

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
SAAD DAHLAB UNIVERSITY OF BLIDA 1

Faculty of Sciences
Department of Computer Science



MASTER'S THESIS

Speciality : Smart Systems Engineering

Presented by :

Mihoubi Nahla Yasmine

Slamani Abdelmalek

THEME

**Development of a RAG-Based Legal Chatbot for Djezzy's
Compliance Services**

Publicly defended on **june**

Dr.	Melyara Mezzi	USDB	Thesis supervisor
Ms.	Nadine Belhadj	Djezzy	Internship Supervisor
Mr.	Walid Zoubir	Djezzy	Internship Supervisor

Cheriguene	USDB	Thesis examiner
Ferfara	USDB	Thesis examiner

Dedication

All praise is due to **Allah**, the Most Gracious, the Most Merciful, who granted me the strength, patience, and perseverance to complete this journey. Without His guidance and countless blessings, none of this would have been possible.

First and foremost, I dedicate this work to my beloved mother, **Souhila Boussouar**. Your *endless love, strength, and sacrifices* have been my rock and guiding light through every step of this journey. Thank you for *always believing in me*.

To my dear sister **Halla** thank you for your *constant encouragement, patience*, and for always being a source of motivation. I couldn't have done this without your support.

To my **father and brother** thank you for your *unwavering support, care*, and for always standing by me in quiet but meaningful ways.

To **Omar Remdani**, my Qur'an teacher your guidance and support were *instrumental as I memorized the Holy Qur'an in the same year as my Master's defense*. Your presence made this dual achievement possible.

To my dear friends **Tarek.B**, **Salah Eddine.T**, and **Mohamed.B** thank you for being there through the highs and lows of this journey.

Tarek and Mohamed, your *support, kindness, and friendship* gave me strength when I needed it most. **Salah**, thank you not only for your collaboration and support throughout these three years, but also for all the *unforgettable laughs*, and your unique ability to turn even the most stressful deadlines into *moments of fun*. You didn't just make the path more meaningful—you made it *entertaining*!

A special thank you to my friends: **Faisel.B**, **Yacin.H**, **Islam.H**, **Ayoub.C**, **Mourad.B**, **Adel.A**, **Nasim.B**, **Mohamed.G**, **Issam.H**, and **Idriss.M**.

To my **teachers and mentors** thank you for your *trust, guidance*, and the knowledge you've imparted throughout my studies.

To my teammate **Nahla Yasmine Mihoubi** thank you for standing by my side throughout the license, the master's program, and our internship. In both the *best times and the most difficult ones*, your support and friendship have meant everything to me. Thanks to you, *this project became a reality*.

And finally, to all my **friends and colleagues** thank you for being part of this journey and for making it so *meaningful*.

Abdelmalek Slamani

Dedication

I dedicate this work, first and foremost, to **God**, for granting me the strength, perseverance, and clarity throughout this journey.

To my beloved parents, **Mam** and **Dad**, thank you for your unconditional love, endless support, and unwavering belief in me. I am here because of you and the countless sacrifices you made to see me succeed.

To my dear brothers, **Akram** and **Chawki**, thank you for your encouragement and presence. And to Akram, your constant support, motivation, and faith in me have meant more than words can express.

To my entire family especially my grandmother, my aunts, cousins, and all those who stood by me during this journey your kindness, prayers, and encouragement have been deeply appreciated.

To my **teachers and mentors** thank you for your *trust, guidance*, and the knowledge you've imparted throughout my studies.

To my friends **Aya.D**, **Amira.B**, and **Yousra.B**, thank you for your friendship, your patience, and for brightening my university experience.

A special thank you to my partner **Abdelmalek** for being an exceptional partner throughout this academic journey, from university life to our internship and especially during the thesis. Your dedication, collaboration, and support were essential in making this achievement possible.

To the **IT Club Community**, thank you for providing an inspiring and dynamic environment that enriched both my academic and personal growth.

Nahla Yasmine Mihoubi

Acknowledgements

Primarily, we extend our deep and heartfelt appreciation to Almighty God (Allah). His boundless mercy and innumerable blessings have guided and sustained us at every stage of this journey. This work has been accomplished through His will and grace.

We express our sincere appreciation to our academic supervisor, **Melyara Mezzi**, for her consistent guidance, insightful feedback, and steadfast support throughout the development of this thesis. Her supervision has significantly contributed to the progress of this work and to our growth as students.

We are deeply grateful to our intern supervisor **Ms. Nadine Belhadj** for her remarkable dedication, availability, and invaluable guidance during our time at *Djezzy*. Her active involvement was essential to the success of this endeavour. We also extend our deepest thanks to **Mr. Walid Zoubir** and **Mr. Rafik Bougdour** for their support and collaboration, as well as to the entire *Legal Department at Djezzy*, whose collective expertise and openness greatly enriched our experience.

We express our gratitude to the *IT Department*, especially to **Abdelkrim Nache** and **Fayçal Benaissa**, for their technical assistance and continuous support throughout the internship.

We are profoundly thankful to our parents and family, whose unconditional love, patience, and belief in us have been our greatest source of strength and motivation.

Finally, we would like to extend our appreciation to the companies and communities that have contributed to our learning journey: **Djezzy**, **Mare Nostrum Advising Algérie**, **Deeplyfe**, and the **IT Club Community**, whose environments have inspired innovation, collaboration, and continuous personal and professional growth.

Abstract

In the context of increasing legal complexity and digital transformation, legal departments in large organizations face increasing demands for timely and accurate information. This thesis addresses the problem of improving access to legal and regulatory knowledge for employees at Djezzy, a major telecommunications company in Algeria. The main objective is to develop an intelligent chatbot system capable of answering legal questions in French, with a focus on compliance, data protection, and internal policies.

To achieve this, we designed a Retrieval-Augmented Generation (RAG) system that combines Semantic Document Retrieval and Natural Language Generation using Large Language Models (LLMs). The architecture is built around a pipeline that indexes legal texts into a vector database and uses local LLMs via Ollama to generate responses based on retrieved content.

The evaluation was, successfully, carried out using automatic and human-based methods. We used semantic similarity, ROUGE lexical metrics, BLEU, and BERTScore to measure response quality and contextual relevance. In addition, a Djezzy legal expert reviewed the output, and large language models were also used to simulate judgement during evaluation.

The chatbot was implemented in a web application called JuriBot. This work demonstrates the viability of integrating RAG-based systems into enterprise legal services, offering a valuable tool for improving compliance efficiency and access to legal information.

Keywords: Legal Chatbot; Legal Assistant; RAG Technique, Large Language Models, Vector Search.

ملخص

في ظل تعقيد الإطار القانوني والتحول الرقمي المتسارع، تواجه الإدارات القانونية في المؤسسات الكبرى تحديات متزايدة لتوفير معلومات دقيقة وفي الوقت المناسب. يتناول هذا البحث مشكلة تحسين الوصول إلى المعرفة القانونية والتنظيمية لموظفي شركة Djezzy، وهي من أكبر شركات الاتصالات في الجزائر. الهدف الرئيسي هو تطوير روبوت دردشة ذكي قادر على الإجابة على الأسئلة القانونية باللغة الفرنسية، مع التركيز على الامتثال، حماية البيانات، والسياسات الداخلية.

لتحقيق ذلك، قمنا بتصميم نظام قائم على تقنية الاسترجاع المعزز بالتوليد (Retrieval-Augmented Generation - RAG)، حيث يجمع بين استرجاع الوثائق بطريقة دلالية وتوليد النصوص باستخدام نماذج لغوية كبيرة (LLMs). يعتمد النظام على معالجة النصوص القانونية وتخزينها في قاعدة بيانات متجهية، مع توليد الإجابات بواسطة نماذج محلية عبر منصة Ollama.

تم تقييم النظام باستخدام طرق آلية وبشرية. شملت التقييمات التشابه الدلالي (باستخدام جيب التمام)، ومقاييس لغوية (ROUGE و BLEU)، و BERTScore لقياس جودة الردود ومدى ملاءمتها للسياق. كما قام خبير قانوني من شركة Djezzy بمراجعة النتائج، وتم استخدام النماذج اللغوية لمحاكاة التقييم القانوني.

تم دمج روبوت الدردشة في تطبيق ويب يدعى JuriBot. ويظهر هذا العمل جدوى دمج أنظمة RAG في الخدمات القانونية المؤسسية، مما يوفر أداة فعالة لتحسين الامتثال وسهولة الوصول إلى المعلومات القانونية.

الكلمات المفتاحية: روبوت دردشة قانوني - الاسترجاع المعزز بالتوليد (RAG) - النماذج اللغوية الكبيرة - الامتثال القانوني - البحث المتجهي - معالجة اللغة القانونية

Résumé

Dans un contexte de complexité juridique croissante et de transformation numérique, les départements juridiques des grandes entreprises font face à une demande accrue d'informations précises et accessibles en temps réel. Ce mémoire traite du problème d'accès à la connaissance juridique et réglementaire pour les employés de Djezzy, un opérateur télécom majeur en Algérie. L'objectif principal est de développer un système de chatbot intelligent capable de répondre à des questions juridiques en français, avec un accent particulier sur la conformité, la protection des données et les politiques internes.

Pour cela, nous avons conçu un système basé sur l'architecture Retrieval-Augmented Generation (RAG), combinant la recherche sémantique de documents et la génération de texte à l'aide de grands modèles de langage (LLMs). Le pipeline mis en place indexe des textes juridiques dans une base vectorielle, puis utilise des modèles locaux via Ollama pour générer des réponses pertinentes à partir du contenu récupéré.

L'évaluation du système a été réalisée à l'aide de méthodes automatiques et humaines. Nous avons utilisé la similarité sémantique (cosinus), des métriques lexicales (ROUGE, BLEU) et BERTScore pour mesurer la qualité des réponses et leur pertinence contextuelle. De plus, un expert juridique de Djezzy a analysé les réponses générées, et des modèles de langage ont été mobilisés pour simuler un jugement automatisé.

Le chatbot a été intégré dans une application web nommée JuriBot. Ce travail montre la faisabilité d'intégrer des systèmes RAG dans les services juridiques d'entreprise, en apportant un outil utile pour améliorer l'efficacité en matière de conformité et l'accès à l'information juridique.

Mots Clés : Agent Conversationnel Juridique ; Assistant Juridique ; Technique de Récupération Augmentée par la Génération ; Grands Modèles de Langage ; Recherche vectorielle

Contents

List of Figures	i
List of Tables	iii
General Introduction	1
1 State of the Art about Chatbots and Language Models	7
1.1 Introduction	7
1.2 Chatbot	7
1.2.1 Definition	7
1.2.2 Chatbot Technologies	8
1.2.3 Comparison of Chatbot Types	11
1.3 Language Modeling	13
1.3.1 Definition	13
1.3.2 History Language Models	14
1.3.3 Statistical Language Models	15
1.3.4 Neural Language Models	15
1.3.5 Pre-trained Language Models	16
1.3.6 Large Language Models	16
1.3.7 Comparison of Language Models	16
1.4 Related Work of Chatbot Applications in Domain-Specific Contexts	18
1.4.1 Study D: Legal Chatbot in India	19
1.4.2 Comparative Analysis	20
1.4.3 Summary	20
1.5 Conclusion	21
2 Design of the RAG-based Legal Chatbot	22
2.1 Introduction	22
2.2 System Overview	22

2.3	Data Collection and Preparation	24
2.3.1	Data Sources	24
2.3.2	Challenges and Data Preprocessing	25
2.3.3	Dealing with Graphs	27
2.3.4	Data preparation and organization	28
2.4	Vector Storage and Indexing	30
2.4.1	Document Chunking	31
2.4.2	Embedding Generation	35
2.4.3	Vector Storage	36
2.4.4	Indexing Algorithms	40
2.5	Contextual Retrieval	43
2.5.1	Retrieval Process	43
2.5.2	Retrieval Strategies	44
2.6	Answer Generation	45
2.6.1	Local Deployment with Ollama	46
2.6.2	Language Model Selection	46
2.6.3	French Language Performance Benchmarks	48
2.6.4	LLM Configuration Parameters	49
2.6.5	LLM Parameters	49
2.7	Conclusion	50
3	Implementation and Evaluation	51
3.1	Introduction	51
3.2	Development Environment	51
3.3	Evaluation Metrics Definition	53
3.3.1	Semantic Similarity Metrics	54
3.3.2	Lexical Overlap Metrics	54
3.3.3	Contextual Quality Metrics	55
3.3.4	System Performance Metrics	56
3.4	Evaluation Results	57
3.5	Analysis and Discussion Metric result	64
3.6	LLM as judge	65
3.7	Human evaluation by Legal Experts at Djezzy	66
3.8	Conclusion of Evaluation	67
3.9	System Demonstration JuriBot Web Application for a Legal RAG Chatbot	69
3.9.1	User Authentication and Access Control	69
3.9.2	Administrative Interface for Content Management	70

3.10 System Workflow and Processing Pipeline	72
3.11 Conclusion	74
General conclusion	75
Bibliography	77

List of Figures

0.1	Structure of Legal Department at Djazzy	2
1.1	ELIZA Chatbot [Wang, 2024].	8
1.2	Retrieval-based architectural model [Sharma et al., 2021].	9
1.3	Virtual Assistant Chatbots Architecture (Google Assistant).	10
1.4	Generative chatbot architectural model	11
1.5	History of Chatbot - Timeline	13
1.6	Types of Language Modeling.	14
1.7	History and development of language models [Wang et al., 2024].	15
2.1	Global system pipeline for RAG	23
2.2	Structure of the dataset	24
2.3	PDF transformation process with PyMuPDF4LLM	27
2.4	Examples of graphs found in the documents	28
2.5	Relational database schema for organizing legal documents and FAQs	30
2.6	visualization of vector store pipeline	31
2.7	Example result of chunking.	34
2.8	Example of 3D Text Embeddings	35
2.9	Database and vector storage schema in PostgreSQL and pgvector extaen- tion	39
2.10	Visualization of the HNSW graph-based structure [Zilliz, 2024].	41
2.11	Illustration of IVF clustering and search process [UnfoldAI, 2024].	41
2.12	Illustration of FI	42
2.13	Vector-Based Similarity Retrieval using HNSW in pgvector	45
2.14	Illustration of the RAG Answer Generation Pipeline	46
3.1	BERTScore Computation Process [Hugging Face, 2019]	56
3.2	Solution Evaluation Steps.	57
3.3	Cosine Similarity Scores visualization	59
3.4	BLEU Score visualization	60

3.5	ROUGE-L by model and index setup	61
3.6	Graphe BERTScore per configuration and models	62
3.7	Average Response Time Comparison	63
3.8	Token Generation Distribution by Model	64
3.9	Login Interface	69
3.10	Chatbot Interface (juribot)	70
3.11	Document List in Data Manager	71
3.12	Add Document Form	72
3.13	System Workflow and Processing Pipeline of the JuriBot Legal Chatbot . .	73

List of Tables

1.1	Comparison of Chatbot Types	12
1.2	Comparison of Different Types of Language Models	17
1.3	Comparison of Domain-Specific Chatbot Studies	20
2.1	Format and characteristics of the document dataset	25
2.2	Comparison of document extraction strategies	26
2.3	Structure of the <code>Domain</code> table	28
2.4	Structure of the <code>Document</code> table	29
2.5	Structure of the <code>Question_Answer</code> table	29
2.6	Overview of Chunking Techniques for RAG	32
2.7	Embedding model specifications	36
2.8	Comparison of representative vector storage systems for RAG-based applications.	38
2.9	Schema of <code>VectorDB_QA</code> Table	40
2.10	Schema of <code>VectorDB_DOC</code> Table	40
2.11	Comparison of Indexing Algorithms	42
2.12	Recommended HNSW Parameters Based on Application Needs	43
2.13	Retrieval strategies used in RAG systems.	44
2.14	Comparative summary of the selected LMs.	48
2.15	Performance of LM on French Benchmarks	49
2.16	Configuration parameters for LLM in a RAG system [OpenAI, 2025].	50
3.1	Summary of the technical specifications of the machines used	52
3.2	Development Tools and Technologies	53
3.3	Overview of Evaluation Metrics Used in the System	57
3.4	Cosine Similarity Scores Across RAG Configurations and Language Models	58
3.5	Lexical Precision Analysis BLEU Score per Configuration	59
3.6	ROUGE-L Score Across RAG Configurations and Language Models	60
3.7	Contextual Quality Evaluation BERTScore	61

3.8	Average Response Time Analysis (seconds)	62
3.9	Average Token Count score	63
3.10	Comparison of LLMs Based on <i>Prompt 1</i> and <i>Prompt 2</i> Evaluation Metrics	66
3.11	Jurist Ranking of Model Responses	67
3.12	Summary of Evaluation Insights Across Methods	68

List of Acronyms

AI Artificial Intelligence. 1, 4, 7, 16

API Application Programming Interface. 10, 12

BAC Baccalaureate Exam. 48

FI Flat Index. i, 42

GPQA Graduate-level Problem Question Answering. 48

HNSW Hierarchical Navigable Small Worlds. i, iii, 5, 38–43

IFEval Instruction Following Evaluation. 48

IVF Inverted File Flat Index. i, 38, 41, 42

LLM Large Language Models. iv, , 6, 13, 16–18, 45, 47–49, 53, 65, 66, 68

LM Language Models. iii, 1, 5, 6, 14, 16–18, 46–49

ML machine learning. 9, 10

MOE Mixture-of-Experts. 47

NLM Neural Language Models. 13, 15, 17

NLP Natural Language Processing. 7, 9, 12, 13, 16, 17, 32

NLU Natural Language Understandin. 18

OTA Optimum Telecom Algeria. 1

PLM Pre-trained Language Models. 13, 16, 17

RAG Retrieval-Augmented Generation. i, iii, , 1, 18–23, 32, 36, 43–47, 49–51, 53, 57, 58, 60, 69

RNN Recurrent Neural Network. 12

SLM Statistical Language Models. 13, 15, 17

SWA Sliding Window Attention. 47

General Introduction

General Background

The digital revolution of organizations, along with the rapid advancement of Artificial Intelligence (AI), has profoundly altered the management of information, especially in critical areas like legal compliance. In large companies, legal departments are increasingly burdened with complex regulatory frameworks, substantial documentation, and increasing internal demand for legal advice. Djazzy, an important telecommunications provider in Algeria, reveals this tendency.

This research aims to create and construct an intelligent legal compliance assistant to maintain continuous regulatory compliance and improve internal communication with the legal department. The proposed solution is a chatbot that utilizes Retrieval-Augmented Generation (RAG), which allows it to retrieve relevant legal documents and generate context-aware responses using advanced Language Models (LM).

Presentation of Host Organization

This project is conducted in collaboration with Djazzy, a leading telecommunications operator in Algeria . Established in 2001 as Optimum Telecom Algeria (OTA), Djazzy offers mobile and internet services to a broad customer base across the country. Until 2022, the company was jointly owned by the Algerian government (51%) and VEON (49%). However, in 2022, Djazzy became wholly owned by the Algerian state and has made significant advancements in digital innovation, including the successful deployment of 4G and 5G networks.

Djazzy's focus on customer service is evident in its wide network of service centers and round-the-clock support. As a key player in Algeria's telecommunications industry, Djazzy provides an ideal environment for the development of an AI-driven legal assistant, aiming to improve the management and accessibility of legal compliance texts for its employees and legal teams.

Legal Department Structure and Role

The legal department at Djezzy is structured into five main divisions, as illustrated in Figure 0.1 :

- **Disputes and Litigation:** Managing conflicts and legal disputes.
- **Legal Contracts:** Drafting, managing, and validating contracts.
- **Legal Intelligence and Corporate Affairs:** Legislative monitoring and strategic legal analysis.
- **Legal Compliance:** Ensuring adherence to laws and regulations.
- **Legal Security Relations:** Data protection and compliance with legal security regulations.

These divisions work closely together to ensure the company's legal sustainability and compliance.

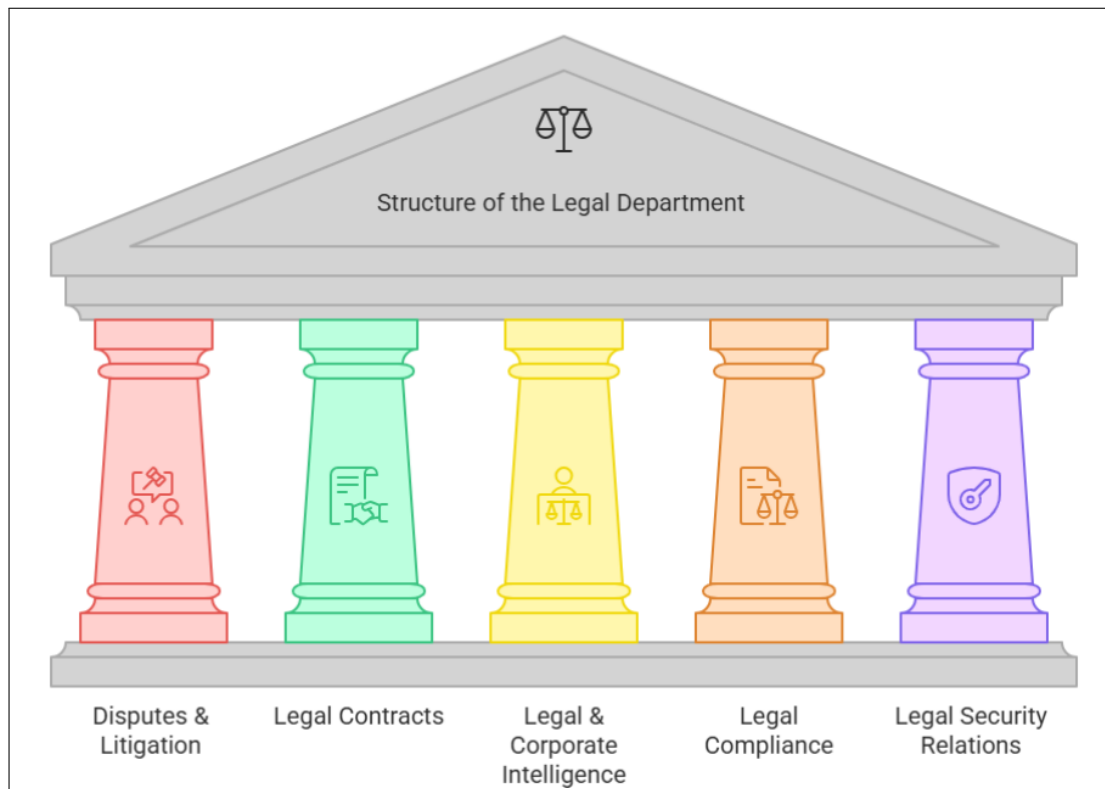


Figure 0.1: Structure of Legal Department at Djezzy

Importance of the Legal Department

The legal department holds a strategic position within DJezzy, particularly for:

- **Continuous regulatory compliance:** Ensuring the company's compliance with all local and international legal necessities.
- **Protecting the company from legal risks:** Anticipating disputes and managing legal risks effectively.
- **Managing contractual relationships:** Supervising all contracts, including those with customers, suppliers, and external partners.
- **Monitoring legislative changes:** Keeping track of legal developments to ensure the company's adaptability to regulatory changes.

