d'équipement informatique (baie de stockage)",
  "Objet :": "** Acquisition d'une nouvelle baie de stockage pour le système d'information de l'Institut Paris Region, incluant fourniture, intégration,
              installation, maintenance et support",
  "Budget :": "** Non spécifié dans le document (structure de prix détaillée demandée via DPGTF)",
  "Durée :": "** Maintenance 5 ans sur site + 1 an supplémentaire ; Support 60 mois (5 ans) ; Projet : livraison mars 2024, mise en service début T2 2024",
  "Modalités de paiement :": "** Non spécifié dans le document",
  "Équipe projet :": "** Non spécifié dans le document (contacts : Jean-Frederic LASSARA pour technique, Stéphane MAIN pour administratif)",
  "SLA :": "** Disponibilité minimale 99,999% annuel en heures ouvrées ; Support téléphonique 8h30-18h30, 5 jours sur 7 ; Télémaintenance possible",
  "Références :": "** Non spécifié dans le document",
  "Cautionnement :": "** Non spécifié dans le document",
  "Grille de notation :": "** Valeur technique 35% ; Prix 35% ; Garantie/évolutivité/support 15% ; Conditions d'exécution 10% ; Livrables 5%",
  "Offre financière :": "** Structure demandée : Solution de stockage (prix unitaire, quantité, montant) ;
                         Matériels (site PRA) ; Licences logiciels ; Maintenance 5 ans ; Maintenance 1 an supplémentaire ; Support 60 mois ;
                         Prestations service (intégration, installation, transfert compétences, rédactionnel) ; Prestation complémentaire (coût journalier) ; Coût total projet",
  "Visite des lieux :": "** Non spécifié dans le document",
  "Date de remise :": "** 19 janvier 2024 à 17h00 (par email à jean-frederic.lassara@institutparisregion.fr)"
}
```


Figure 4.29: Example of the ground truth JSON file used for system evaluation

After processing the uploaded tender document, the proposed AI pipeline generates a structured output containing the extracted information.

The following fogures illustrate an example of the structured extraction results generated by the system.


```
{
    "objet": [
      "Acquisition d'une nouvelle infrastructure de stockage.",
      "Implémentation en datacenter pour répondre aux besoins croissants en volumétrie et exploitation de données massives.",
      "Renforcer et consolider l'infrastructure de stockage, augmenter les performances et capacités actuelles.",
      "Bénéficier des dernières avancées technologiques, préparer aux technologies Big Data et Deep Learning.",
      "Mettre en place un plan de reprise informatique (PRI)."
    ],
    "budget": {
      "montant": "Non spécifié dans le document.",
      "exigences": [
        "L'offre financière doit présenter un Coût total du projet.",
        "La clarté et la complétude de l'offre financière seront un critère important de notation."
      ]
    },
    "duree_et_calendrier": {
      "duree_totale_infrastructure": "Cycle de vie supérieur à 10 ans.",
      "phases_principales_projet": {
        "Q4_2023": "Cahier des charges",
        "Q1_2024": "Installation Datacenter (DC)",
        "seconde_quinzaine_mars_2024": "Livraison du matériel et installation physique.",
        "debut_Q2_2024": "Mise en service, paramétrages et tests de la nouvelle infrastructure DC.",
        "Q2_Q3_2024": "Réplication provisoire.",
        "Q4_2024": "Mise en production et déménagement."
      },
      "garantie_constructeur": "Durée à compter des dates de livraison.",
      "engagement_prix_maintenance": "Engagement de prix sur 10 ans.",
      "contrats_de_maintenance_specifiques": [
        "Période initiale : 5 ans pour les équipements/logiciels acquis ou loués.",
        "Reconduction : Annuellement en fonction de la solution proposée.",
        "Prestation de maintenance corrective particulière, adaptée à la période de couverture des garanties constructeur."
      ],
      "maintenance_incluse_offre_financiere": [
        "Maintenance 5 ans sur site pour l'ensemble des matériels.",
        "Maintenance 1 an supplémentaire sur site pour l'ensemble des matériels."