Challenges Faced by the Legal Department

Despite its importance, some challenges are facing the legal department::

- **High Workload:** The team is often overwhelmed by repetitive tasks, such as answering routine legal queries and reviewing contracts.
- **Delays in Response Times:** Slow to respond to basic legal questions or compliance matters raised by employees from other departments.
- **Complexity of Regulations:** Constantly evolving laws, such as those governing data protection, require ongoing monitoring and adaptation.
- **Information Management:** Difficulty maintaining consistent and accessible legal knowledge across the organization.
- **Limited Focus on Strategic Tasks:** Time spent on repetitive tasks reduces the department's capacity to focus on high-value, strategic activities.

These challenges emphasize the need for innovative solutions to enhance efficiency and support the department's critical functions.

Problem Statement

Completing legal enquiries is painstaking and complicated. With the workload, it becomes harder to respond in a timely manner due to legal regulations. This negatively affects the focus on strategic tasks. This form of inefficiency can lead to greater legal risks and breaches in compliance.

The principal issue dealt with in this research is phrased as follows:

“How can Djazzy legal department enhance its functional processes with regard to regulatory compliance in fast and constantly changing legal circumstances?”

In order to work on this problem, this project investigates on how an AI powered chatbot can automate the legal knowledge management process.

Key Research Challenges

There are multiple problems to solve when creating a legal chatbot for Djazzy’s compliance:

- **Updating Legal Information:** Any legal changes must be added on the chatbot’s knowledge base immediately.
- **Internal Platforms Integration:** The chatbot should be connected to Djazzy’s platforms (Intranet, CRM, ERP,...).
- **Compliance of Data Regulations:** The system shall follow the Algerian law and international compliance regulations
- **Extracting Legal Terminology:** Different departments use distinct vocabularies, making it essential for the chatbot to accurately recognize and interpret legal terms

Research Question

Supporting research questions include:

- In what ways can automation facilitate the undertaking of legal compliance management tasks?
- What should be the best approaches to building the structure of an AI based legal knowledge system?
- How is the integration of an AI chatbot into Djazzy corporate’s ecosystem carried out?

- What protective measures need to be put in place to safeguard legal information contained in AI systems?

Research Objectives

In order to tackle these difficulties, a legal compliance chatbot for Djezzy powered by Ai is the centerpiece of this study. The goals of the study are:

- Construct a chatbot that can accurately respond to simple, repetitive legal questions.
- Develop a dynamic knowledge base that updates legal information automatically.
- Ensuring the security and confidentiality of Djezzy’s legal documents within the chatbot’s knowledge base

Research Conception

The methodology followed in this work is based on applied research and includes the following stages:

1. **Data Collection and Preprocessing:** A legal corpus was assembled from internal documents, including policies, regulations, and a sheet provided by Djezzy’s legal department. Text extraction and structuring were performed using PyMuPDF4LLM.
2. **Vectorization and Indexing:** The texts were split using recursive chunking techniques and encoded into semantic vectors using the CamemBERT model. Vectors were stored in a PostgreSQL database extended with the `pgvector` module, indexed using the Hierarchical Navigable Small Worlds (HNSW) algorithm.
3. **RAG-based System Design:** using LangChain to connect all component document retrieval and response generation. LM were deployed locally via Ollama, and the user interface was developed using Django.
4. **System Evaluation:** Multiple LLMs were evaluated using metrics such as Precision, Recall, F1-score, and BERTScore. Human evaluation was also conducted by legal experts to assess the accuracy and relevance of the chatbot’s answers.

This methodology ensures a technically sound and business-relevant solution aligned with the needs of Djezzy’s ecosystem.

Thesis Organization

This thesis is organized into three main chapters:

- **Chapter 1 : State of the Art:** Reviews existing chatbot technologies and LM, and compares domain-specific chatbot implementations, including legal use cases.
- **Chapter 2 : Solution Modeling:** Describes the proposed RAG-based architecture, including data preprocessing, vectorization, storage, indexing, and answer generation.
- **Chapter 3 : Implementation and Evaluation:** Details the technical implementation, presents experimental results, compares Large Language Models (LLM) performance, and includes feedback from human evaluators.

The thesis concludes with a general conclusion summarizing the contributions of the work and proposing future directions for improvement.

Chapter 1

State of the Art about Chatbots and Language Models

1.1 Introduction

In this chapter, we explore the evolution and current state of chatbot technologies and language models, which form the foundation of modern conversational AI. We start by defining what a chatbot is and examining different types, from rule-based systems to advanced generative models. We then delve into language modeling, tracing its development from statistical methods to large-scale neural networks. By understanding these advancements, we gain insight into how AI-powered chatbots are transforming interactions across industries.

1.2 Chatbot

In this section, a presentation about Chatbot definition, technologies, history, and types is provided.

1.2.1 Definition

A chatbot is a "computer program designed to simulate conversation with human users, especially over the Internet." [[Adamopoulou and Moussiades, 2020](#)] These systems use Natural Language Processing (NLP) and sentiment analysis to communicate in human language, either through text or oral speech, with humans or other chatbots.

1.2.2 Chatbot Technologies

Chatbot development has been shaped by four primary technological paradigms, each tailored to specific use cases and levels of complexity [Hussain et al., 2019].

1.2.2.1 Rules-based chatbots

Rules-based chatbots use conditional logic to create automated conversation flows. These bots act as interactive where a conversation designer programs predefined combinations of questions and answers so the chatbot can understand the user's input and respond accurately.

One classic example of a rule-based chatbot is ELIZA, developed in the 1960s. ELIZA mimicked a psychotherapist by using pre-programmed patterns and keywords to generate responses. It was groundbreaking for its time but also highlighted the limitations of rule-based systems [Adamopoulou and Moussiades, 2020].

These chatbots are easy to train and work well for straightforward, predefined questions. However, they struggle with complex or unexpected queries. Since their responses depend entirely on pre-written content, they often get stuck when faced with unanticipated user requests.

When a rules-based chatbot cannot grasp the user's input, it may miss key details or repeatedly ask for the same information, leading to a frustrating user experience. In many cases, it transfers the user to a live agent for help. When this option is not available, the chatbot risks becoming a gatekeeper, further frustrating the user.

Figure 1.1 below combines a schema and an interface example to illustrate how ELIZA, a rule-based chatbot, functions.

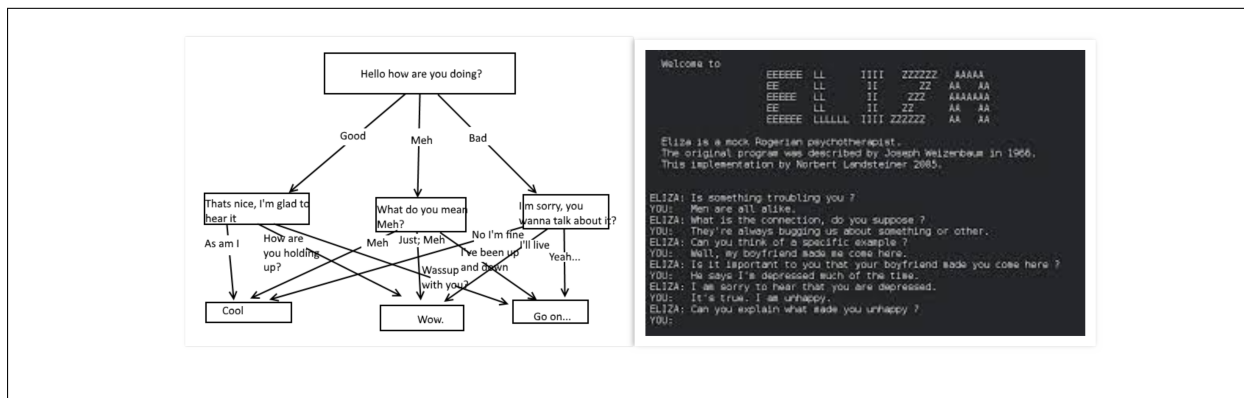


Figure 1.1: ELIZA Chatbot [Wang, 2024].

1.2.2.2 Retrieval-based chatbots

Retrieval-based chatbots use machine learning (ML) and NLP to select the best responses from a predefined database. These systems analyze user queries and rank possible responses to determine the most appropriate one.

for example IBM Watson Assistant, which gained recognition after IBM Watson’s success in winning the TV quiz show Jeopardy! in 2011. Watson Assistant is used in various industries for customer support, healthcare, and enterprise solutions. By leveraging a knowledge base and advanced NLP techniques, it provides contextually relevant answers to user queries.

Retrieval-based chatbots are reliable and ensure accurate responses as all outputs are predefined. They can handle broader scenarios compared to rule-based systems, but their performance is limited by the quality and coverage of their database.

Figure 1.2 shows the basic architecture of the Retrieval based chatbot system. Here, a user utters an input in the form of a query, the system then starts searching for a response.

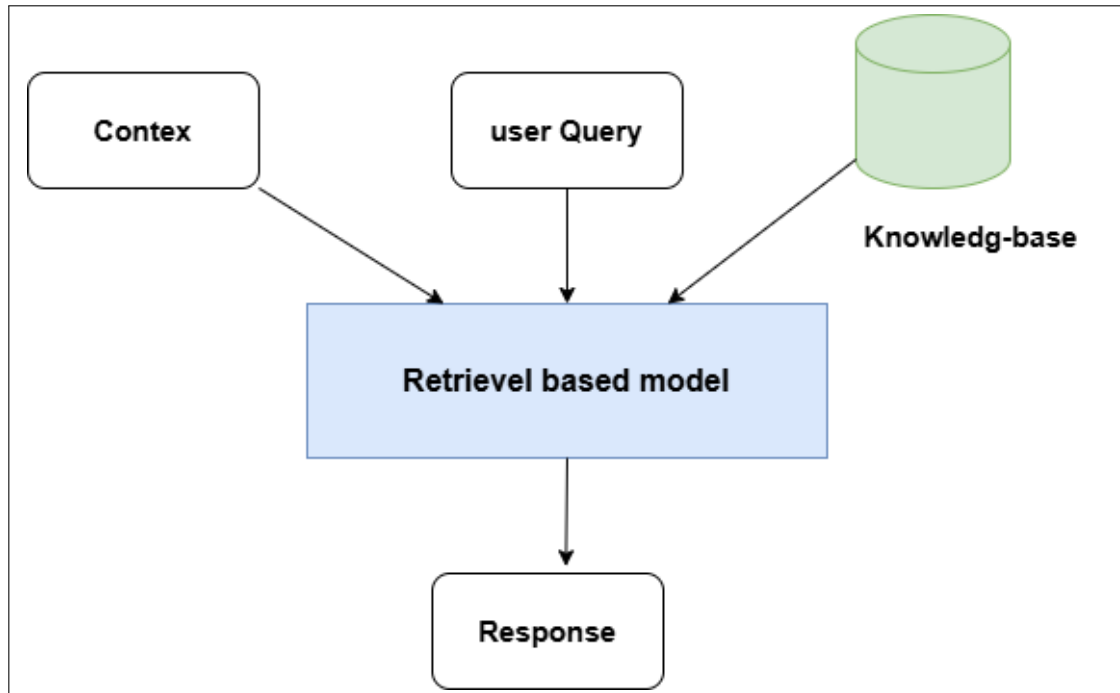


Figure 1.2: Retrieval-based architectural model [Sharma et al., 2021].

1.2.2.3 Virtual Assistant Chatbots

Virtual assistants like Siri, Alexa, and Google Assistant represent a specialized category of chatbots. They integrate voice recognition, natural language understanding, and context awareness to assist users with various tasks. These systems combine advanced AI models

with Application Programming Interface (API) to provide personalized responses and control devices.

Siri uses on-device ML alongside cloud-based AI to process commands. Alexa employs skills to extend functionality, making it versatile for home automation and entertainment. Google Assistant integrates seamlessly with Google's ecosystem for personalized recommendations and task management [Kumar S and Sheshadri, 2024].

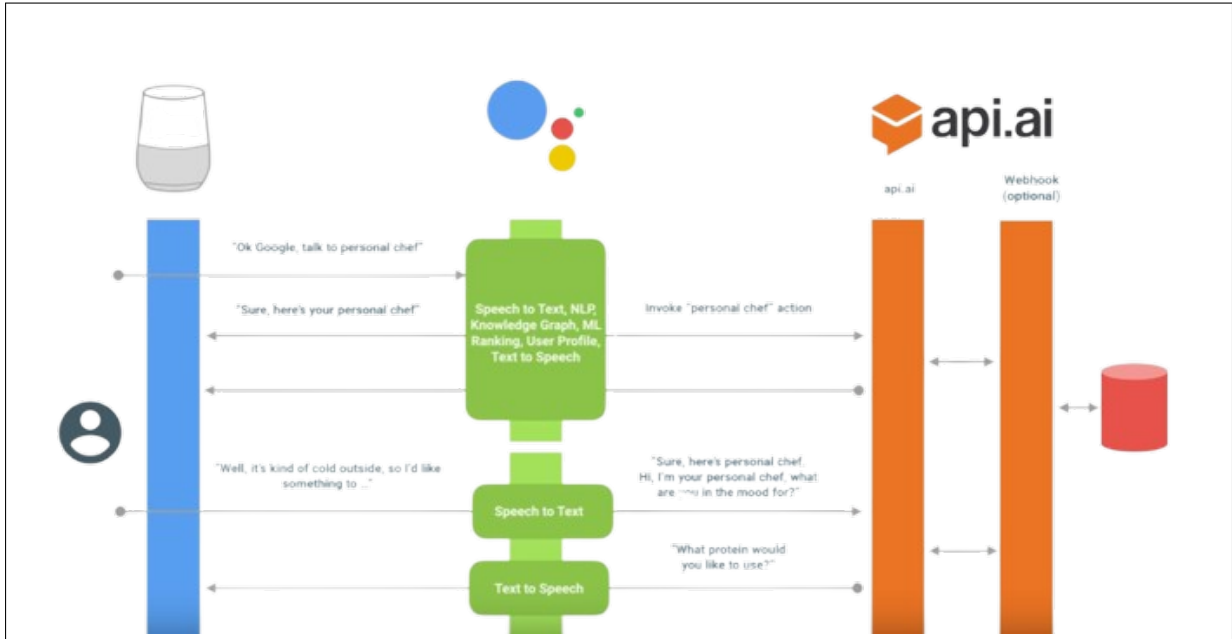


Figure 1.3: Virtual Assistant Chatbots Architecture (Google Assistant).

1.2.2.4 Generative Chatbots

Generative chatbots use transformers to dynamically create new responses. Unlike retrieval-based systems, these chatbots do not rely on predefined templates. They are trained on large datasets and can generate coherent, contextually relevant replies.

Modern AI systems like ChatGPT, Google Bard, and Gemini fall under this category. These chatbots are highly flexible and capable of open-domain conversations, making them suitable for creative and contextually rich interactions. However, they require significant computational resources and may produce irrelevant or biased responses if not properly trained [Dam et al., 2024].

Figure 1.4 illustrates the architecture of a generative chatbot. The generative model is trained on a dataset and, once deployed, takes user queries as input to produce relevant responses.

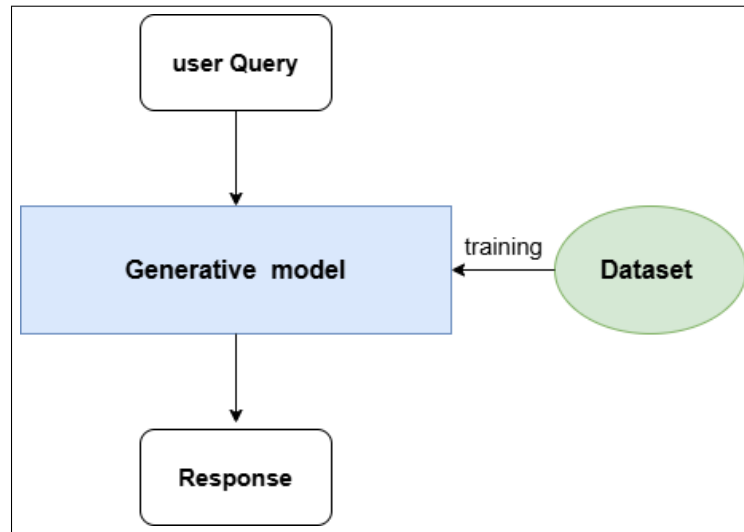


Figure 1.4: Generative chatbot architectural model

1.2.3 Comparison of Chatbot Types

Table 1.1 compares four different types of chatbots: Rule-Based, Retrieval-Based, Virtual Assistant, and Generative Chatbots. Each chatbot type is evaluated across various characteristics such as operational mechanism, system complexity, adaptability, response quality, data dependency, computational requirements, use cases, and limitations.

Table 1.1: Comparison of Chatbot Types

Feature	Rule-Based Chatbots	Retrieval-Based Chatbots	Virtual Assistant Chatbots	Generative Chatbots
Operational Mechanism	Follows pre-defined rules, logic, and pattern matching.	Matches input with predefined responses based on similarity.	Integrates voice recognition, NLP, and context-awareness for task management.	Generates responses using deep learning models (e.g., Recurrent Neural Network (RNN), Transformer).
Complexity	Low complexity, rule-based logic.	Moderate complexity with NLP and retrieval models.	High complexity, combining multiple AI subsystems.	High complexity, requiring deep learning models and large datasets.
Adaptability	Low adaptability; rigid to inputs.	Moderate adaptability based on data quality.	High adaptability; improves through user interaction.	High adaptability; can generate novel responses.
Response Quality	Simple, structured, suited for closed domains.	High quality, limited to the knowledge base.	Contextually relevant, personalized, and task-specific.	Variable; may lack coherence but can be fluent.
Data Dependency	Minimal data; relies on pre-defined rules.	High dependency on a curated response database.	Very high; relies on continuous data and external APIs.	Very high; requires large labeled datasets.
Computational Cost	Low; minimal processing power.	Moderate; requires NLP and database queries.	High; requires real-time speech and language processing.	High; computationally expensive for training and inference.
Use Cases	FAQs, simple customer service.	Customer support, information retrieval.	Personal assistants (e.g., Siri, Alexa), smart home automation.	Open-domain conversations, creative AI agents.
Limitations	Limited scalability and flexibility.	Limited by the scope of the database.	Ecosystem dependent, limited by task scope.	May generate irrelevant responses; data-intensive.

Figure 1.5 illustrates the historical timeline of chatbot development, starting with Turing's foundational ideas and progressing to modern large language models like ChatGPT.

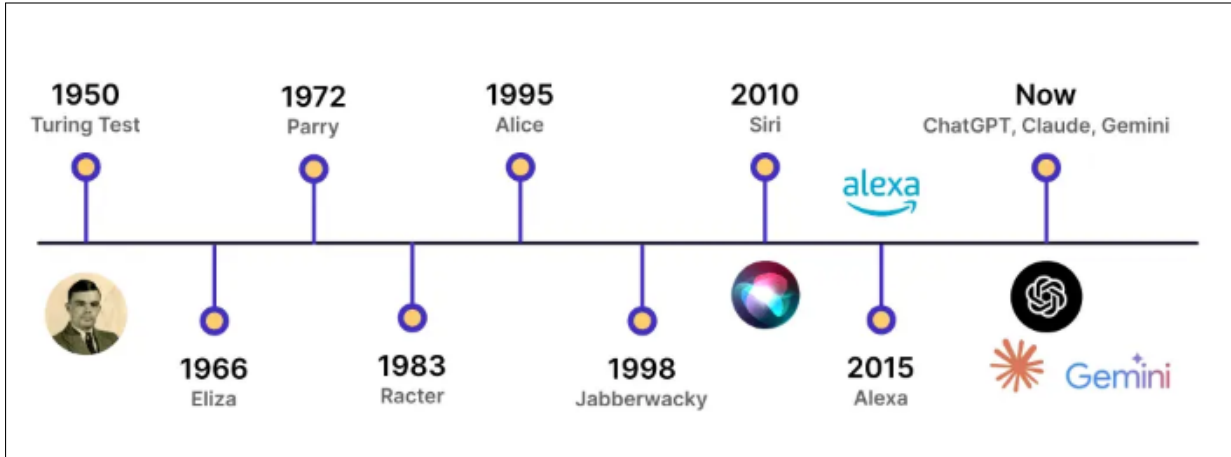


Figure 1.5: History of Chatbot - Timeline

1.3 Language Modeling

In this section, a concise overview of the definition, historical development, and various types of language models is presented, with particular emphasis on large language models.

1.3.1 Definition

Language Models in NLP are probabilistic and statistical methods used to determine the likelihood of a given sequence of words occurring in a text based on previous words. It helps predict which word is most likely to come next in a sentence.

As illustrated in Figure 1.6, language models can be categorized into different types, including Statistical Language Models (SLM), Neural Language Models (NLM), Pre-trained Language Models (PLM), and LLM. These models are widely used in applications such as speech recognition, machine translation, sentiment analysis, and text generation.

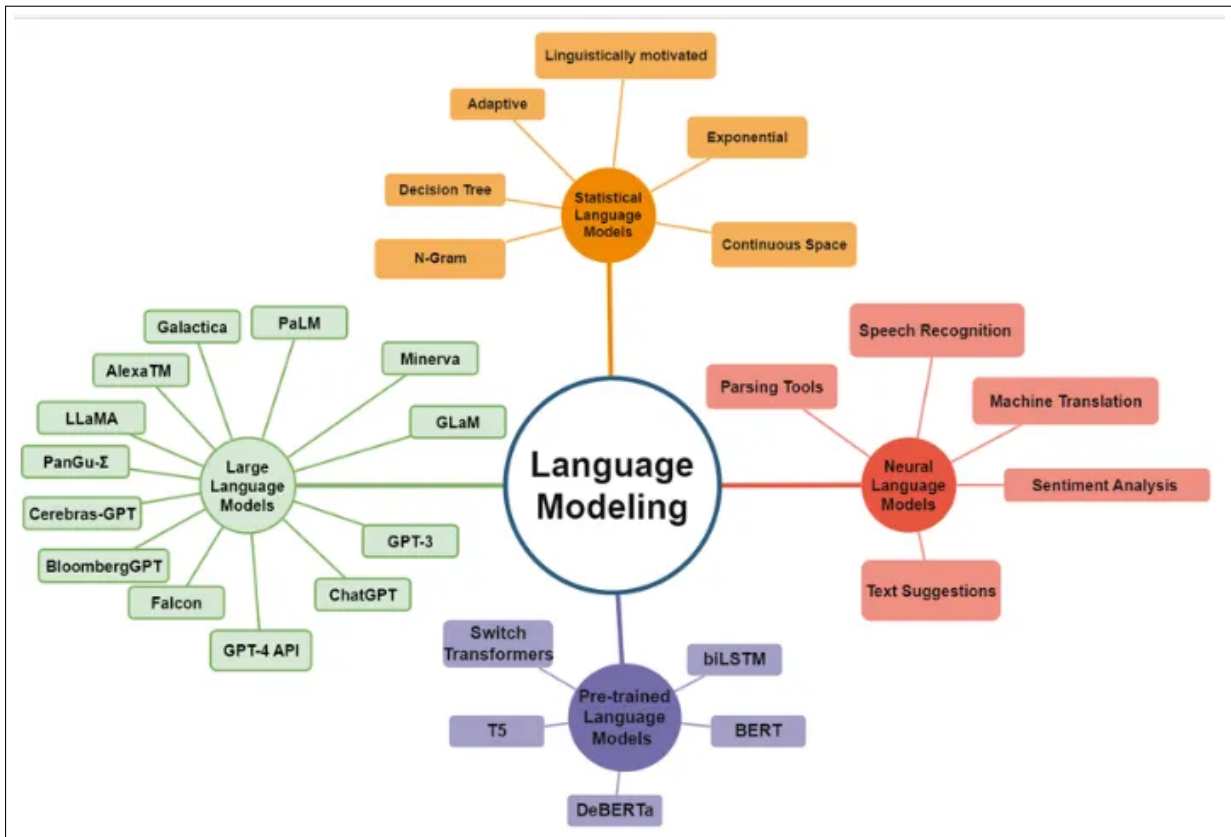


Figure 1.6: Types of Language Modeling.

1.3.2 History Language Models

The developmental stages of Language Models include Statistical Language Models , Neural Language Models , Pre-trained Language Models , and Large Language Models .

Figure 1.7 provides a visual representation of the progression and milestones in the development of LM .

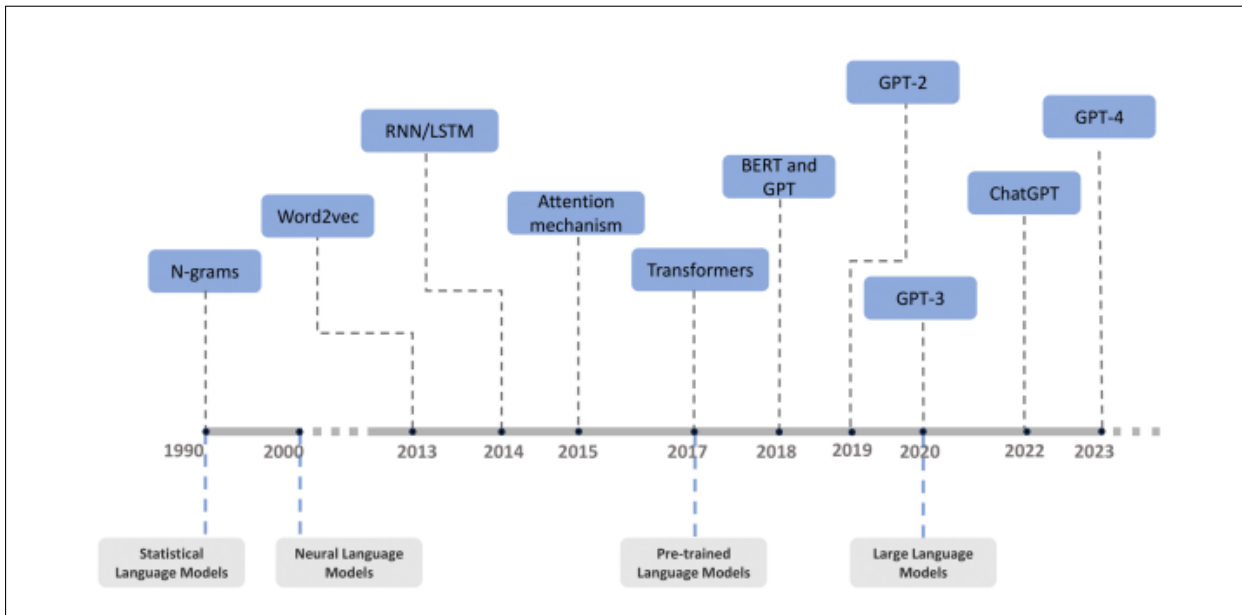


Figure 1.7: History and development of language models [Wang et al., 2024].

1.3.3 Statistical Language Models

SLMs emerged in the 1990s as probabilistic frameworks to model the likelihood of sentences appearing in text. These models calculate the probability of a sentence using *n-gram approaches*, where each word depends on the preceding $n - 1$ words. For example, in a sentence like “*I am very happy*”, the probability of the entire sentence is computed by considering conditional probabilities such as $P(\text{happy} \mid \text{I, am, very})$. However, SLMs face challenges due to the exponential growth of possible word combinations with larger n , leading to storage and accuracy limitations. To address this, n is often restricted to 2 or 3, which simplifies computations but reduces model accuracy.

1.3.4 Neural Language Models

NLMs represent a significant advancement by utilizing neural networks to predict word probabilities, allowing for better handling of longer sequences. Central to NLMs is the concept of *word vectors*, numerical representations capturing semantic relationships between words (e.g., “*cat*” is closer to “*dog*” than “*tree*”). Techniques like *Word2Vec* generate dense word embeddings, enabling machines to process language more effectively. NLMs feature an architecture with input, hidden, and output layers, where the hidden layer applies non-linear transformations (e.g., *sigmoid*, *tanh*) to predict the next word. Outputs are normalized using the *Softmax function* to produce probabilities.

1.3.5 Pre-trained Language Models

PLMs introduce a two-phase training process:

1. **Pre-training:** The model learns general linguistic patterns from massive unlabeled datasets, including syntax, semantics, and logical structures.
2. **Fine-tuning:** The model is adapted to specific tasks such as text summarization or question answering using smaller, labeled datasets.

This "*pre-training and fine-tuning*" paradigm allows PLMs to achieve exceptional performance across diverse NLP applications. Notable examples include *BERT* and *GPT-2*, which have set new benchmarks in tasks like machine translation and sentiment analysis.

1.3.6 Large Language Models

LLMs are advanced AI systems designed to process and understand human language. They are referred to as "large" due to their extensive architecture, often consisting of hundreds of millions or even billions of parameters that determine their behavior. These models are pre-trained on vast corpora of text data. LLMs represent a significant research direction in the field of NLP and have garnered increasing attention from researchers. They are being applied across various domains, including medicine, coding, and mathematics, to address domain-specific challenges. Additionally, LLMs are effective in tasks like instruction-following, enabling them to assist in solving complex, specialized problems.

LLMs, such as *GPT-3*, *GPT-4*, and *PaLM*, mark the culmination of LM evolution, with billions or even trillions of parameters trained on vast text corpora. They build upon PLMs by emphasizing scalability and alignment with human values. Unlike earlier models, LLMs integrate pre-training on extensive corpora with fine-tuning to adapt to user commands and ethical considerations. Their exceptional size and computational power enable capabilities such as contextual reasoning and task adaptability. For instance, *GPT-3*, with 175 billion parameters, surpasses *GPT-2*'s 1.5 billion parameters in both scale and performance. This dramatic expansion in model size and dataset volume allows LLMs to excel in general-purpose and specialized applications [Wang et al., 2024].

1.3.7 Comparison of Language Models

LMs have evolved significantly over time, from simple probabilistic methods to advanced deep learning-based architectures. The table below provides a comparative overview of different types of LMs, highlighting their key features, limitations, and examples.

Table 1.2 provides a structured comparison of different types of LMs, ranging from early probabilistic models to modern deep learning-based LLMs. It highlights the key advancements in NLP, showcasing how each generation of models has improved in terms of capabilities, computational complexity, and real-world applications.

Table 1.2: Comparison of Different Types of Language Models

Model Type	Era	Key Features	Limitations	Examples
SLMs	1990s	<ul style="list-style-type: none">- Based on probabilistic n-gram models- Computes word probability using previous $n - 1$ words	<ul style="list-style-type: none">- Struggles with long-range dependencies- Exponential growth of word combinations- Limited accuracy	N-gram models, Hidden Markov Models (HMM)
NLMs	2010s	<ul style="list-style-type: none">- Uses neural networks for word prediction- Employs word embeddings (e.g., Word2Vec)- Can handle longer sequences than SLMs	<ul style="list-style-type: none">- Computationally expensive- Requires large amounts of training data	Word2Vec, GloVe, FastText
PLMs	Late 2010s	<ul style="list-style-type: none">- Two-phase training: pre-training and fine-tuning- Learns general linguistic patterns from large corpora- Transfer learning allows adaptation to specific tasks	<ul style="list-style-type: none">- Requires significant computational resources- Can inherit biases from training data	BERT, GPT-2, RoBERTa
SLMs	2020s	<ul style="list-style-type: none">- Billions of parameters for improved performance- Trained on massive datasets- Fine-tuned for instruction-following and ethical alignment- Enables contextual reasoning and adaptability	<ul style="list-style-type: none">- Extremely high computational and storage requirements- Risk of hallucination and biases- Expensive to train and deploy	GPT-3, GPT-4, PaLM, LLaMA

1.4 Related Work of Chatbot Applications in Domain-Specific Contexts

The most recent advances in conversational AI have led in the creation of domain-specific chatbots that merge classic rule-based systems with LLMs to RAG systems. This section provides a comparative analysis of selected studies demonstrating the use of chatbots in sectors such restaurant services, academic libraries, healthcare, and the legal sector.

1.4.0.1 Study A: Retrieval-Augmented Generation Based Restaurant Chatbot

A team from San Jose State University conducted a study last July on a restaurant chatbot using RAG [Bhat et al., 2024]. They combined the T5 LM with a Neo4j knowledge graph to improve chatbot responses. The system uses question-answer clusters stored in Neo4j, linked through TF-IDF embeddings, to retrieve accurate answer tokens. This helps improve both contextual understanding and response accuracy. This approach improved the chatbot’s ability to understand context and generate precise replies. The results were impressive:

- Intent recognition: 92.55
- Entity extraction: 84.55
- Response quality: A BLEU score of 0.60 shows strong language understanding.

1.4.0.2 Study B: Conception of a Library Chatbot

In this article [Malick and Marone, 2024], the goal was to develop an AI chatbot to help university libraries, especially in Senegal, improve their information services. They used Botpress (a powerful all-in-one platform for creating AI agents, powered by the latest models) as their NLP-driven conversational AI platform. The chatbot utilizes Natural Language Understanding (NLU) and GPT models developed by OpenAI. The chatbot was trained using frequently asked questions and user inquiries.

For evaluation, they relied on scenario-based testing rather than predefined quantitative metrics. The creators achieved their goals by improving the service, providing 24/7 availability for university libraries, and reducing the workload of librarians by handling simple inquiries.

1.4.0.3 Study C: Medical and Healthcare Chatbots

In two different research papers, they used different technologies.

The initial study employed a rule-based chatbot, working like ELIZA with a keyword-based approach. They created a database using MySQL, and for evaluation, provided the chatbot to 100 individuals for testing. The outcome was a 91 [Banu Prakash and Saravanan, 2023]

The second article, Med-Bot AI-powered [Bhatt and Vaghela, 2023], focused on a medical chatbot based on RAG using the LLaMA-2 model. It processed medical PDFs using PyPDFDirectoryLoader, and used RecursiveCharacterTextSplitter to extract and chunk the PDF text. Then, they embedded the chunks and stored them using ChromaDB for retrieval. They connected the embeddings to the LLM using LangChain, and optimized the setup with AutoGPTQ to reduce memory usage while keeping good accuracy.

In both articles, there were no specific metrics or benchmarks used for evaluation they mostly focused on efficient retrieval .

1.4.1 Study D: Legal Chatbot in India

In a recent publication from the end of February [Panchal et al., 2024], researchers developed **LawPal**, a comprehensive legal RAG system for India, utilizing the **DeepSeek-R1:5B** model. A knowledge base was constructed using the *Indian Constitution* and various legal research papers, with real-time updates integrated through web scraping.

The data was segmented into chunks of 500–700 characters with overlapping segments. These were embedded into 1,024-dimensional vectors using DeepSeek-R1:5B. The embeddings were stored in **FAISS**, optimized through hierarchical indexing and **Product Quantization (PQ)** combined with **Inverted File Indexing (IVF)** to enhance retrieval speed and accuracy.

Evaluation

- **Retrieval Performance (FAISS):**
 - Cosine similarity used for semantic search.
 - Recall: approximately 90%.
 - Query speed: 10–50 ms per query (total retrieval time: 1–2 minutes).
- **Generation Performance (DeepSeek-R1:5B):**
 - Metrics used: BLEU, ROUGE, Legal Consistency Score (LCS) – exact scores not provided.

- Human evaluation showed over 90% accuracy.
- Full query resolution time: 2–4 minutes.

The system leveraged **LangChain** to connect the LLM to the knowledge base and applied *prompt engineering* techniques to ensure legally grounded and accurate responses.

1.4.2 Comparative Analysis

Table 2.3 delineates the key elements of each study to improve comprehension of the methodology and efficacy of domain-specific chatbot systems. 1.3 summarizes the key elements of each study.

Table 1.3: Comparison of Domain-Specific Chatbot Studies

Study	Domain	Technology / Model	Evaluation Method	Main Results
Study A: RAG Restaurant Chatbot	Restaurant	T5 + Neo4j Knowledge Graph + TF-IDF	Intent/entity accuracy, BLEU score	92.55% intent accuracy, BLEU: 0.60
Study B: Library Chatbot	Academic Libraries (Senegal)	Botpress + NLU + OpenAI GPT	Scenario testing (no metrics)	24/7 availability, librarian workload reduced
Study C1: ELIZA-style Medical Bot	Healthcare	Rule-based + Keyword matching + MySQL	User testing (100 users)	91% recommendation success
Study C2: Med-Bot AI-powered	Healthcare	LLaMA-2 + ChromaDB + LangChain + AutoGPTQ	No formal metrics, focus on retrieval	Efficient PDF retrieval pipeline
Study D: LawPal (Legal RAG)	Legal (India)	DeepSeek-R1:5B + FAISS + LangChain + Web scraping	Recall, BLEU, ROUGE, LCS + human review	Recall 90%, 90%+ human evaluation accuracy

1.4.3 Summary

Each study demonstrates that chatbot design is strongly influenced by the specific needs of its domain. In the restaurant sector, structured knowledge graphs and precise intent recognition are key to delivering effective customer interactions. For academic libraries, the focus lies on availability and ease of use, which justified the use of scenario-based evaluation. In the healthcare domain, there is a clear contrast some systems rely on

rule-based approaches for simplicity, while others adopt advanced RAG architectures to enhance information retrieval. Meanwhile, in the legal field, accuracy and traceability are essential, leading to the use of optimized vector search, human evaluation, and legally grounded prompts. While all systems aim to improve user experience, their design choices and evaluation methods reflect the unique priorities of their respective domains.

1.5 Conclusion

This chapter explored the evolution of chatbot technology and language models, focusing their applications in various sectors. Since each model type has particular benefits, RAG is the most suitable approach for our research because of its capacity to integrate precision with adaptability. RAG designs have demonstrated efficacy in sectors including as health-care, hospitality, and legal services, rendering them suitable for our investigation into the development of efficient, domain-specific chatbots. Following efforts will concentrate on modilisation our legal chatbot project.

Chapter 2

Design of the RAG-based Legal Chatbot

2.1 Introduction

In this chapter, we describe how we built the core of our RAG system for the legal chatbot. We explain how documents are split into chunks, converted into vectors, stored in vector databases, and retrieved based on user queries. We also present the tools, storage options, retrieval strategies, and language models used to generate relevant and accurate responses in French.

2.2 System Overview

The developed system adopts a rag architecture, as illustrated in Figure [2.1](#), combining information retrieval techniques with the generative capabilities of a language model to deliver accurate and contextually relevant answers to user queries.

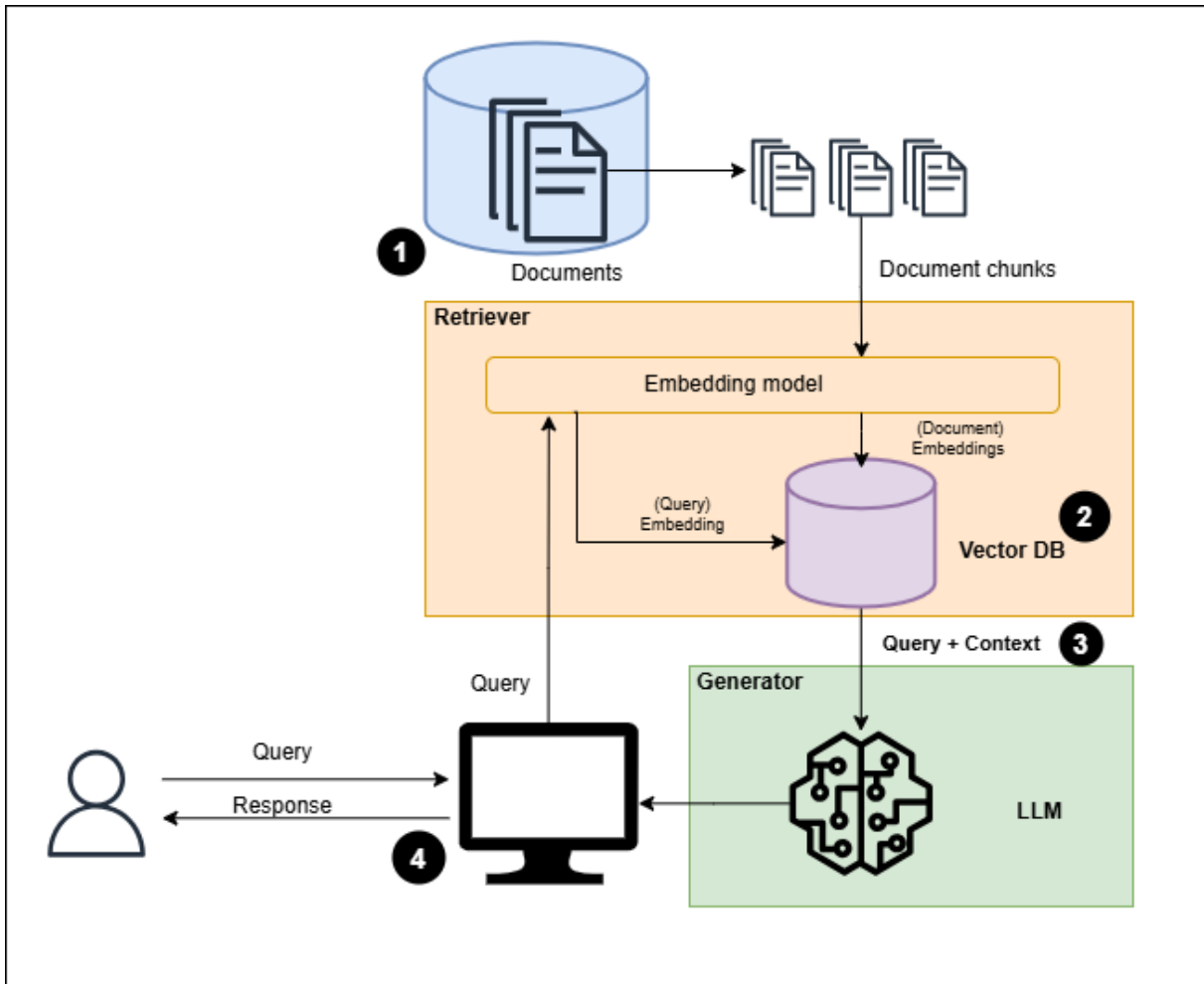


Figure 2.1: Global system pipeline for RAG

The system comprises four main stages, each corresponding to a module illustrated in the figure:

- **Data Collection and Preparation**

This stage involves gathering various legal documents, such as Q&A sheets, policy texts, and regulatory guidelines. These documents are then split into smaller, manageable chunks to enable more effective retrieval and embedding. Chunking ensures that only the most relevant portions are retrieved, rather than entire documents, which improves accuracy.

- **Vector Database Creation**

Once prepared, the chunks are converted into high-dimensional vector representations using an embedding model. These embeddings are stored in a **vector database**, which supports fast and efficient semantic search. This acts as the system's memory for retrieving relevant information.

- **Contextual Retrieval**

When a user submits a query, it is also embedded into a vector. The system compares this query vector to the document vectors in the database and retrieves the most relevant chunks, which serve as the context for the final answer.

- **Answer Generation**

The retrieved context and user query are sent together to a **large language model (LLM)**. The model then generates a clear and accurate response based on the provided context, which is returned to the user.

2.3 Data Collection and Preparation

The dataset used in this study was collected from various sources, including Djezzy's internal files and official Algerian journals. The data is primarily in French and consists of a combination of text, tables, graphs, and images, with text as the principal component.

This section offers a comprehensive analysis of the data sources, their attributes, and the preprocessing procedures necessary for data integration into a chatbot system.

2.3.1 Data Sources

Figure 2.2 illustrates the organization and structure of the data. It highlights the links between legal documents, domain classifications, and question-answer datasets.