      ],
```


Figure 4.30: Structured extraction results generated by the proposed AI system part 1

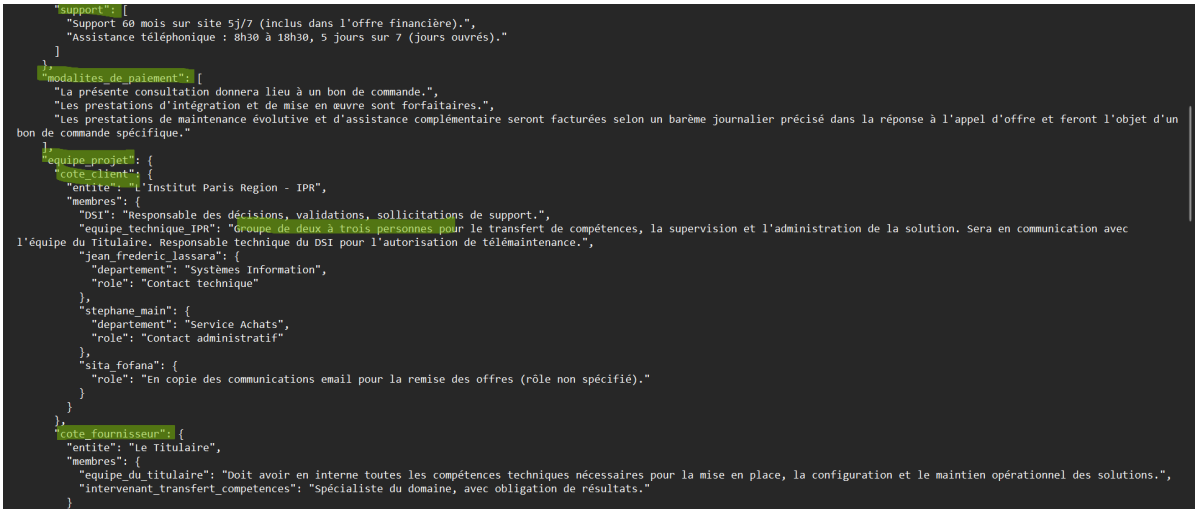

Figure 4.31: Structured extraction results generated by the proposed AI system part 2

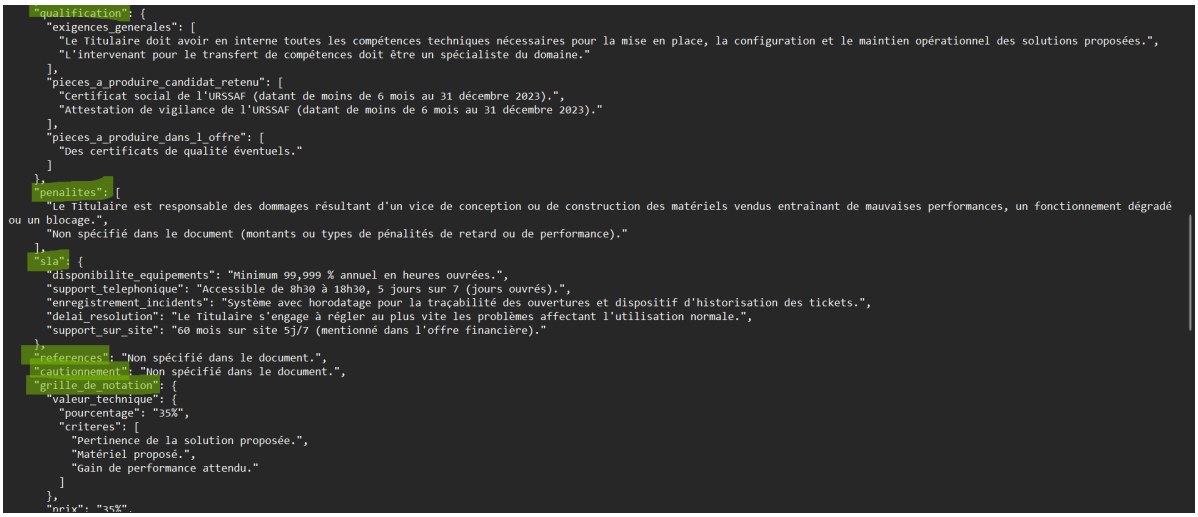

Figure 4.32: Structured extraction results generated by the proposed AI system part 3

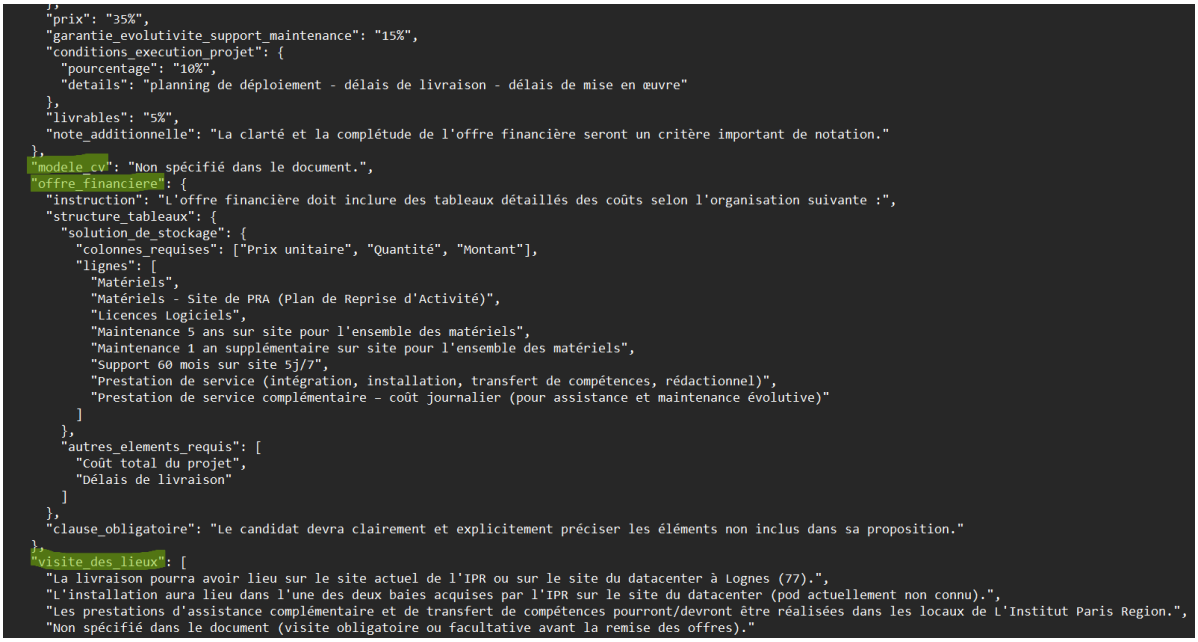

Figure 4.33: Structured extraction results generated by the proposed AI system part 4