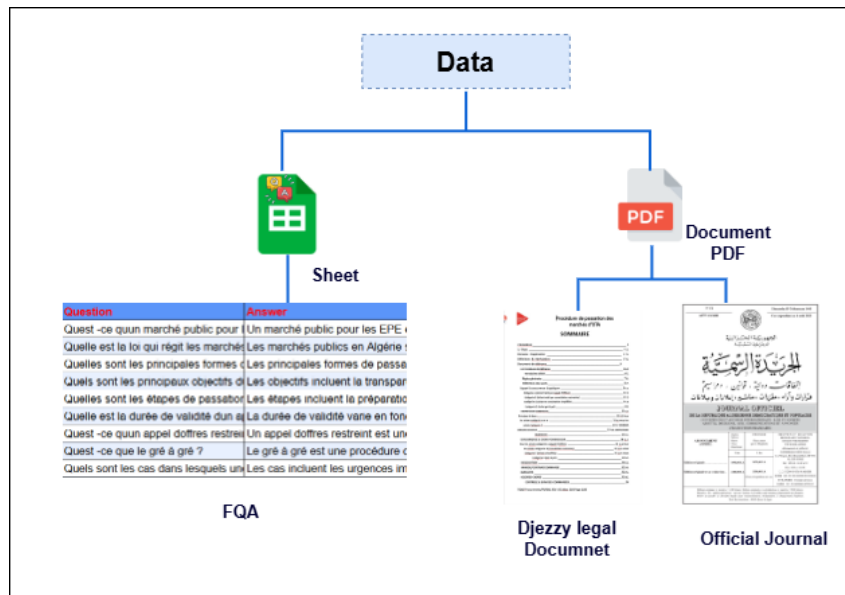


Figure 2.2: Structure of the dataset

The dataset is divided into two main categories:

2.3.1.1 Question-Answer Sheet

This set comprises a structured document offering approximately 150 frequently asked legal questions along with their respective answers (Q&A Sheet). It constitutes the essential knowledge repository for the chatbot to respond to critical legal inquiries relevant to the company.

2.3.1.2 Legal Documents

The dataset also includes various legal documents (in PDF format), categorized as follows:

- **Algerian Official Journals:** Contain applicable laws and regulations.
- **Djezzy's Internal Legal Documents:** Created by the legal department to simplify internal procedures. These are critical for tailoring the chatbot's responses to the company's specific needs.

Format and Characteristics:

To better understand the composition of our dataset, Table 2.1 summarizes the format, language, and content characteristics of the legal documents used in our system.

Table 2.1: Format and characteristics of the document dataset

Category	Details
Number of Documents	17 PDFs
Length	7 to 229 pages
Language	French
Content Types	Text (majority), tables, graphs, and images
Document Types	Official laws, internal legal guidelines, scanned and digital PDFs
Processing Required	OCR for scanned documents, text extraction, and structuring

2.3.2 Challenges and Data Preprocessing

Several preprocessing steps were required to make the data applicable to the chatbot. The following sections outline the issues encountered and the techniques applied.

2.3.2.1 Issues Faced During Document Extraction

The initial attempts at extracting text from legal PDFs revealed several limitations:

- **Multilingual Content:** Some official texts included Arabic passages that were difficult to process using standard tools.
- **Scanned Documents:** A portion of the dataset consisted of scanned documents with low legibility, requiring Optical Character Recognition (OCR).
- **Complex Layouts:** The presence of tables, images, multi-column formats, and various heading levels made sequential reading and parsing more difficult.

2.3.2.2 Comparison of Extraction Strategies

To address these issues, two different extraction strategies were tested and evaluated on a representative sample of documents:

Criterion	Strategy 1	Strategy 2
Arabic Language Support	Not supported	Partially handled
Table Parsing Accuracy	Poor	Good
Column Layout Handling	Incorrect reading order	Sequential order preserved
Formatting Preservation	Structure lost	Markdown structure retained
OCR for Scanned Files	Not integrated	Requires manual completion

Table 2.2: Comparison of document extraction strategies

Based on this comparison, a more advanced strategy using PyMuPDF4LLM was adopted. This later was able to convert documents into a structured Markdown format, preserving headers, formatting, and layout, and significantly simplifying the annotation and enrichment process.

2.3.2.3 Adoption of PyMuPDF4LLM

To overcome these limitations, we adopted PyMuPDF4LLM, a robust tool that converts PDF pages into Markdown format with high precision. It effectively handles text, tables, and images, streamlining the preprocessing workflow without requiring additional treatment .

How PyMuPDF4LLM Works:

- **Text and Table Extraction:** Detects standard text, tables, and multi-column layouts, ensuring the correct reading sequence.
- **Formatting:** Identifies headers, bold or italic text, lists, and code blocks, marking them appropriately.
- **Image Extraction:** Extracts images and vector graphics, storing them as separate files while embedding textual content in Markdown.

Figure 2.3 illustrates how PyMuPDF4LLM processes a PDF document.

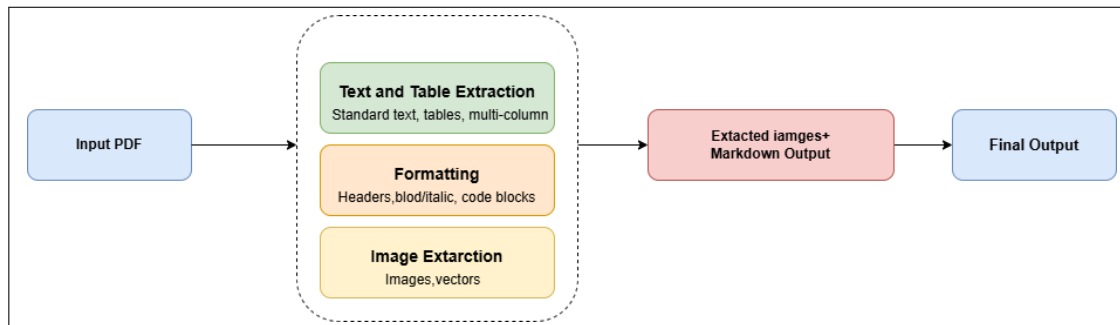


Figure 2.3: PDF transformation process with PyMuPDF4LLM

2.3.3 Dealing with Graphs

For graphs and other visual content that PyMuPDF4LLM could not fully process, we collaborated with legal experts to manually extract the data. We also used GPT-4 to help convert graphs into text.

Figure 2.4 illustrates examples of graphs from the documents.

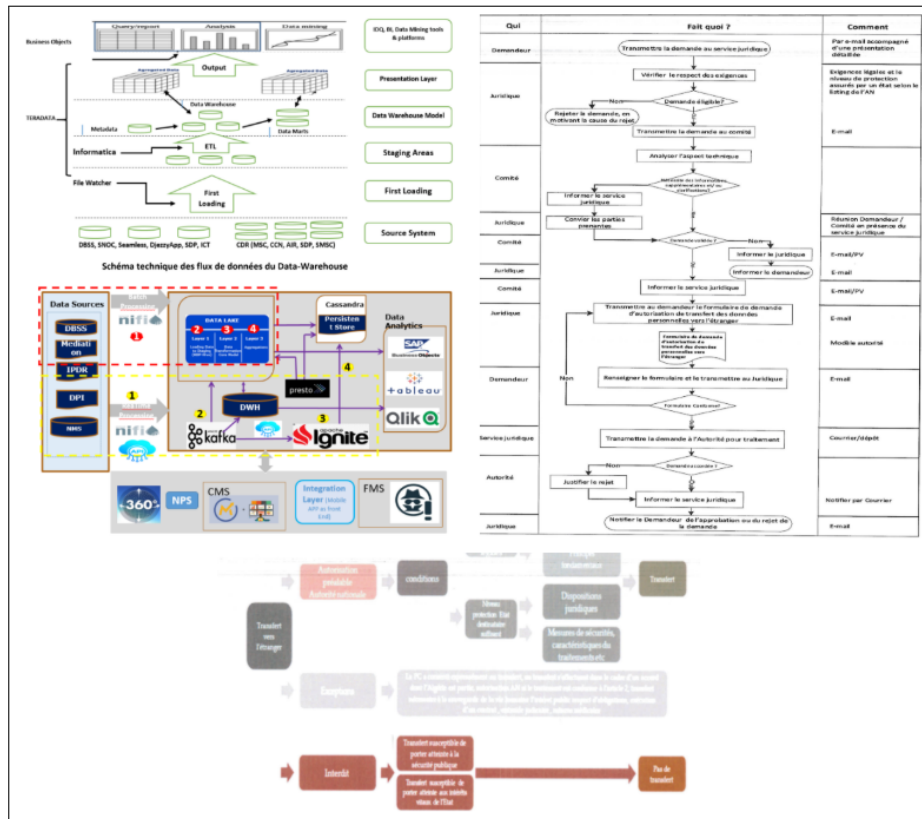


Figure 2.4: Examples of graphs found in the documents

2.3.4 Data preparation and organization

To efficiently manage and store the preprocessed data, we designed a relational database to organize the documents and their metadata. The database structure consists of three main tables: **Domain**, **Document**, and **FAQ**.

2.3.4.1 Database Structure

To ensure clarity, consistency, and efficient access, the data was structured into normalized relational tables, each playing a specific role in organizing legal content and its metadata.

Domain Table: Represents different legal domains or categories (e.g., labor law, contract law) the structure of this table is detailed in Table 2.3 .

Table 2.3: Structure of the Domain table

columns	Type	Description
id	Primary Key	Unique identifier for each domain
name	Text	Name of the legal domain
description	Text	Short description of the domain

Document Table: Stores details of each legal document associated with a domain, the details are presented in table 2.3

Table 2.4: Structure of the Document table

columns	Type	Description
id	Primary Key	Unique identifier for each document
name	Text	Title or name of the legal document
pdf	File/Text	File path or the actual uploaded PDF file
txtofpdf	Long Text	Full unstructured text extracted from the PDF
jsonofpdf	JSON	Structured content of the PDF in JSON format
resume	Text	Summary or abstract of the document
date_insertion	DateTime	Date and time the document was inserted into the database
domain_id	Foreign Key	References id in the Domain table

Question_Answer Table: The structure and attributes of the Question_Answer table are outlined in Table 2.5, its objective is to store legal questions and their corresponding answers, each associated with a specific domain.

Table 2.5: Structure of the Question_Answer table

Field	Type	Description
id	Primary Key	Unique identifier for each question-answer pair
question_text	Text	The legal question posed
answer_text	Text	The corresponding legal answer
domain_id	Foreign Key	References id in the Domain table

Figure 2.5 shows the relational database schema and the links between the different tables.

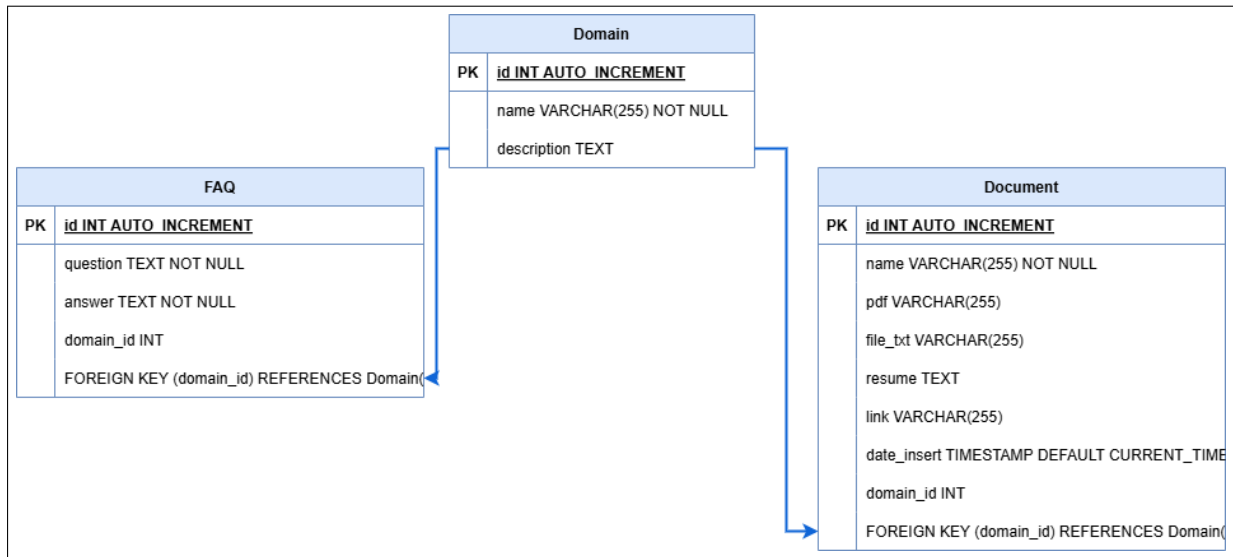


Figure 2.5: Relational database schema for organizing legal documents and FAQs

- The **Domain** table is the core entity that connects documents and question-answer pairs.
- Each **Document** belongs to a domain via the `domain_id` foreign key, forming a one-to-many relationship.
- Each **Question_Answer** is linked to a domain in a similar one-to-many fashion. **Question_Answer**; they relate independently through their shared domain.

2.4 Vector Storage and Indexing

To ensure the effective retrieval of relevant legal information. The system depends on two essentials, which are **document chunking** and **vector storage with indexing** to properly retrieve relevant legal information. These procedures ensure that data or information is not only saved in a number format but can also be retrieved quickly and accurately when queried.

Figure 2.6 illustrates this vector store pipeline, showing the transformation of document chunks into text chunks, the embedding process, and the final storage in the vector database.

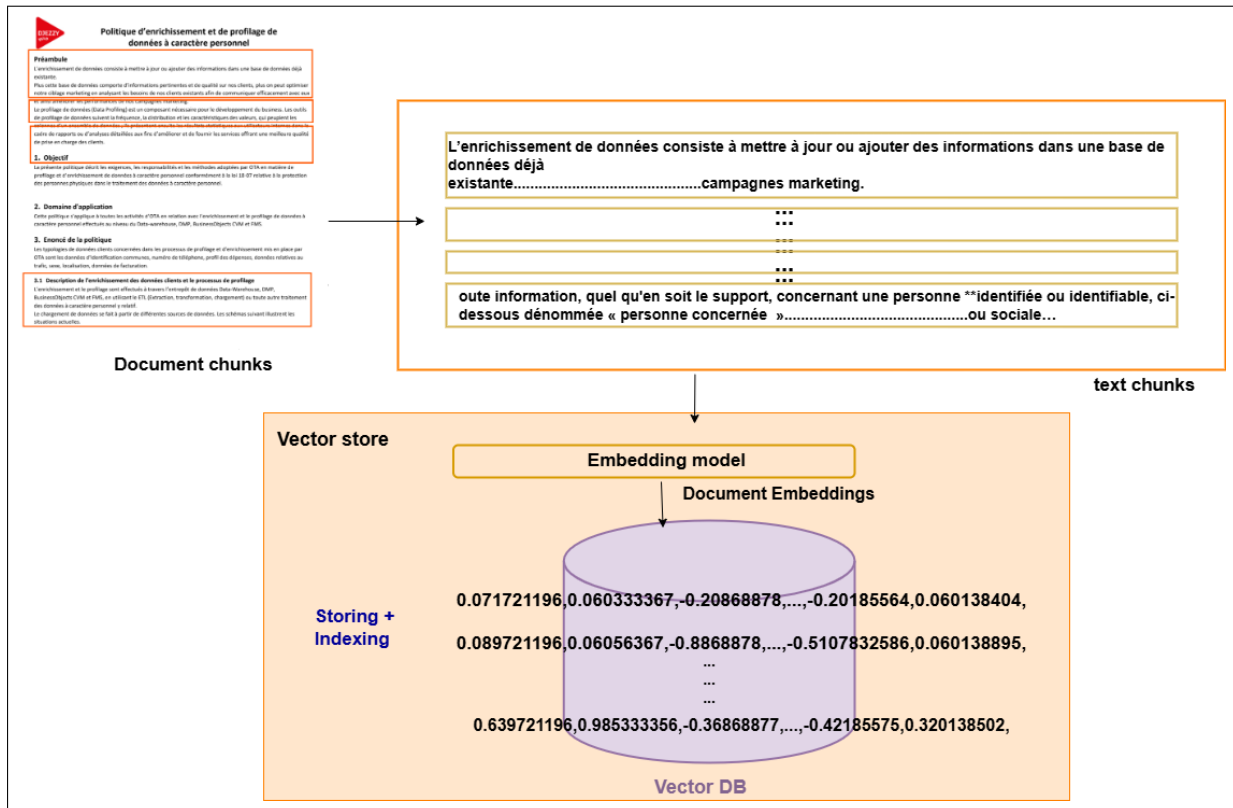


Figure 2.6: visualization of vector store pipeline

2.4.1 Document Chunking

Before generating embeddings, documents must be split into smaller units, known as **chunks**. Chunking improves retrieval precision by enabling the system to match only the most relevant parts of a document.

Various chunking techniques exist, depending on the target use case and data structure [Kshirsagar, 2024].

A comparative summary of these methods including their descriptions, ideal use cases, and standard tools is provided in Table 2.6.

Chunking Strategy	Description	Use Case / Advantage	Example Tools / Methods
Fixed-Size Chunking	Divides text into equal-sized segments based on character or word count, with optional overlap.	Useful when chunk size is more important than context; suitable for large texts.	<code>CharacterTextSplitter</code> (LangChain)
Recursive Chunking	Splits text recursively using separators like <code> </code> , <code>periods</code> , <code>etc.</code> , to preserve structure.	Effective for semi-structured text with logical breaks.	<code>RecursiveCharacterTextSplitter</code> (LangChain)
Semantic Chunking	Uses meaning or context to determine split points via embeddings or language models.	Ideal for summarization, search, and QA tasks.	NLP models, embedding techniques
Overlap-Based Chunking	Adds overlapping content between chunks to retain context.	Helps maintain continuity in chat-based or sequential tasks.	Configurable in most chunking tools

Table 2.6: Overview of Chunking Techniques for RAG

As legal texts frequently exhibit natural hierarchical structures such as sections, sub-sections, and articles, we opted for **recursive chunking** in our preprocessing of legal documents, but we implemented our own **custom chunking logic** adaptable to our extraction process and the specificities of legal text that we have. This approach allowed us to preserve the full hierarchy of the text, including main section titles, subsections, and sub-subsections, ensuring that the contextual relationships within legal documents. Preserving this structure enables the system to maintain the semantic integrity of each segment, which is especially beneficial in retrieval-based applications.

An example of our chunking logic is shown in Algorithm 1, which outlines the custom procedure we implemented.

Algorithm 1 Recursive Chunking for Legal Documents

```
1: function EXTRACT_SECTIONS(text)
2:   sections  $\leftarrow$  {}
3:   current_title  $\leftarrow$  None
4:   current_content  $\leftarrow$  []
5:   for all line in text.splitlines() do
6:     line  $\leftarrow$  line.strip()
7:     if line.startswith(" ") then
8:       SAVE_PREVIOUS_SECTION
9:       current_title  $\leftarrow$  line[4:]
10:      current_content  $\leftarrow$  []
11:    else if line.startswith(" ") then
12:      SAVE_PREVIOUS_SECTION
13:      current_title  $\leftarrow$  current_title + " > " + line[5:]
14:      current_content  $\leftarrow$  [] line  $\neq$  "" and not line.startswith(" ")
15:      Append line to current_content
16:    end if
17:  end for
18:  SAVE_PREVIOUS_SECTION
19:  return sections
20: function SAVE_PREVIOUS_SECTION
21:   if current_title and current_content  $\neq$  [] then
22:     sections[current_title]  $\leftarrow$  "\n".join(current_content)
23:   end if
24: end function
25: end function
```

Explanation:

The function `extract_sections()` processes a legal text with Markdown-style headings. It distinguishes major sections (lines starting with " ") and nested subsections (lines with " "). The content under each section is gathered until a new title appears. The resulting structure is a dictionary where the keys represent section titles (e.g., `Introduction` > `Subsection`) and the values are the corresponding grouped text blocks.

An example of our chunking logic is shown in Figure 2.7, which preserves the hierarchical structure and produces semantically clear chunks for better information retrieval.

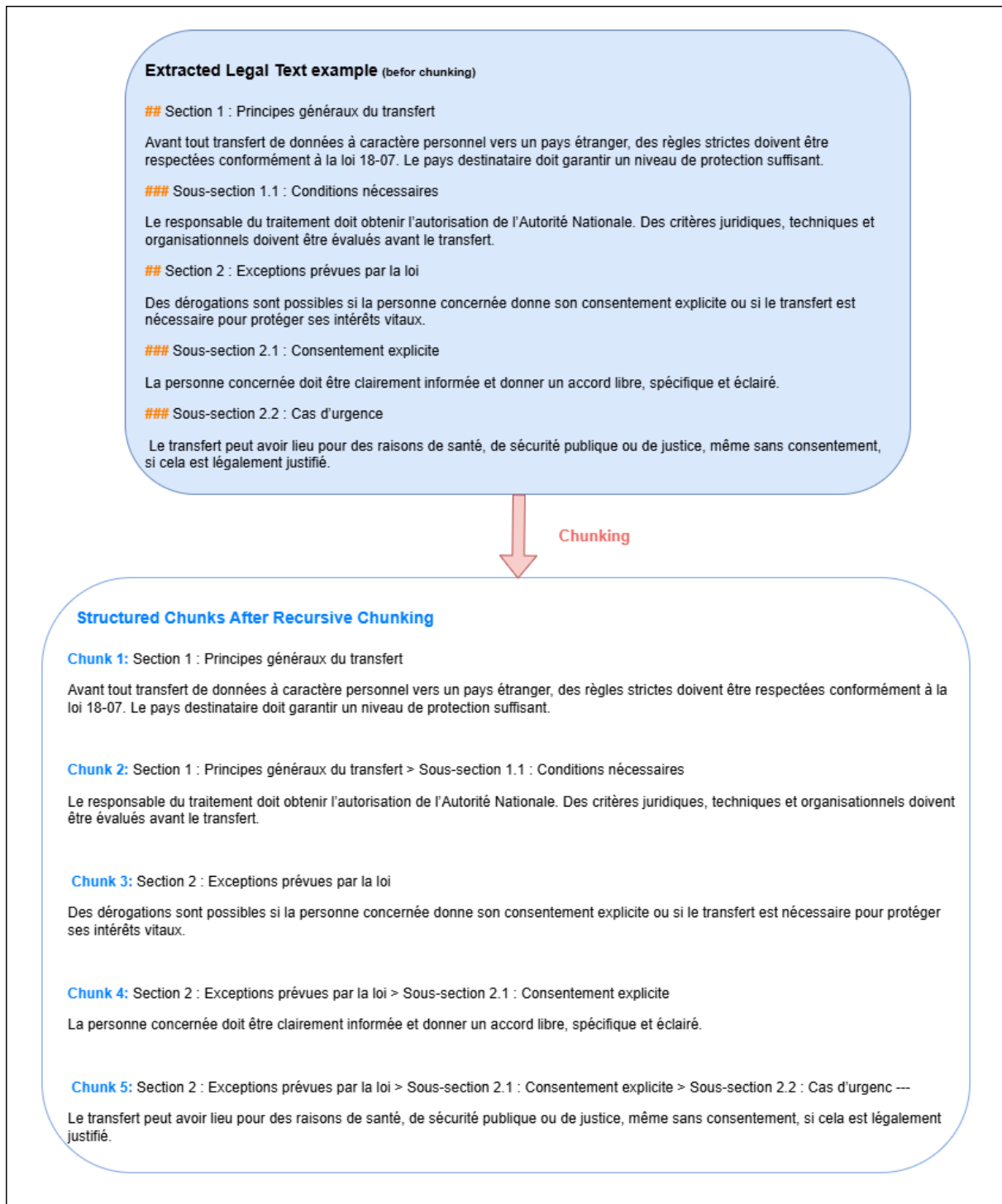


Figure 2.7: Example result of chunking.

For the QA dataset, we chose to create a chunk for each question along with its corresponding answer so that each pair forms a single coherent chunk.

2.4.2 Embedding Generation

After **chunking**, each text segment is transformed into a numerical representation known as a **text embedding**. A **text embedding** is a vector of real numbers assigned by an **embedding model** that encapsulates the **semantic meaning** of a sentence.

When an embedding model produces a vector representation of a sentence, it also situates the vector inside a multidimensional space according to its assigned values. The dimensional space's size differs for each model, resulting in corresponding variations in the vector values. But still, all models arrange the vectors so that sentences with similar meanings are in less distance to each other [IBM, 2024].

Most embedding models produce vectors with multiple dimensions, ranging from hundreds to thousands, making visualization impossible. Should an embedding model produce a three-dimensional vector, it may appear as follows 2.8.

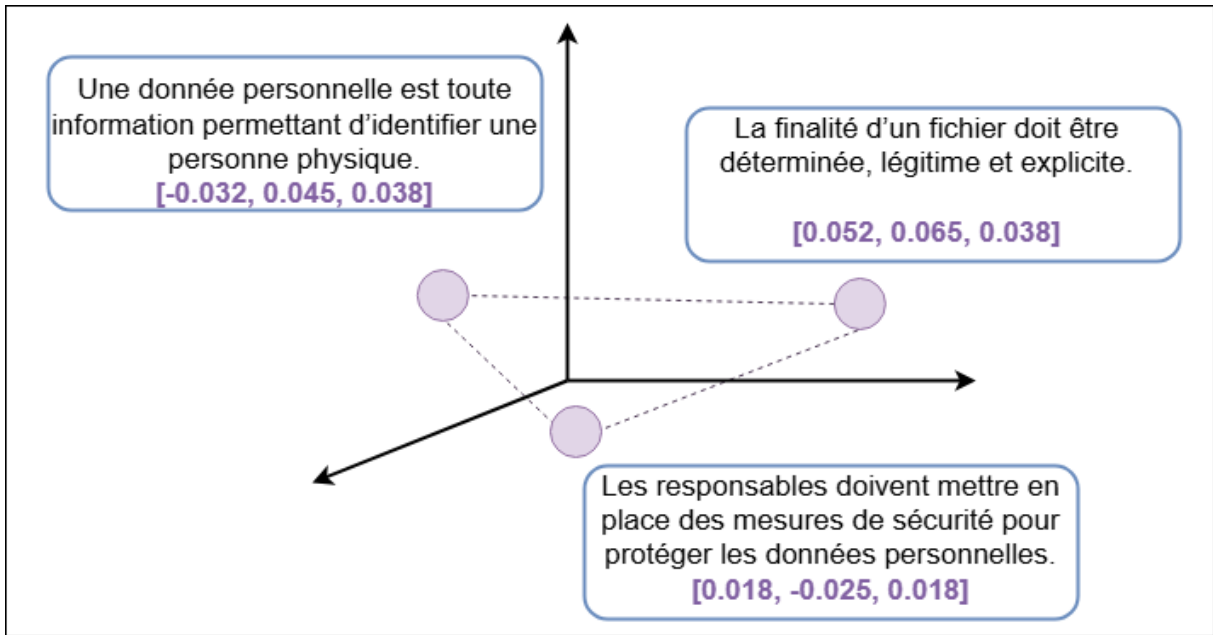


Figure 2.8: Example of 3D Text Embeddings

The vector values depicted in the figure 2.8 are fictitious and are provided solely to elucidate this hypothetical scenario.

In our system, we selected the **CamemBERT-based model** from Hugging Face. This model is particularly well-suited for French language tasks, as it is trained on French textual data and fine-tuned for generating sentence embeddings. The decision was motivated by the fact that our documents are written in French, and using a multilingual or English-centric model may result in weaker vector representations.

Table 2.7 presents the main characteristics of the embedding model used in our pipeline

[[Hugging Face, 2021](#)]:

Table 2.7: Embedding model specifications

Property	Description
Model name	dangvantuan/sentence-camembert-base
Model size	~110M parameters (based on CamemBERT-base architecture)
Embedding dimensionality	768
Purpose	Sentence-level semantic embedding optimized for French
Price	Open-source and free to use under the MIT license

2.4.3 Vector Storage

Once document chunks are embedded into vectors, these high-dimensional representations must be stored in a specialized data structure called a *vector store*, or vector database that is optimized for the efficient storage, indexing, and similarity search of dense vector representations. Its principal function within RAG systems is to enable fast and accurate nearest neighbor searches, which retrieve semantically similar document segments during inference.

Categories of Vector Storage Systems: Vector databases vary in design, scalability, and supported features. Based on current research and industry practice [[Johnson et al., 2019a](#), [pgvector Contributors, 2024](#), [Qdrant Team, 2024](#)], they can be classified into the following categories:

- **ANN Libraries:** These provide efficient in-memory ANN search but typically lack features like persistence, CRUD operations, or structured query support. Example: FAISS¹.
- **Database Extensions:** Extensions like `pgvector` integrate vector operations into existing relational databases (e.g., PostgreSQL), enabling hybrid queries combining structured filters and semantic similarity.
- **Dedicated Vector Databases:** Platforms such as Qdrant², Milvus³, and Weaviate⁴ are purpose-built for semantic search and typically include distributed indexing, filtering, and cloud-native capabilities.

¹<https://ai.meta.com/tools/faiss/>

²<https://qdrant.tech/>

³<https://milvus.io/>

⁴<https://weaviate.io/>

- **Lightweight Embedded Databases:** Solutions like ChromaDB⁵ are designed for rapid prototyping and are often embedded within development frameworks like LangChain⁶.

⁵<https://docs.trychroma.com/>

⁶<https://www.langchain.com/>

Vector Store	Category	Open Source	CRUD	Hybrid Search	ANN Perf.	Indexing Algorithm	Complexity
FAISS [Johnson et al., 2019a]	ANN Library	Yes	No	No	High	Inverted File Flat Index (IVF), HNSW, PQ	Medium
pgvector [pgvector Contributors, 2024]	PostgreSQL Extension	Yes	Yes	Yes	Moderate	IVF, HNSW	Low
Chroma DB [ChromaDB Developers, 2024]	Embedded Store	Yes	Yes	Limited	Low–Moderate	Basic cosine indexing	Low
Qdrant [Qdrant Team, 2024]	Dedicated Vector DB	Yes	Yes	Yes	High	HNSW	Medium
Weaviate [Weaviate Team, 2024]	Semantic Engine	Yes	Yes	Yes	High	HNSW	Medium
Pinecone [Pinecone Systems Inc., 2024]	Managed Vector DB	No	Yes	Yes	High	Proprietary (likely HNSW)	Low
Milvus [Zilliz, 2024]	Distributed DB	Yes	Yes	Yes	High	IVF, HNSW, ANNOY	Medium–High

Table 2.8: Comparison of representative vector storage systems for RAG-based applications.

Based on the comparison in Table 2.8, we conclude that the best option for our case is **pgvector**, which provides a sample interface with PostgreSQL. This decision was af-

ected by various practical and technological factors. `pgvector` is a native extension of PostgreSQL, it obviates the necessity of managing a separate vector database, in turn significantly reducing deployment complexity and system maintenance overhead. In the end, and most importantly, `pgvector` facilitates comprehensive CRUD (Create, Read, Update, Delete) operations, simplifying the management, modification, or elimination of embeddings as our legal corpus evolves.

Additionally, while `pgvector` type can be extended with indexing algorithms (e.g., HNSW) to enable faster and more scalable semantic retrieval, these attributes collectively rendered `pgvector` a practical and effective option for our retrieval-augmented legal assistant.

The Figure 2.9 presents the final database schema implemented using PostgreSQL, enhanced with dedicated vector tables for document chunks and question-answer pairs. The system integrates structured metadata storage and vector embeddings within a unified architecture.

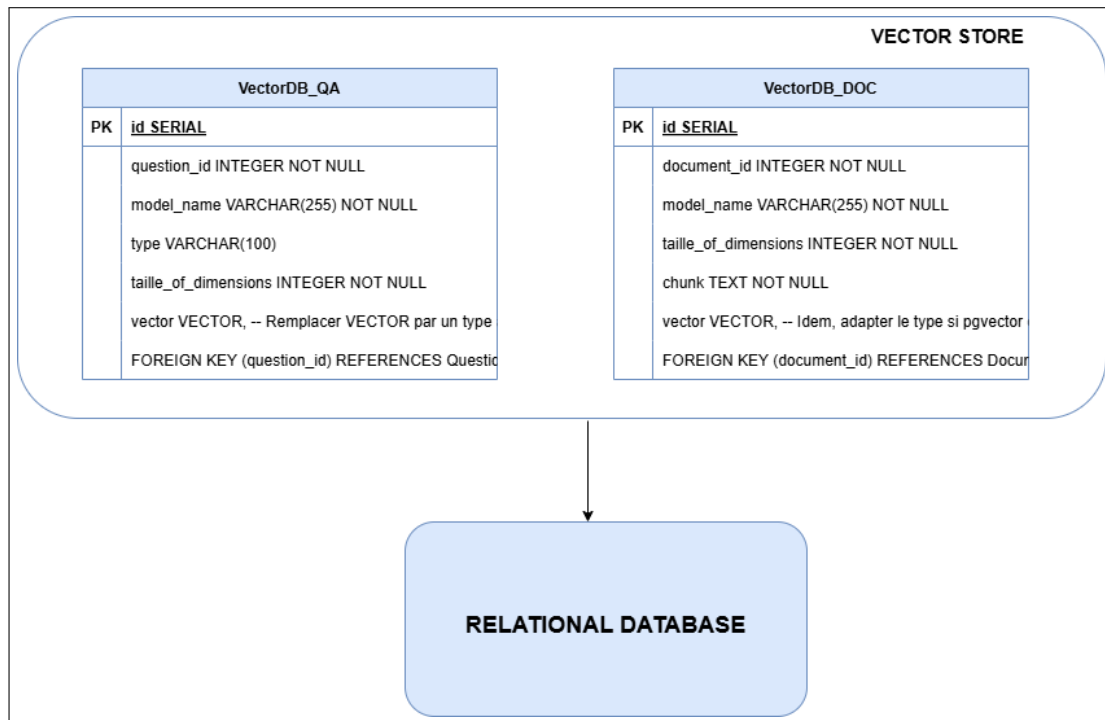


Figure 2.9: Database and vector storage schema in PostgreSQL and `pgvector` extension

VectorDB_QA Table: This table stores vector embeddings representations of legal questions for retrieval and similarity matching.

Table 2.9: Schema of VectorDB_QA Table

Field	Type	Description
id (PK)	Integer	Unique identifier for each vector record.
question_id (FK)	Integer	References the corresponding question in the Question_Answer table.
model_name	Text	Name of the model used to generate the vector.
type	Text	Type of embedding (e.g., semantic, contextual).
taille_of_dimensions	Integer	Dimensionality of the embedding vector.
vector	Vector	The embedding vector storing the representation.

VectorDB_DOC Table: This table 2.10 contains vector embeddings of segmented legal documents (chunks), allowing the system to retrieve relevant passages during the response generation proces.

Table 2.10: Schema of VectorDB_DOC Table

Field	Type	Description
id (PK)	Integer	Unique identifier for each vector record.
document_id (FK)	Integer	References the corresponding document.
model_name	Text	Name of the model used for generating the vector.
taille_of_dimensions	Integer	Dimensionality of the embedding vector.
chunk	Text	The text chunk extracted from the document.
vector	Vector	The embedding vector storing the representation.

each table is linked with his table text in the relational database.

- Each **VectorDB_QA** entry is linked to a **Question_Answer**.
- Each **VectorDB_DOC** entry is linked to a **Document**.
- There is no direct link between **Document** and

2.4.4 Indexing Algorithms

To enable fast vector similarity searches, vector stores use indexing algorithms. These techniques reduce the number of computations required during query processing.

HNSW:

HNSW builds a graph of vectors, where each node is connected to its nearest neighbors. It enables approximate nearest neighbor search with logarithmic complexity, used in real-time systems needing quick response [Malkov and Yashunin, 2020]. Figure 2.10 illustrates the layered and navigable structure of the HNSW graph.

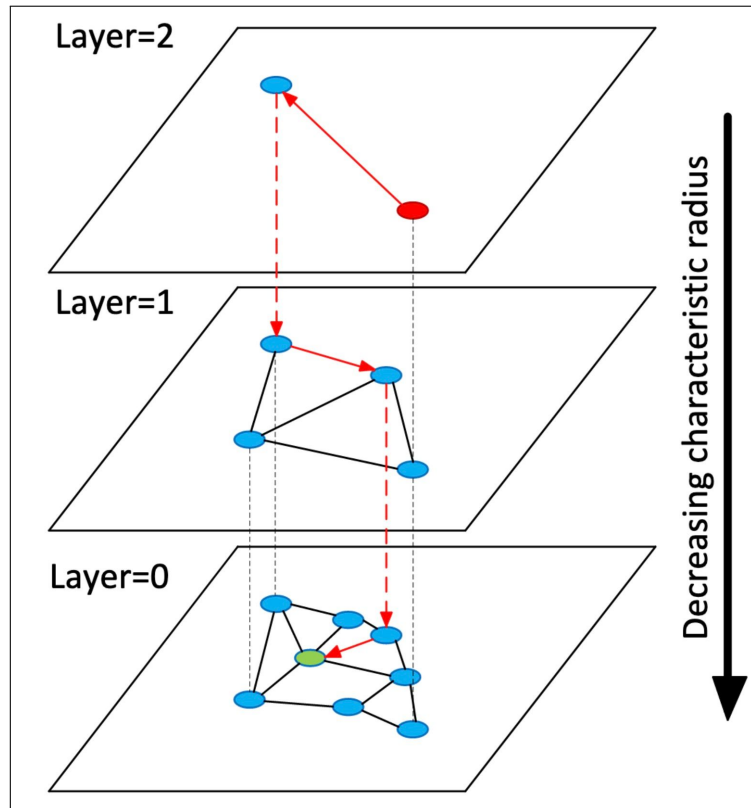


Figure 2.10: Visualization of the HNSW graph-based structure [Zilliz, 2024].

IVF

IVF partitions vectors into clusters and searches only within the most relevant ones to reduce search space drastically [Johnson et al., 2019b]. This figure 2.11 demonstrates how the IVF algorithm divides the dataset into clusters and conducts the search only within the most relevant ones.

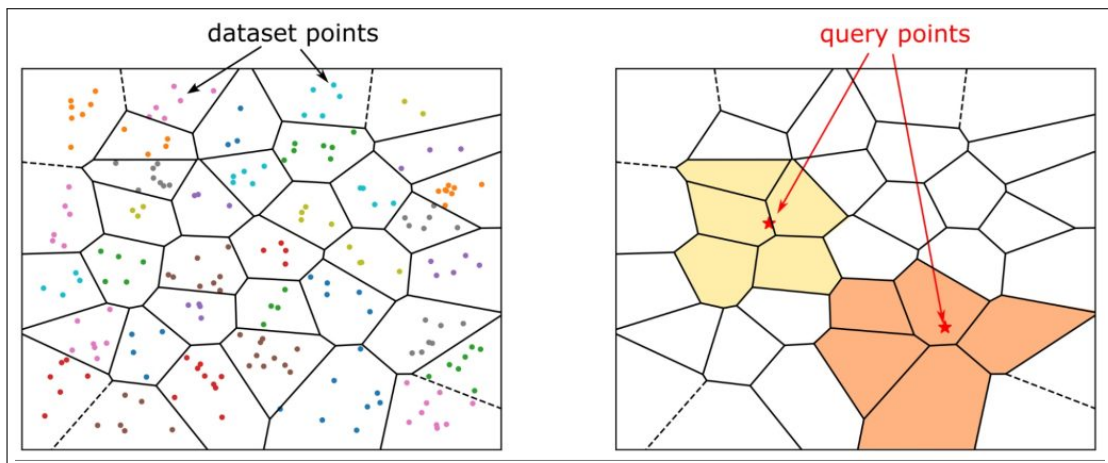


Figure 2.11: Illustration of IVF clustering and search process [UnfoldAI, 2024].

Flat Index

Flat Index (FI) or Brute Forces searches through all vectors with no approximation it is another name for the brute-force search we saw earlier, which is also done by KNN. Thus, all vectors are stored in a single index structure without any hierarchical organization [Johnson et al., 2019b]. The figure 2.12 shows the brute-force approach used in flat indexing, where all vectors are linearly scanned to identify the nearest neighbors. While accurate, this method is not scalable for large datasets

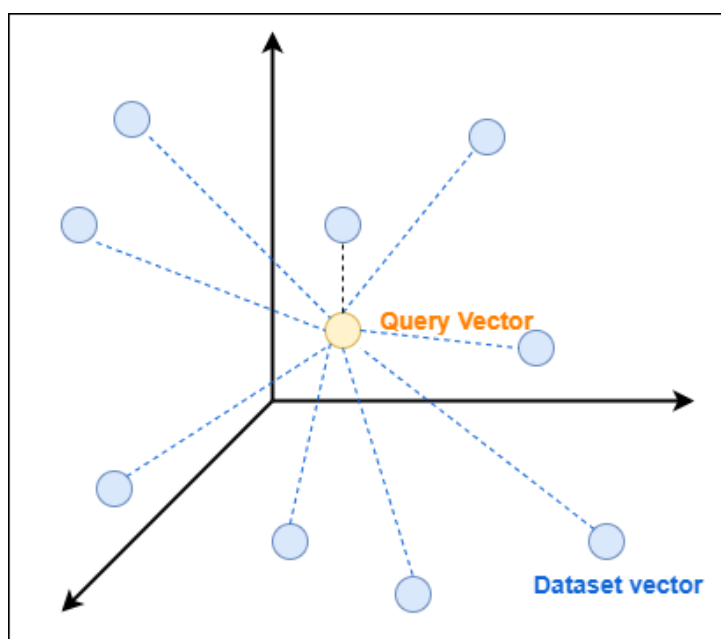


Figure 2.12: Illustration of FI .

To better understand the characteristics of these indexing methods, the following table 2.11 provides a comparative summary in terms of speed, accuracy, scalability, and ideal use cases:

Table 2.11: Comparison of Indexing Algorithms

Indexing Algorithm	Type	Speed	Accuracy	Scalability	Best Use Case
HNSW	Approximate (Graph)	High	High	High	Large datasets, fast ANN search
IVF	Approximate (Clustering)	Medium	Medium-High	Very High	Very large-scale applications
Flat	Exact (Brute Force)	Low	Very High	Low	Small datasets with high precision

In our solution, we adopted **HNSW indexing** for the **semantic similarity search** and this choice because It offers a **high recall and fast retrieval timewi**, which is essential in chatbots, and it is well-supported by **PGVector** The table below provides recommended **hyperparameters** based on the specific objective: The table below provides recommended **HNSW hyperparameters** based on the specific objective:

Objective	M	ef_construction	ef
High Precision	48	400	200
Low Latency (Faster Search)	16	100	50
Balanced Trade-off	32	200	100

Table 2.12: Recommended HNSW Parameters Based on Application Needs

Explanation of Parameters:

- **M:** Number of bi-directional links created for each node. Higher M improves accuracy but increases memory usage.
- **ef_construction:** Controls the quality of the indexing phase. Larger values lead to better graph structure but slower indexing.
- **ef:** Size of the candidate list used during search. Increasing ef improves recall but also increases latency.

This parameter tuning enables tailoring HNSW behavior to the performance and accuracy requirements of the application.

2.5 Contextual Retrieval

Once the document chunks have been transformed into vector embeddings and stored in a vector database, the next critical step in a RAG pipeline is *contextual retrieval*. This process is responsible for identifying and retrieving the most semantically relevant pieces of information from the knowledge base based on a user's query. These retrieved chunks form the input context for the language model responsible for answer generation.

2.5.1 Retrieval Process

The retrieval process typically consists of the following stages:

- **Query Embedding:** The user's question is encoded into a vector using the same embedding model used for the documents.

- **Similarity Search:** The system compares the embedded query with all document embeddings using a similarity metric (e.g., cosine similarity).
- **Top- k Selection:** The top- k most relevant document chunks are selected based on their similarity scores.
- **Context Assembly:** The selected chunks are aggregated into a coherent context window that will be provided to the language model during the generation phase.

2.5.2 Retrieval Strategies

Several advanced retrieval strategies have been developed to enhance retrieval effectiveness and reduce redundancy. These strategies are often implemented using frameworks such as LangChain and LlamaIndex.

Table 2.13: Retrieval strategies used in RAG systems.

Method	Description	Typical Use Case
Similarity Search	Retrieves the top- k most similar chunks to the query.	Standard approach in most RAG systems.
MMR (Max Marginal Relevance)	Promotes both relevance and diversity in retrieved chunks.	Prevents redundant information in responses.
Hybrid Search	Combines dense (vector-based) and sparse (keyword-based) retrieval.	Enhances retrieval when keywords are important.
Metadata Filtering	Filters chunks based on metadata such as author, document type, or date.	Useful when structured metadata is available.