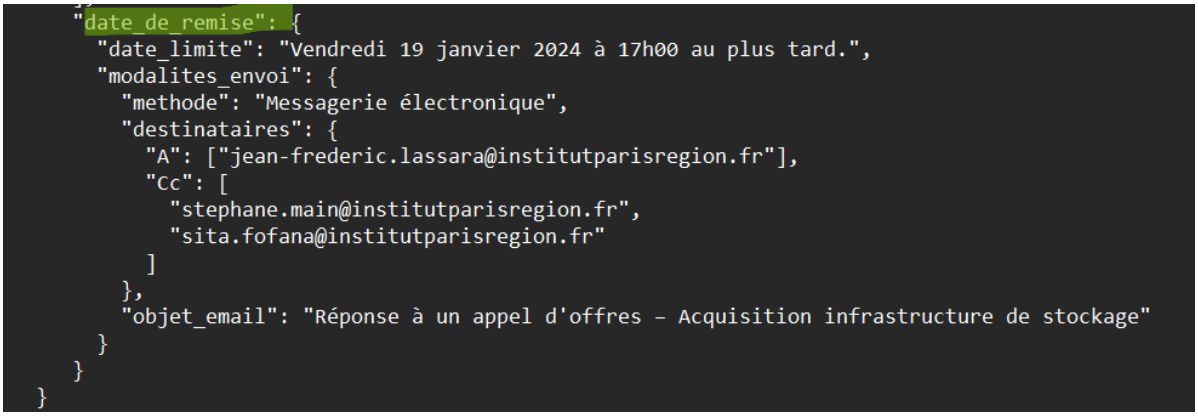

Figure 4.34: Structured extraction results generated by the proposed AI system part 5

The comparison between the generated outputs and the ground truth data enables the identification of **correctly extracted fields, partially correct extractions, missing information, and incorrectly inferred values**. This evaluation methodology improves the scientific validity of the proposed system and provides a reliable benchmark for measuring extraction quality and retrieval effectiveness.