In our system, we use a similarity search method to find the most relevant parts of the documents for each user question. To build a complete and helpful context, we combine two types of document chunks:

Document Chunks: 3 chunks taken from regular document splitting

QnA Chunks: 3 chunks taken from question-and-answer style splitting

Total Context: Up to 6 chunks are used per query

We only keep chunks that have a **similarity score of 0.6 or higher** measured with cosine similarity. This helps us make sure that the selected chunks are closely related to the user's question.

The entire retrieval process comprising the query embedding, vector store interaction, similarity computation using cosine similarity, HNSW indexing, and final context

construction is illustrated in 2.13, which provides a schematic overview of our retrieval strategy.

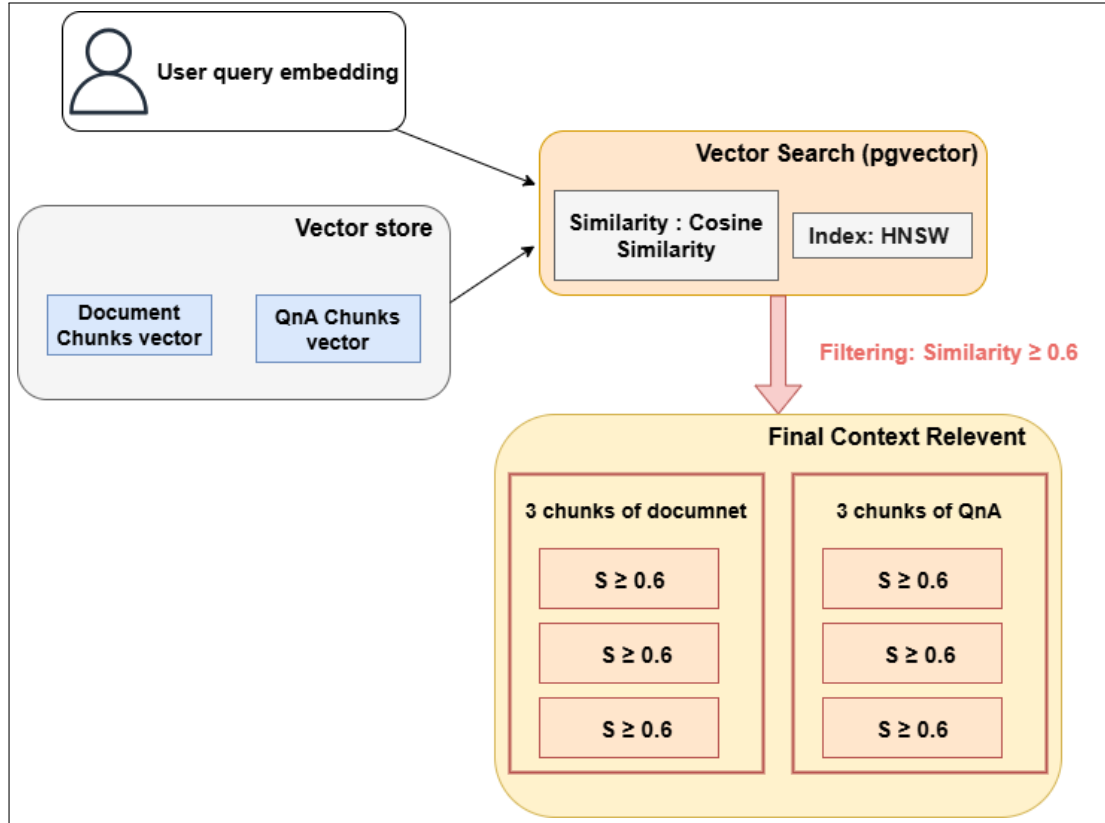


Figure 2.13: Vector-Based Similarity Retrieval using HNSW in pgvector

2.6 Answer Generation

The final stage of the RAG pipeline involves generating a response using a LLM. This response is conditioned on both the retrieved contextual information and the original user query.

To initiate generation, a structured prompt is composed. This prompt typically includes:

- The set of retrieved document chunks (context).
- The user’s original query.
- Optional system-level instructions (e.g., tone of answer, language specification, constraints).

The resulting prompt is then passed to a large language model for response generation. All these components are illustrated in Figure 2.14, which depicts the pipeline and steps of the generation process, showing how the retrieval component feeds relevant context to the generator along with system prompts and the user query to produce the final response.

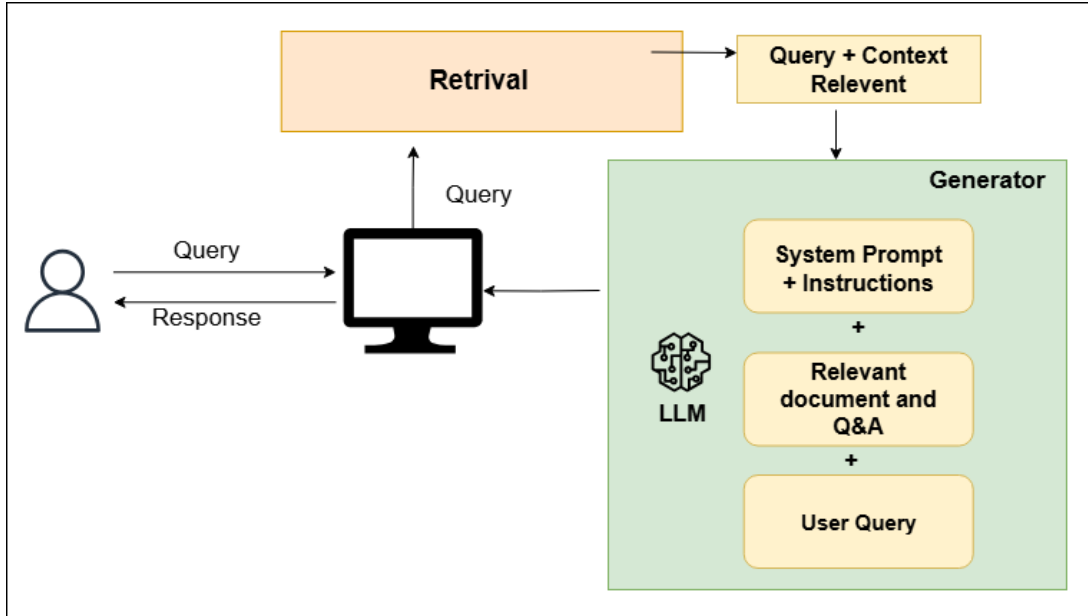


Figure 2.14: Illustration of the RAG Answer Generation Pipeline

2.6.1 Local Deployment with Ollama

For this implementation, we use Ollama local model deployment to ensure data privacy and eliminate dependence on external APIs. Ollama is an open-source tool that provides a streamlined approach to running large language models locally on personal computers and servers. It handles model downloading, quantization, and inference optimization automatically, making it accessible for researchers and developers to deploy state-of-the-art language models without requiring cloud services or extensive technical configuration [Ollama, 2024]. This approach maintains acceptable performance for real-time applications while ensuring complete control over data processing and model execution.

2.6.2 Language Model Selection

The selection of the LM for answer generation is a critical component of the RAG pipeline. The model determines the fluency, factual accuracy, and latency of the final response. In this study, we focus on three specific open-source models deployed locally through Ollama: **LLaMA 3 8B**, **Qwen 2.5 7B**, and **Mistral 7B Instruct v0.3**. These models were

selected not only for their performance but also because they are multilingual and offer strong support for the French language, which is the primary language of our data and users.

Qwen3-8B

Qwen3 is the latest generation of LLMs developed by Alibaba Cloud in 2025. It includes both dense and Mixture-of-Experts (MOE) variants. The Qwen3-8B model stands out for its ability to switch between two processing modes: a “thinking” mode (optimized for reasoning, mathematics, and coding) and a “non-thinking” mode (faster and more generic responses) [Cloud, 2025].

Note: MOE is an architecture where only a subset of model parameters are activated per input, improving efficiency without degrading performance.

Mistral-7B-v0.3

Mistral-7B-v0.3 is a high-performance yet lightweight open-source model, developed by Mistral AI. It integrates an advanced Sliding Window Attention (SWA) mechanism, enabling efficient handling of long-context tasks without excessive memory usage [AI, 2024b].

Note: SWA is a technique that segments long inputs into overlapping windows for scalable attention computation.

Meta LLaMA 3-8B

The **Meta LLaMA 3-8B** model represents the third generation of Meta’s LLaMA series, significantly improving reasoning and instruction-following capabilities. Although multilingual, it maintains a strong focus on English-centric datasets and benchmarks [AI, 2024a].

Comparative Table of Language Models

Table 2.14 provides a comparative summary of the LMs selected for local deployment in our RAG system. These models were chosen based on their ability to run efficiently on local environments while maintaining strong performance for multilingual applications

Table 2.14: Comparative summary of the selected LMs.

Model	Parameters	Context Length	License	Key Strengths	Developer	Ollama Command	Local Storage (GB)
Qwen3-8B	8B	Up to 128K	Apache 2.0	Multilingual, long-context, strong reasoning	Alibaba Cloud (QwenLM)	<code>ollama run qwen3:8b</code>	5.2
Mistral-7B-v0.3	7B	8K (with SWA)	Apache 2.0	Efficient, lightweight, fast inference	Mistral AI	<code>ollama run mistral:7b-instruct-v0.3</code>	4.4
LLaMA 3-8B	8B	8K (extendable)	Meta License	Reasoning, code generation, instruction-following	Meta AI	<code>ollama run llama3:8b</code>	4.7

2.6.3 French Language Performance Benchmarks

Based on the official French government LLM leaderboard developed by the French Ministry of Education, Inria, LNE, CNRS and GENCI in collaboration with Hugging Face, the performance of our selected models on French language tasks can be evaluated.

This leaderboard evaluates open-source models specifically on tasks relevant to French, reflecting the importance of linguistic and cultural adaptation in AI systems designed for francophone contexts.

- **Instruction Following Evaluation (IFEval):** Measures the model’s ability to understand and follow instructions in French, which is crucial for interactive applications.
- **Graduate-level Problem Question Answering (GPQA):** Assesses the model’s scientific and specialized knowledge through doctoral-level question answering.
- **Baccalaureate Exam (BAC):** Tests academic understanding based on the French high school final exam, using a culturally specific dataset provided by the Ministry of Education.

Although the **Qwen 3** model, released in May 2025, has not yet been officially evaluated on the French LLM leaderboard, its predecessor **Qwen 2.5** provides valuable insights into

the capabilities of the model family. Similarly, although the 7B version of LLaMA 3 was not listed, we referred to the results of the smaller model `LLaMA-3.2-3B-Instruct` to estimate potential performance.

To better understand the performance we added larger models in the same families, we also considered results from high-performing versions such as `Qwen2.5-72B-Instruct`, `LLaMA-3.3-70B-Instruct`, and `Mistral-Large-Instruct-2411` [Coordination Nationale pour l'IA et al., 2025].

Table 2.15: Performance of LM on French Benchmarks

Model Family	Overall AVERAGE	IFEval FR	GPQA FR	BAC FR
Mistral-7B-Instruct-v0.3	21.88%	37.77%	7.94%	19.92%
LLaMA-3.2-3B-Instruct	16.04%	27.16%	8.67%	12.29%
Qwen 2.5	14.48%	15.37%	19.17%	8.90%
<i>Reference (Large Models)</i>				
LLaMA-3.3-70B-Instruct	38.98%	53.35%	34.07%	29.52%
Mistral-Large-Instruct-2411	37.03%	61.24%	26.25%	23.59%
Qwen2.5-72B-Instruct	33.48%	50.53%	17.70%	32.20%

As shown in Table 2.15, larger models such as shown in Table 2.15, smaller models like `Mistral-7B-Instruct-v0.3` and `LLaMA-3.2-3B-Instruct` deliver solid performance given their size. `Mistral-7B` excels particularly in instruction following (IFEval), while `LLaMA-3.2-3B` shows balanced results across tasks. These findings suggest that models with 7 to 8 billion parameters provide an effective trade-off between accuracy and resource requirements, making them suitable for local deployment.

2.6.4 LLM Configuration Parameters

2.6.5 LLM Parameters

To call an LLM in a RAG system, several key parameters need to be defined to control the behavior and output of the model. These parameters govern aspects such as creativity, length of the generated response, and diversity of the output. The following table presents the main LLM parameters used in a RAG system configuration, along with their roles and recommended value ranges:

Table 2.16: Configuration parameters for LLM in a RAG system [OpenAI, 2025].

Parameter	Description	Range
temperature	Controls the randomness and creativity of the output. Higher values (e.g., 1.0) yield more random responses, while lower values (e.g., 0.0) produce deterministic ones.	0.0 – 1.0
max_tokens	Defines the maximum number of tokens in the generated response. Useful for controlling response length in real-time systems.	100 – 500 tokens
top_p	Applies nucleus sampling, selecting from the top p cumulative probability mass. Balances diversity and coherence.	0.0 – 1.0
frequency penalty	Penalizes repetition by lowering the probability of previously used tokens. Enhances diversity in word choice.	0.0 – 2.0
presence penalty	Discourages the reuse of tokens already present in the prompt or prior outputs. Promotes novelty and avoids redundancy.	0.0 – 2.0

2.7 Conclusion

This chapter outlined the full RAG pipeline—from chunking and vector storage to retrieval and answer generation. We chose pgvector for its simplicity and integration with PostgreSQL, semantic search for retrieval, and used models that support the French language, like Mistral for generating answers in French. All these components work together to build an efficient and smart legal assistant.

Chapter 3

Implementation and Evaluation

3.1 Introduction

In this chapter, we present the detailed implementation process and evaluation strategy of the proposed legal RAG-based chatbot system, JuriBot, designed to support employees at Djezzy in accessing regulatory and legal information efficiently. The chapter begins by outlining the development environment and technologies employed during the system's construction. We then define a multi-faceted evaluation framework covering semantic similarity, lexical overlap, contextual quality, and system performance.

To ensure a comprehensive assessment of the chatbot's responses, we complement automatic evaluation metrics with a human centered evaluation involving legal experts at Djezzy. Additionally, we explore the role of Large Language Models (LLMs) as evaluators in the absence of ground truth. The chapter concludes with a presentation of the JuriBot web application.

3.2 Development Environment

In this section, the hardware and software tools employed for the development and evaluation of the RAG-based legal chatbot, JuriBot, are presented.

Hardware Environment

For the development of the legal RAG application, two machines were used: a personal laptop and a laboratory machine (referred to as LBO) provided by the university.

The personal machine was primarily employed for local development, preprocessing of legal documents, vector indexing with pgvector , and testing using a lightweight language

model deployed via Ollama. In contrast, the LBO machine was used for more demanding tasks such as embedding generation using transformer-based models, fine-tuning and evaluation .

The retrieval component, which included semantic search, was tested on both machines to compare response times and memory usage. The LBO machine proved essential during the evaluation retrieval and generation , which required extended RAM and faster I/O performance.

Table 3.1 summarizes the technical specifications of the machines used in the development and testing phases.

Table 3.1: Summary of the technical specifications of the machines used

Machine	Processor	RAM	Storage	GPU	OS	Role
Developer Laptop	Intel Core i5-11700H	16 GB	512 GB SSD	None	Windows 11	Local dev, testing
LBO Machine	Intel Xeon W-2235 @ 3.80GHz	128 GB	1 TB NVMe SSD + 2 TB HDD	NVIDIA RTX 4090 (24 GB VRAM)	Windows 11 Pro	Embedding generation, semantic search, fine-tuning, evaluation

Software Environment

The creation of our legal chatbot application required a complete software stack that incorporates the web development framework, advanced natural language processing libraries and privacy preservation techniques. Our technology selection prioritized data privacy, system performance, and development efficiency. We implemented a local-first strategy for language model deployment utilizing Ollama to protect company document data within our controlled environment, while employing PostgreSQL with the pgvector extension to develop a hybrid database solution adept at efficiently managing both structured and vector data. The table below summarizes all the tools and technologies used during the development process:

Table 3.2: Development Tools and Technologies

Category	Tool	Description and Usage in Project
Programming Languages	Python 3.11	Main programming language used for backend logic, data processing, and LLM integration.
	JavaScript	Frontend development for interactive user interface components.
	HTML5/CSS3	Web interface structure and styling implementation.
Frameworks and Libraries	Django	Web framework for backend API development and web application architecture.
	LangChain	Used to build the RAG pipeline and manage document processing, embedding, retrieval flows, and prompt chain.
	PyMuPDF4llm	PDF text extraction and document preprocessing for knowledge base creation.
Language Models and Embeddings	Ollama	Used to run LLMs locally, such as Mistral, Qwen, and LLaMA, to ensure offline, private, and performant generation capabilities.
	Hugging Face Transformers	Embedding models for document vectorization and query processing.
Database Systems	PostgreSQL 7	Primary relational database for structured data storage and user management.
	pgvector Extension	Vector database extension for storing and querying document embeddings.
Development Tools	Visual Studio Code	Integrated development environment for code editing and debugging.
	Git	Version control system for collaborative development and code management.
Analysis and Visualization	Jupyter Notebook	Data analysis, model evaluation, and performance visualization.
	Matplotlib	Statistical charts and performance metrics visualization.

3.3 Evaluation Metrics Definition

To comprehensively evaluate the performance of our system, we adopt for evaluation metrics that assess semantic similarity, lexical overlap, contextual relevance, and system performance.

3.3.1 Semantic Similarity Metrics

Cosine Similarity quantifies the semantic closeness between the generated and reference responses by measuring the cosine of the angle between their respective embedding vectors:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (3.1)$$

where \mathbf{A} and \mathbf{B} are the embedding vectors of the generated and reference responses respectively, and n denotes the dimensionality of the embedding space.

3.3.2 Lexical Overlap Metrics

BLEU Score

Bilingual Evaluation Understudy measures the degree of n-gram precision by comparing n-grams in the generated response with those in the reference response. The BLEU score is defined as:

$$\text{BLEU} = \text{BP} \cdot \exp \left(\sum_{n=1}^N w_n \log p_n \right) \quad (3.2)$$

where:

- p_n is the modified n-gram precision for n-grams of length n
- w_n is the weight for each n-gram order commonly $w_n = \frac{1}{N}$
- N is the maximum n-gram order, typically set to 4
- BP is the brevity penalty, defined as:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \quad (3.3)$$

Here, c is the length of the candidate response, and r is the length of the reference response.

ROUGE-L Score

evaluates the quality of generated text based on the longest common subsequence (LCS) between the candidate and reference texts. The ROUGE-L score is computed using the F-measure as:

$$\text{ROUGE-L} = \frac{(1 + \beta^2) \cdot R_{\text{lcs}} \cdot P_{\text{lcs}}}{\beta^2 \cdot R_{\text{lcs}} + P_{\text{lcs}}} \quad (3.4)$$

where:

$$R_{\text{lcs}} = \frac{\text{LCS}(X, Y)}{m} \quad (3.5)$$

$$P_{\text{lcs}} = \frac{\text{LCS}(X, Y)}{n} \quad (3.6)$$

In this context, $\text{LCS}(X, Y)$ is the length of the longest common subsequence between sequences X (reference) and Y (candidate), m is the length of the reference, n is the length of the candidate, and β controls the trade off between recall and precision (typically $\beta = 1$).

3.3.3 Contextual Quality Metrics

BERTScore (F1)

leverages contextual embeddings from pretrained BERT models to evaluate semantic similarity at the token level. The F1 variant of BERTScore is defined as the harmonic mean of precision and recall:

$$\text{BERTScore-F1} = 2 \cdot \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \quad (3.7)$$

The precision and recall are computed as follows:

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_i \in \hat{x}} \max_{x_j \in x} \mathbf{x}_j^\top \hat{\mathbf{x}}_i \quad (3.8)$$

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_j \in x} \max_{\hat{x}_i \in \hat{x}} \mathbf{x}_j^\top \hat{\mathbf{x}}_i \quad (3.9)$$

where $x = \{x_1, x_2, \dots, x_k\}$ and $\hat{x} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_l\}$ represent the token sets of the reference and candidate texts, respectively. The vectors \mathbf{x}_j and $\hat{\mathbf{x}}_i$ are the contextual embeddings derived from a BERT model.

This figure illustrates the computational workflow of BERTScore evaluation

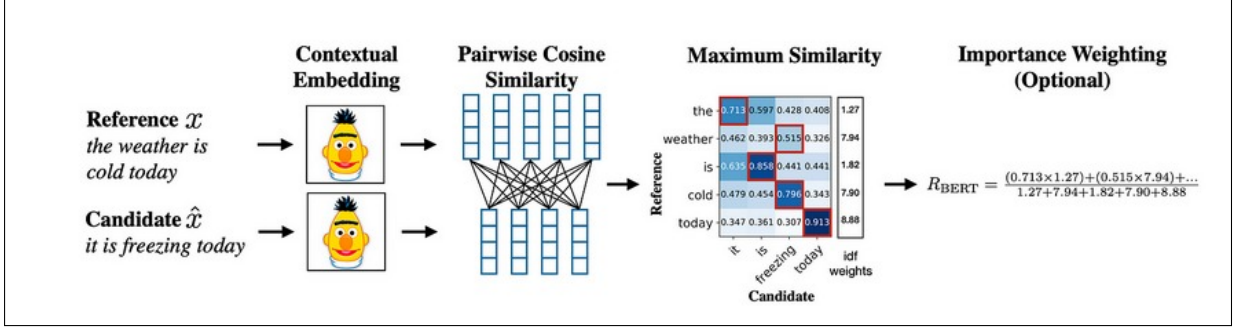


Figure 3.1: BERTScore Computation Process [Hugging Face, 2019]

3.3.4 System Performance Metrics

Response Time

quantifies the latency between query submission and the delivery of the final response. It is computed as:

$$T_{\text{response}} = T_{\text{retrieval}} + T_{\text{generation}} \quad (3.10)$$

where:

- $T_{\text{retrieval}}$ is the duration required to retrieve relevant documents,
- $T_{\text{generation}}$ is the time taken for the language model to generate a response,

Token Count

refers to the number of tokens in the generated response, providing insight into response verbosity and model behavior. It is formally defined as:

$$\text{Token Count} = |\text{tokenize}(\text{response})| \quad (3.11)$$

where $\text{tokenize}(\cdot)$ is the tokenizer function specific to the embedding model used.