4.6.4 Discussion of Results

The experimental results demonstrate the effectiveness of the proposed intelligent tender analysis system in extracting structured information from complex procurement documents. The system was evaluated using real-world tender documents and validated against manually extracted ground-truth data in order to assess extraction accuracy, retrieval relevance, and overall robustness.

A comparative analysis between a traditional RAG approach and the proposed enhanced field-oriented RAG architecture highlights a significant improvement in both extraction quality and contextual completeness.

The traditional RAG implementation was capable of extracting general information from the tender document. However, the generated output remained relatively limited, concise, and insufficiently structured for enterprise-level exploitation. Several extracted fields lacked detail, contextual relationships, and hierarchical organization. For example, the duration field only contained a single delivery date, while important project phases, maintenance durations, support conditions, and deployment schedules were omitted.

In contrast, the proposed system produced substantially richer and more detailed outputs. The introduction of field-oriented semantic retrieval enabled the system to target specific categories of tender information such as:

- Budget and financial requirements
- Project duration and deployment phases
- SLA and support conditions
- Team composition and responsibilities
- Financial offer structure
- Evaluation and scoring criteria

Instead of relying on a single global retrieval query, the system executed multiple specialized semantic queries independently. This strategy significantly improved retrieval precision and reduced the loss of relevant contextual information.

The generated results show that the proposed architecture successfully extracted highly detailed information that was either partially retrieved or completely absent in the traditional RAG output. For instance, the enhanced system identified:

- Detailed project phases and execution timeline
- Maintenance and support durations
- Internal and external stakeholder roles
- Structured financial offer requirements
- SLA constraints and availability metrics
- Technical qualification requirements
- Infrastructure deployment details

The comparison with the manually extracted ground truth also demonstrates a strong semantic alignment between the generated output and the expected tender information. In several sections, the proposed system even provided additional contextual details that were implicitly present in the document but not included in the manual extraction.

This improvement can be attributed to several key design choices implemented in the proposed architecture:

- Semantic embedding generation using SentenceTransformers
- Vector similarity retrieval using FAISS
- Field-oriented retrieval queries
- Overlapping token-based chunking strategy
- OCR-enhanced document processing using Gemini 2.5 Flash
- Strict prompt-engineering constraints for structured generation

The OCR module also demonstrated strong robustness when processing scanned and heterogeneous PDF documents. The Gemini-based OCR approach preserved document structure, including headings, tables, and bullet points, thereby reducing information loss commonly observed in traditional OCR systems.

The experimental evaluation confirms that the proposed field-oriented RAG architecture significantly outperforms a traditional retrieval-based approach in terms of completeness, contextual richness, and structured information extraction quality.

Overall, the implemented solution provides an effective, scalable, and enterprise-oriented framework for intelligent tender document analysis, capable of considerably reducing manual processing effort while improving the accessibility and exploitation of procurement information.