Summary Table of Evaluation Metrics

To summarize the evaluation criteria presented above, we illustrate in Table 3.3 the main indicators used to evaluate our system this table highlights the category of each metric, its main objective, and the specific stage of the evaluation process at which it is applied.

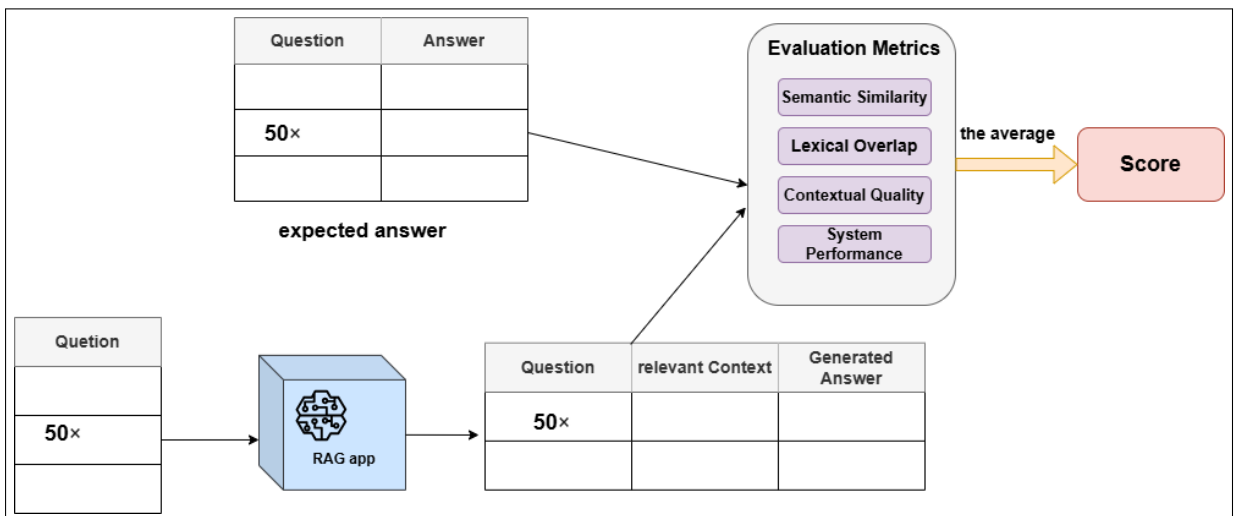
Table 3.3: Overview of Evaluation Metrics Used in the System

Metric Category	Metric	Purpose	Evaluation Stage
Semantic Similarity	Cosine Similarity	Measures semantic similarity between embeddings	Retrieval and Post-generation
Lexical Overlap	BLEU Score	Assesses n-gram overlap with reference responses.	Post-generation
	ROUGE-L Score	Measures longest common subsequence between texts.	Post-generation
Contextual Quality	BERTScore (F1)	Evaluates contextual similarity using BERT embeddings.	Post-generation
System Performance	Response Time	Measures total time from query to response.	End-to-end performance
	Token Count	Measures output length in number of tokens.	Post-generation

3.4 Evaluation Results

In this section, we analyze the results of our RAG legal assistant system across several evaluation metrics. Each result is averaged over a set of 50 legal questions from our evaluation datasets. The analysis involves a comparative study of three language model *Mistral*, *LLaMA*, and *Qwen* under four distinct retrieval configurations: *chat_naive* without indexing, and different HNSW parameters *hnsw_16_100_50*, *hnsw_32_200_100*, and *hnsw_48_400_200*.

Figure 3.2 illustrates the different steps followed during the evaluation process.

**Figure 3.2:** Solution Evaluation Steps.

For this purpose, we created an evaluation sheet composed of **50 questions**. Each entry in this final sheet contains the following components to calculate the score:

Question: A user query representing a realistic legal information need.

Retrieved Context: The document chunks retrieved by the retrieval module in response to the question.

Ground Truth Answer: The expected or reference answer that the system is supposed to generate.

Generated Answer: The response generated by our RAG system based on the retrieved context.

The following tables and figure summarize the averaged results per model and configuration:

Cosine Similarity

Table 3.4 and the figure 3.3 present the average result Cosine Similarity results

Table 3.4: Cosine Similarity Scores Across RAG Configurations and Language Models

Configuration	Mistral	LLaMA	Qwen
chat_naive	0.664	0.647	0.524
hnsw_16_100_50	0.668	0.648	0.522
hnsw_32_200_100	0.653	0.640	0.535
hnsw_48_400_200	0.669	0.644	0.523

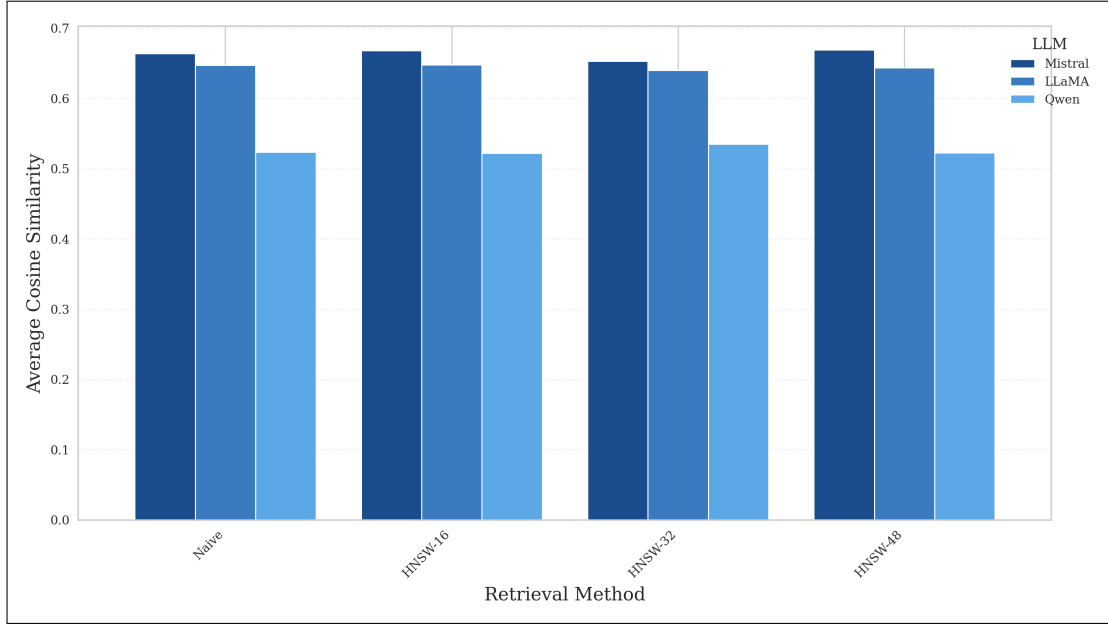


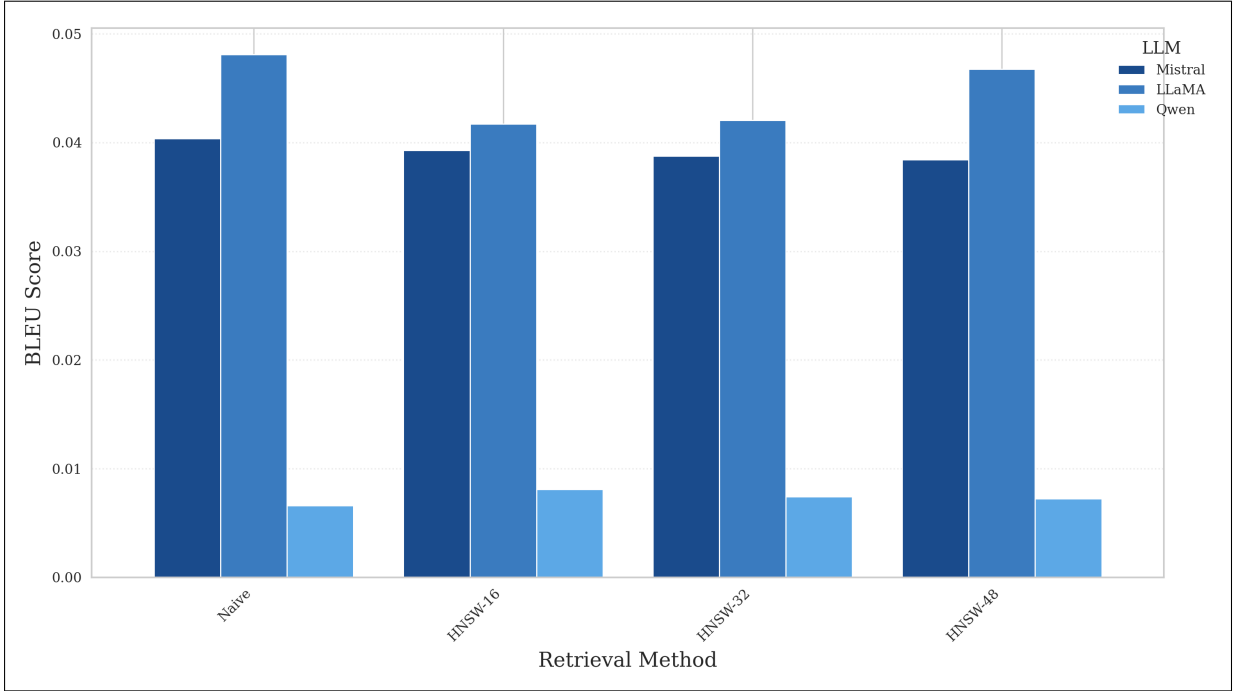
Figure 3.3: Cosine Similarity Scores visualization

We observe that Mistral performs better across all configurations, with a peak of 0.669 in *hnsw_48_400_200*. LLaMA is close behind, while Qwen trails with scores around 0.52–0.53. This confirms that Mistral better align with reference embeddings, especially with optimized retrieval.

BLEU Score

Table 3.5: Lexical Precision Analysis BLEU Score per Configuration

Configuration	Mistral	LLaMA	Qwen
chat_naive	0.0403546	0.0481094	0.00661872
hnsw_16_100_50	0.0392833	0.0417098	0.00808007
hnsw_32_200_100	0.0387495	0.0420329	0.00743188
hnsw_48_400_200	0.0384406	0.0467462	0.00724863

**Figure 3.4:** BLEU Score visualization

We observe that BLEU scores metrics present in table 3.5 and their visualization in figure 3.4 are generally low (<0.05), which is expected in generation tasks like legal QA. Mistral and LLAMA perform similarly and Qwen performs poorly with values around 0.006–0.008. The low BLEU values highlight the limitation of using strict n-gram overlap in evaluating generative models in our system RAG.

ROUG-L Score

Table 3.6: ROUGE-L Score Across RAG Configurations and Language Models

Configuration	Mistral	LLaMA	Qwen
chat_naive	0.203361	0.198452	0.074578
hns_w_16_100_50	0.197653	0.195919	0.0798206
hns_w_32_200_100	0.200961	0.196781	0.0755899
hns_w_48_400_200	0.199013	0.206612	0.0769895

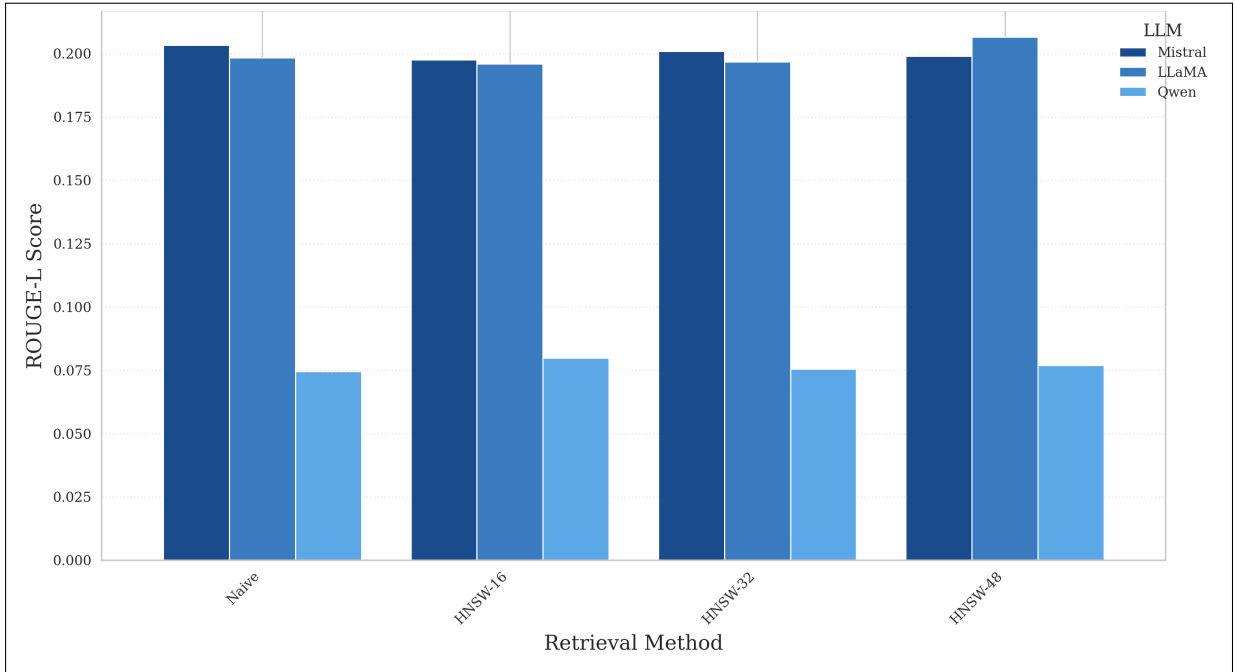


Figure 3.5: ROUGE-L by model and index setup

In Table 3.6, Mistral and LLAMA lead again with ROUGE-L scores around 0.20, while Qwen lags at 0.07–0.08. This further supports the semantic evaluations. LLAMA with *hns_w_48_400_200* achieves the highest ROUGE-L (0.206), suggesting that richer context helps it generate response closer to reference responses.

BERTScore

Table 3.7: Contextual Quality Evaluation BERTScore

Configuration	Mistral	LLaMA	Qwen
chat_naive	0.877376	0.872804	0.823332
hns_w_16_100_50	0.875758	0.871411	0.827881
hns_w_32_200_100	0.876162	0.872973	0.826282
hns_w_48_400_200	0.874569	0.874672	0.822507

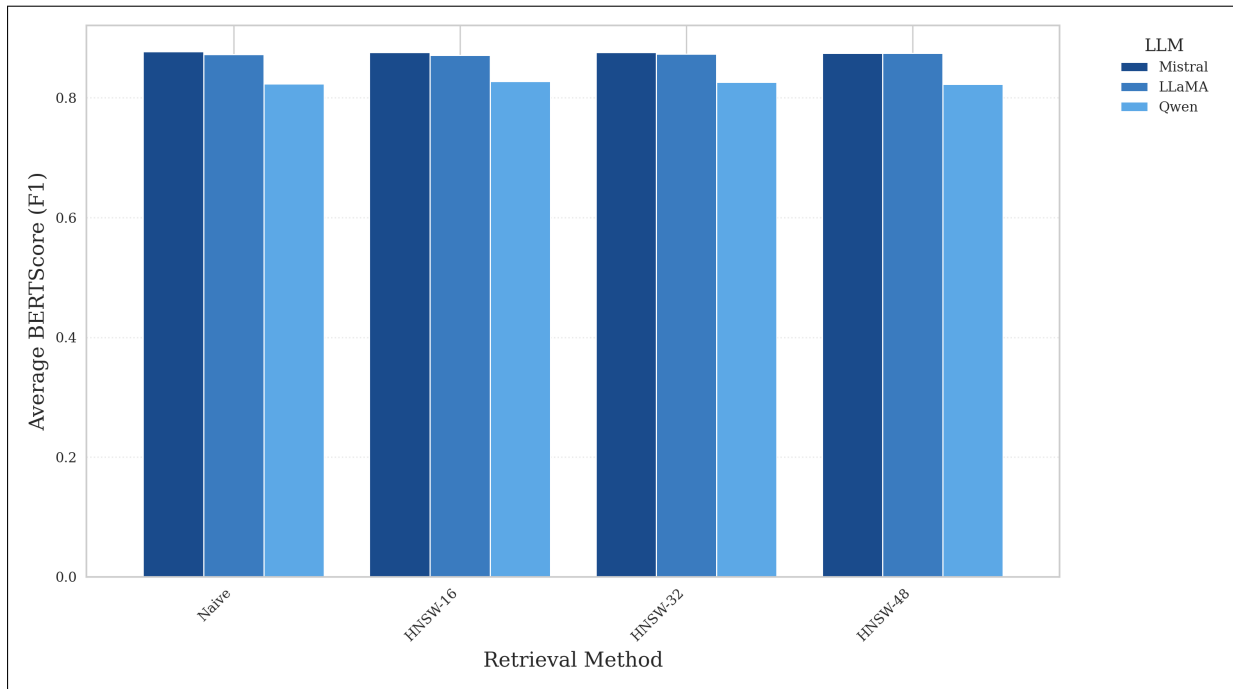


Figure 3.6: Graphe BERTScore per configuration and models

As shown in both Table 3.7 and Figure 3.6, All models show high context meaning. Mistral and LLaMA achieve high BERTScores (0.87–0.88), reflecting strong contextual matching. Qwen scores lower (0.82), consistent with earlier observations. LLaMA shows stable performance across retrieval strategies, while Mistral slightly drops with deeper indexes.

Response Time

Table 3.8: Average Response Time Analysis (seconds)

Configuration	Mistral	LLaMA	Qwen
chat_naive	1.89213	1.63277	6.94787
hns_w_16_100_50	1.87766	1.72128	7.11957
hns_w_32_200_100	1.88681	1.64489	6.81234
hns_w_48_400_200	1.97894	1.62298	7.19957

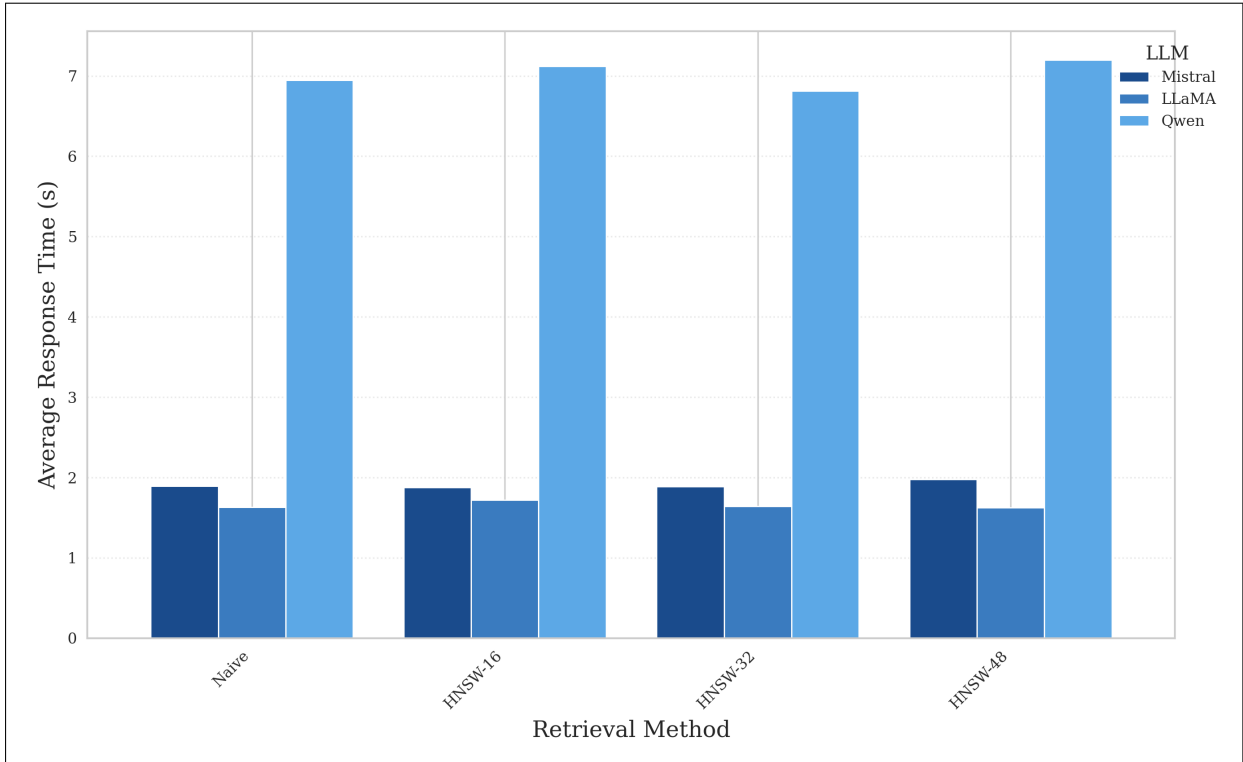
**Figure 3.7:** Average Response Time Comparison

Table 3.8 presents the average response time for each configuration. Mistral and LLaMA exhibit low latency, typically under 2 seconds, while Qwen is significantly slower, averaging over 6 seconds in all configurations. The fastest setup is *hns_w_48_400_200* with LLaMA (1.63 s), and the slowest is Qwen with *hns_w_48_400_200*. These results indicate that Mistral and LLaMA are more suitable for realtime legal assistant applications.