4.6.5 Challenges Encountered

During the development and experimentation phases of the intelligent tender analysis system, several technical and methodological challenges were identified. These challenges mainly relate to document quality variability, retrieval redundancy, computational efficiency, and consistency in large language model outputs. Each issue required targeted engineering solutions to ensure the robustness and reliability of the final system.

OCR Accuracy on Low-Quality Scanned Documents

One of the primary challenges was the variability in OCR performance when processing low-quality or scanned PDF documents. In such cases, the extracted text often contained missing characters, broken structure, or misaligned formatting, which negatively impacted downstream retrieval and extraction quality.

To mitigate this issue, a multimodal OCR approach based on the `gemini-2.5-flash` model was adopted. Unlike traditional OCR engines, this model leverages contextual understanding of document structure, allowing more accurate reconstruction of headings, tables, and structured content. In addition, a page-level detection mechanism was implemented, where pages with fewer than 50 extracted characters were automatically forwarded to the OCR pipeline.

Duplicate Retrieved Chunks

Another significant challenge was the presence of redundant or semantically identical chunks during the retrieval phase. Since multiple field-based queries may retrieve overlapping regions of the document, duplicate content frequently appeared in the final aggregated context.

To address this, a deduplication mechanism was implemented at two levels. First, during retrieval aggregation, previously seen chunks were filtered out. Second, after LLM generation, additional post-processing functions were applied:

```python
def deduplicate_lines(text):
    seen = set[Any]()
    out = []
    for line in text.split("\n"):
        line = line.strip()
        if line and line not in seen:
            seen.add(line)
            out.append(line)
    return "\n".join(out)

summary_text = deduplicate_lines(summary_text)
```

Figure 4.35: Deduplication mechanism

```python
def clean_markdown(md_text):
    lines = md_text.split("\n")
    cleaned = []
    seen_headers = set[Any]()

    for line in lines:
        line = line.strip()
        if not line:
            continue

        header = re.match(r"\*\*(.+?)\*\*", line)
        if header:
            h = header.group(1).strip()
            if h in seen_headers:
                continue
            seen_headers.add(h)

        cleaned.append(line)

    return "\n".join(cleaned)

final_output = clean_markdown(summary_text)
```

Figure 4.36: Markdown

This ensured that repeated semantic content did not degrade the clarity or conciseness of the final output.

Long-Document Processing Time

Processing large tender documents introduced computational overhead, particularly during embedding generation and FAISS indexing. The combination of OCR processing, tokenization, and vector encoding increased execution time for large-scale inputs.

To optimize performance, the system adopts a chunk-based processing strategy with controlled token limits (400 tokens with overlap of 80 tokens). This reduces memory

overhead while maintaining contextual continuity. Additionally, FAISS `IndexFlatIP` was used to ensure efficient similarity search even over large embedding spaces.

Prompt Consistency and Output Stability

Ensuring consistent output structure from the large language model represented another challenge. Without strict constraints, the model occasionally produced variations in formatting or included minor hallucinated elements.

This was addressed through carefully engineered prompt design, explicitly enforcing:

- Strict reliance on retrieved context only
- Prohibition of external knowledge injection
- Standardized Markdown output format
- Explicit handling of missing fields ("Non spécifié dans le document")

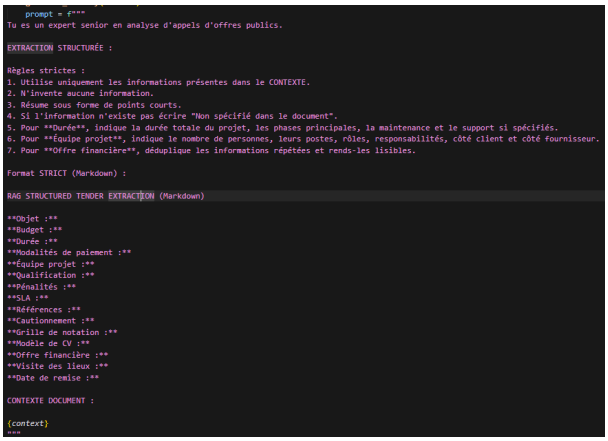

Figure 4.37: LLLM promt

The structured prompt design significantly improved output stability across different documents.

Retrieval Precision Optimization

Initial experiments revealed that a single generic retrieval query was insufficient to capture all relevant tender information. Important sections such as SLA conditions, financial structure, or evaluation criteria were sometimes partially retrieved or missed entirely.

To overcome this limitation, a field-oriented semantic retrieval strategy was introduced. Instead of relying on one global query, multiple domain-specific queries were defined (e.g., Budget, Duration, SLA, Team, Financial Offer). Each query independently retrieves relevant chunks from the FAISS index, and results are then merged with deduplication.

This approach significantly improved retrieval recall and ensured systematic coverage of all critical tender dimensions.

Conclusion of Challenges

Overall, the challenges encountered during system development were addressed through a combination of multimodal AI integration, retrieval optimization techniques, and strict prompt engineering. These improvements contributed significantly to the robustness, scalability, and accuracy of the final intelligent tender analysis pipeline.

4.6.6 Chapter Summary

This chapter presented the complete implementation of the proposed intelligent tender analysis system, detailing both its AI-driven backend pipeline and its integration within a full-stack web application.

It first described the system architecture and development environment, including the core programming languages, frameworks, and external libraries used to support document processing, semantic retrieval, and large language model integration. Particular emphasis was placed on the modular design choices that ensure scalability, maintainability, and flexibility in deployment.

The chapter then provided a detailed breakdown of the AI processing pipeline, covering all stages from raw PDF ingestion to final structured output generation. This included adaptive PDF parsing, OCR-based text extraction using a multimodal model, text preprocessing and normalization, token-based chunking, embedding generation using SentenceTransformers, and vector indexing using FAISS. The field-oriented semantic retrieval strategy was also introduced as a key enhancement over traditional retrieval approaches, enabling more precise and structured extraction of tender-specific information.

Furthermore, the integration of the system within the Django web platform was presented, highlighting the separation of concerns between the presentation layer, business logic, and retrieval components. The role of services-based architecture and modular design files was emphasized as a key factor in ensuring clean system organization.

Finally, the chapter discussed the challenges encountered during development and the corresponding solutions implemented to address issues such as OCR inaccuracies, retrieval redundancy, long-document processing, and prompt stability.

Overall, this chapter demonstrated how the combination of semantic search techniques, generative AI models, and web engineering practices results in a robust and scalable system for intelligent tender document analysis in real-world enterprise environments.

Bibliography

[1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson, 2023.

[2] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. 2007.

[3] Dragomir Radev et al. Introduction to the special issue on automatic summarization. *Computational Linguistics*, 2002.

[4] Tom Brown et al. Language models are few-shot learners. *NeurIPS*, 2020.

[5] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 1988.

[6] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

[7] Ashish Vaswani et al. Attention is all you need. In *NeurIPS*, 2017.

[8] Wenhui Wang et al. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 2020.

[9] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop*, 2004.

[10] Kishore Papineni et al. Bleu: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[11] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *ACL*, 2018.

[12] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[13] Darren Edge et al. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[14] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*, 2024.

[15] Teng Lin, Yizhang Zhu, Yuyu Luo, and Nan Tang. Srag: Structured retrieval-augmented generation for multi-entity question answering over wikipedia graph. *arXiv preprint*, 2025.

[16] Michail Dadopoulos, Anestis Ladas, Stratos Moschidis, and Ioannis Negkakis. Metadata-driven retrieval-augmented generation for financial question answering. *arXiv preprint*, 2025.

[17] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning (ECML)*, pages 137–142. Springer, 1998.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.