Token Count

Table 3.9: Average Token Count score

Configuration	Mistral	LLaMA	Qwen
chat_naive	155.894	149.149	900.404
hns_w_16_100_50	154.255	150.426	865.745
hns_w_32_200_100	157.149	144.553	875.128
hns_w_48_400_200	168.660	146.766	911.447

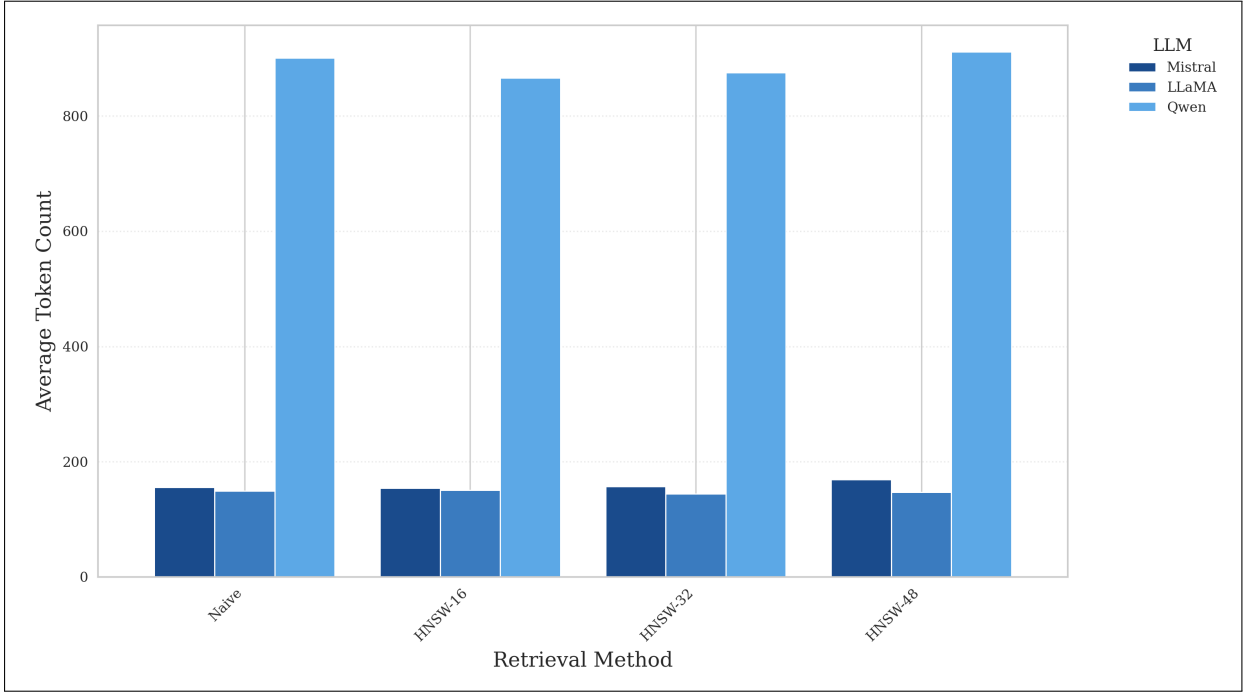


Figure 3.8: Token Generation Distribution by Model

Table 3.9 summarizes the average token counts. Qwen generates much longer responses (over 860 tokens), while Mistral and LLaMA remain below 170 tokens. The verbose output from Qwen may indicate over-generation or insufficient context filtering. The *hns_w_48_400_200* setup leads to the longest answers for all models, possibly due to larger context windows retrieved.

3.5 Analysis and Discussion Metric result

The results show that **Qwen** produces more tokens, which implies the longest response time. This is because **Qwen** “think before answering,” which may explain both the increased number of generated tokens and the longer processing duration. but , this comes at the cost of lower similarity scores and reduced BLEU performance.

LLaMA has good balance between speed and response quality, especially with the *hns_w_48_400_200* configuration. This makes it a strong candidate in both efficiency and accuracy are required.

In general, **Mistral** and **LLaMA** often produce very similar outputs, which makes it difficult to clearly see the differences between retrieval configurations. This is likely due to the small size of the evaluation dataset, which constrains the ability to detect statistically significant variations.

3.6 LLM as judge

The concept of LLM as a Judge refers to the use of a large language model to act as an evaluator or judge of other LLM generated responses. Instead of relying solely on human annotators, an LLM is instructed through a detailed prompt to assess the quality of generated answers against ground-truth references using specific criteria. This method enables scalable, consistent, and cost-effective evaluation across large datasets.

This evaluation approach has gained traction in recent research, especially in domains where expert-level judgements are necessary, such as medicine and law in our case. [Zheng et al., 2023, Chiang et al., 2023] demonstrated that well-instructed LLMs can produce judgements that align closely with human expert evaluations, making them valuable tools for benchmarking generative models in specialized context .

Methodology used

We employed the LLM as Judge using two evaluation protocols prompt types, both conducted on the same set of 50 legal question-answer pairs. For each case, we provided the evaluation LLM with the original question, the ground truth what mean expected answer, and the response generated by a Qwen , Mistral, or LLaMA.

Prompt 1: Single Score Evaluation

In this setting, the evaluating LLM was instructed to provide a single overall score (from 1 to 5) based on three criteria: legal accuracy, clarity, and relevance. This produces global judgment for each response without breaking it down into submetrics.

Prompt 2: Multi-Metric Evaluation

The second approach used a more detailed evaluation framework where the LLM judge assessed each response across eight distinct criteria are :accuracy,completeness, clarity, relevance, pertinence or context adequacy , completeness and coherence .

Evaluation Results and Analysis

In both evaluation protocols in Table 3.10 show that **Qwen** continuously performed the best, demonstrating its strength in legal accuracy, clarity, and relevance.

Qwen received the highest average score (3.94) in the first evaluation *Prompt 1*, which used a single overall score based on general legal quality. **LLaMA** (3.74) and **Mistral**

Table 3.10: Comparison of LLMs Based on *Prompt 1* and *Prompt 2* Evaluation Metrics

Metric	Qwen	LLaMA	Mistral	Ranking Notes
Prompt 1: Mean Scores				
Mean Score	3.94	3.74	3.42	Qwen (1), LLaMA (2), Mistral (3)
Prompt 2: Multi-Metric Scores (Averaged)				
Accuracy	4.15	3.87	3.89	
Completeness	4.34	3.66	3.84	
Clarity	4.43	4.34	4.31	
Relevance	4.62	4.11	4.31	
Pertinence	4.45	3.98	4.09	
Completeness	4.15	3.55	3.78	
Coherence	4.68	4.40	4.47	
Global Score	4.30	3.83	3.89	Qwen (1), Mistral (2), LLaMA (3)

(3.42) came after it. With a focus on fluency and immediate relevance, this ranking most likely represents the evaluating LLM’s overall assessment of each response.

The second evaluation, *Prompt 2*, on the other hand, used a more detailed, multimetric approach that covered eight different dimensions, including coherence, accuracy, and completeness. In this case, **Mistral** (3.89) marginally outperformed **LLaMA** (3.83), but **Qwen** continued to be the best-performing model with a global average of (4.30).

Qwen consistently ranked first in both evaluation settings, demonstrating the best overall performance in terms of legal accuracy, clarity, and relevance. But between the two evaluations, **Mistral** and **LLaMA** were in different positions. **LLaMA** placed second in *Prompt 1*, but **Mistral** marginally beat it in *Prompt 2*.

This shift implies that their performances are extremely similar, and the evaluation technique may have an impact on such slight variations. Additionally, it confirms that rankings become less stable when scores are tight, particularly when moving from a global impression *Prompt 1* to a detailed breakdown *Prompt 2*. This emphasizes how crucial *multimetric evaluation* is for a more accurate and nuanced comparison.

3.7 Human evaluation by Legal Experts at Djazzy

A jurist from the host company, **Djazzy**, participated in a human evaluation to complement the evaluation metrics. The objective was to assess the practical utility and legal relevance of the responses generated by the three tested models **Qwen**, **LLaMA**, and

Mistral in a real world legal context.

During the evaluation, the jurist posed a set of representative questions, including both *complex legal scenarios* requiring in depth reasoning and *frequently asked legal questions by employees*. The responses from each model were then compared and categorized based on their *legal sufficiency, clarity, and conciseness*.

The table 3.11 below presents the jurist’s ranking of the models:

Table 3.11: Jurist Ranking of Model Responses

Model	Simple Legal Questions	Complex Legal Questions
LLaMA	1 st	2 nd
Qwen	2 nd	1 st
Mistral	3 rd	3 rd

Overall, due to its concise and straightforward answers, **LLaMA** was preferred for addressing *frequent employee queries*.

However, for *more complex legal issues*, the jurist ranked **Qwen** highest, highlighting its detailed and contextually rich responses. Although **Mistral** produced generally acceptable answers, it consistently ranked lower in both simple and complex cases.

3.8 Conclusion of Evaluation

To conclude, we summarise our main observations from each evaluation perspective in the table 3.12 :

Table 3.12: Summary of Evaluation Insights Across Methods

Evaluation Type	Evaluation Summary	Implications for the System
Automatic Metrics	Lexical metrics (BLEU, ROUGE) show poor correlation with actual quality in legal QA. Semantic metrics (BERTScore, Cosine Similarity) better capture relevance and fluency.	Lexical overlap is not sufficient for generative legal tasks; semantic metrics are more informative.
LLM-as-Judge	Qwen ranked highest in both single and multi-metric evaluations; Mistral and LLaMA were very close, with rankings shifting depending on evaluation style.	LLMs can reliably assess response quality, especially in domain-specific contexts like legal QA.
Human Evaluation	Legal expert preferred LLaMA for employee questions (clarity and speed) and Qwen for complex legal reasoning. Mistral ranked consistently lower.	The assistant is primarily used by employees, so LLaMA is more practical. However, Qwen is retained for expert-level queries.

Based on these findings, we draw the following conclusions:

- Lexical metrics alone are inadequate for evaluating generative systems in legal domains, semantic and contextual metrics provide more meaningful insights.
- LLM-based evaluators offer a promising, scalable alternative to human judgment, especially when detailed, multi-criteria prompts are used.
- Human evaluation confirms the practical preference for LLaMA, due to its faster response time and concise answers in everyday usage scenarios.
- To ensure flexibility, we decided to deploy both Qwen and LLaMA, allowing the system to adapt to different query types and user needs.

This hybrid deployment strategy ensures a balance between efficiency, usability, and legal reasoning capability, making the assistant suitable for both employees and legal experts at **Djezzy**.

3.9 System Demonstration JuriBot Web Application for a Legal RAG Chatbot

This section presents the final version of our developed web application, named JuriBot, designed specifically for the legal and compliance department at Djezzy. It describes the full usage flow and features available to both regular users employees and administrators .

Figure 3.9 illustrates the login interface of the system.

3.9.1 User Authentication and Access Control

The section presents our system’s user interaction with login and chatbot interface

Login Interface

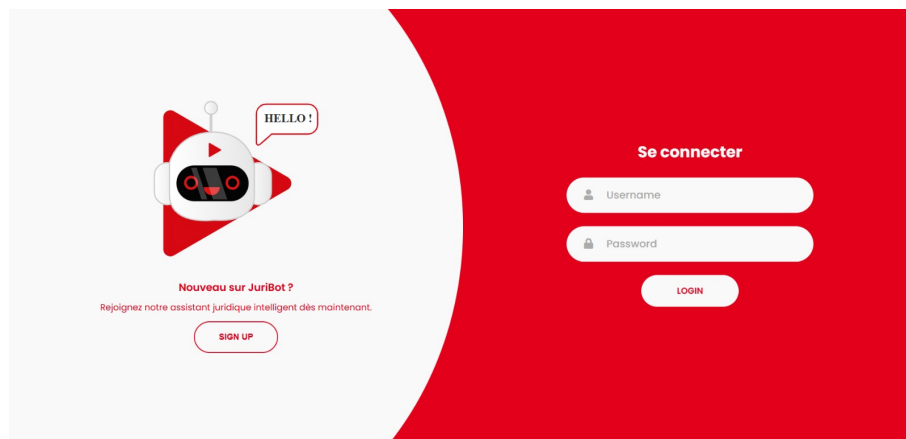


Figure 3.9: Login Interface

The application begins with a secure authentication process, requiring users to enter valid credentials to gain access to the chatbot functionality. This login mechanism ensures that only authorized personnel can interact with the legal chatbot and access sensitive legal information.

Upon successful authentication, users are automatically redirected to the main chatbot interface, where they can begin interacting with the system.

Chatbot Interaction Interface

Figure 3.10 presents the main chatbot interface of the application.

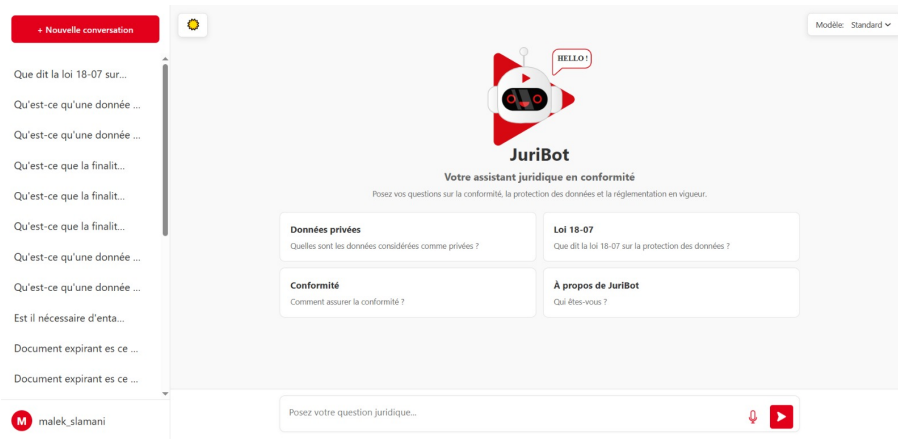


Figure 3.10: Chatbot Interface (juribot)

In this interface, users can pose legal questions related to compliance, data privacy, or regulatory matters and receive real-time assistance from **JuriBot**. The interface offers several key features:

- **Text Input:** Users can type their legal inquiries directly into the chat interface.
- **Voice Input:** Through integration with the e Speech-to-Text API, users can verbally communicate their questions.
- **Frequently Asked Topics:** Provides pre-defined shortcuts to common legal topics for quicker access.
- **Conversation History:** Maintains contextual continuity throughout the user session for coherent dialogue.

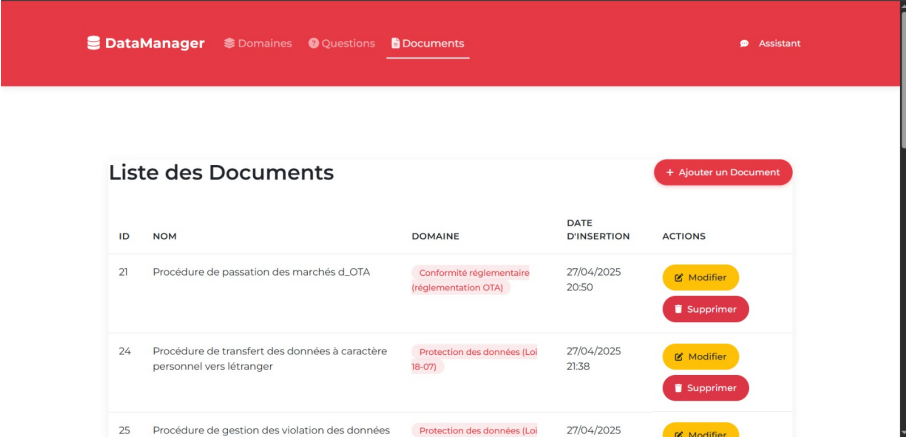
The chatbot interface is designed with usability in mind, ensuring an intuitive and efficient experience for legal professionals in need of rapid legal guidance.

3.9.2 Administrative Interface for Content Management

The admin has access to manage the dataset to add document and QA pairs .

Document Management System

Figure 3.11 displays the list of uploaded documents within the data manager section of the application.



ID	NOM	DOMAINE	DATE D'INSERTION	ACTIONS
21	Procédure de passation des marchés d_OTA	Conformité réglementaire (réglementation OTA)	27/04/2025 20:50	<input checked="" type="checkbox"/> Modifier <input type="button" value="Supprimer"/>
24	Procédure de transfert des données à caractère personnel vers l'étranger	Protection des données (Loi 18-07)	27/04/2025 21:38	<input checked="" type="checkbox"/> Modifier <input type="button" value="Supprimer"/>
25	Procédure de gestion des violations des données	Protection des données (Loi 18-07)	27/04/2025	<input checked="" type="checkbox"/> Modifier

Figure 3.11: Document List in Data Manager

The interface includes clearly defined columns for:

- **Document Title:** Displays the name or title of the legal document.
- **Document Type:** Indicates the category or format of the document.
- **Upload Date:** Shows the date the document was added to the system.
- **Available Actions:** Provides options to view, edit, or delete each document.

This structured and organized interface allows administrators to efficiently access and manage legal documents, helping to ensure the legal knowledge base remains accurate, up-to-date, and comprehensive.

Document Upload Interface

Figure 3.12 illustrates the document addition form interface.

Figure 3.12: Add Document Form

This component allows administrators to input relevant metadata for new legal documents, including :

- **Document Title:** The name or title assigned to the legal document.
- **Document Type:** Specifies the nature or category of the document.
- **Creation Date:** Indicates when the document was originally created.
- **File Upload Functionality:** Enables the attachment of document files for inclusion in the system.

This structured input form enhances the overall document management workflow within the Legal Chatbot Application, promoting efficient access, classification, and retrieval of legal information.

3.10 System Workflow and Processing Pipeline

Figure 3.13 illustrates the complete architecture and operational workflow of the *JuriBot* system, developed to support legal compliance tasks within the Djezzy organization.

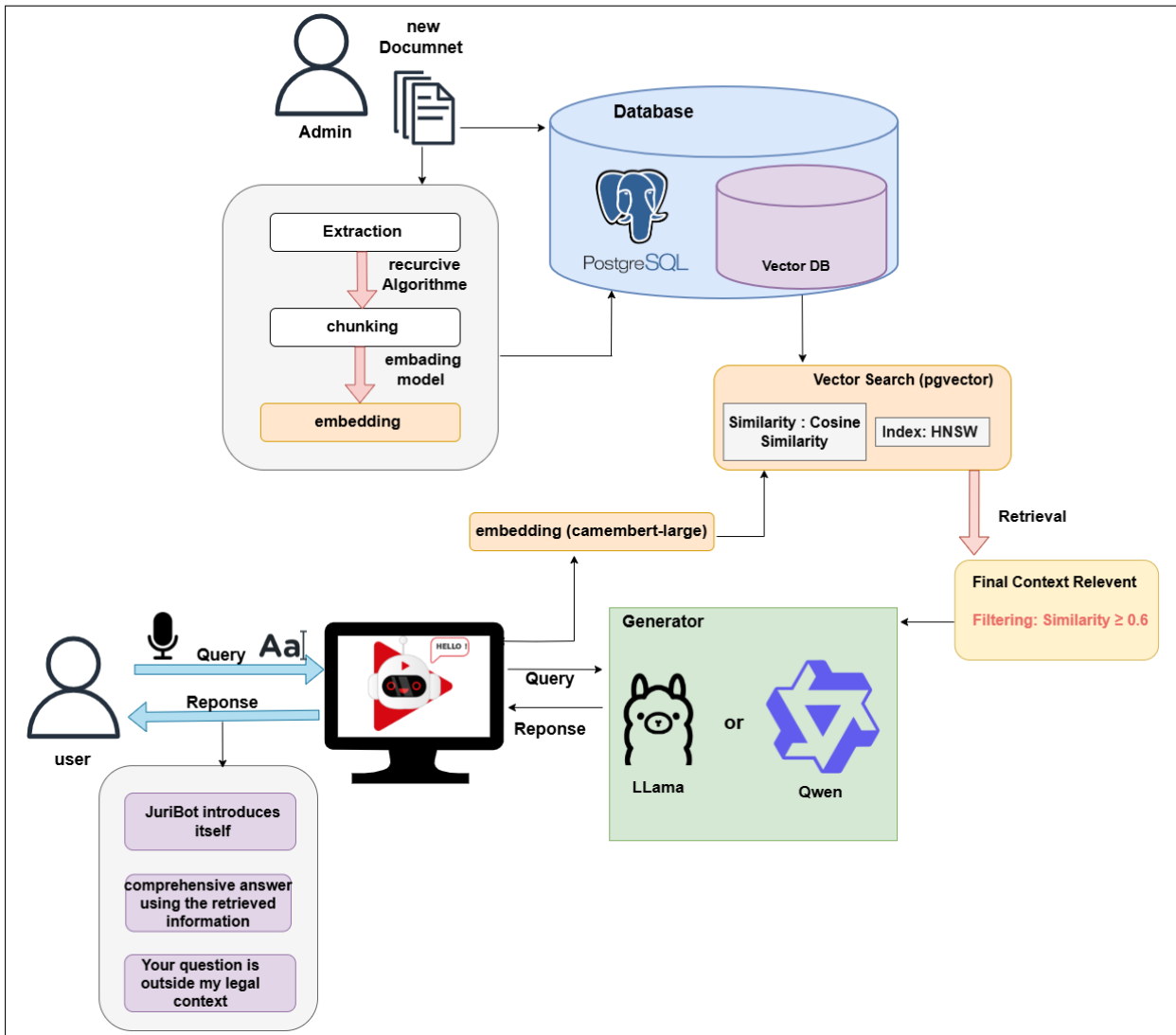


Figure 3.13: System Workflow and Processing Pipeline of the JuriBot Legal Chatbot

The pipeline begins with the **administrator interface**, which enables authorized personnel to upload legal documents along with associated question–answer (QA) pairs. Once uploaded, the documents undergo processing via a recursive chunking algorithm, followed by embedding using a French pre-trained language model, specifically *CamemBERT*. The resulting dense vector representations are then stored in a PostgreSQL database extended with the `pgvector` module, which allows for hybrid storage that includes both traditional relational metadata and vector embeddings.

When a user interacts with the chatbot—through either voice or text input—the system embeds the incoming query using the same embedding model and performs semantic similarity search over the stored vectors. This retrieval operation uses cosine similarity, and results are filtered based on a similarity threshold (typically ≥ 0.6). The filtered

contextual content is then forwarded to a large language model (LLM), such as *LLaMA* or *Qwen*, for final response generation.

The LLM response strategy is designed to be context-sensitive and follows three primary response paths:

- **Greeting Response:** If the user input matches typical greeting patterns (e.g., “Bonjour”, “Hello”, “Who are you?”), the system returns a self-introduction such as: *“Bonjour, je suis JuriBot, votre assistant juridique...”*
- **Contextual Response:** When relevant legal content is retrieved, the LLM generates a detailed, context-aware response using the information from the database.
- **Out-of-Context Response:** If no relevant context is found, the chatbot responds with a polite fallback message: *“Your question is outside my legal context. I can only answer questions based on the legal knowledge in my database.”*

3.11 Conclusion

This chapter demonstrated the successful implementation and thorough evaluation of the JuriBot legal chatbot. We developed a functional RAG-based system capable of retrieving and generating accurate legal responses in French. Through a combination of semantic, lexical, and performance metrics, as well as human evaluation by legal experts, we confirmed the system’s effectiveness and relevance in a real-world context. The evaluation revealed that *LLaMA* and *Qwen* models performed best in terms of clarity and legal relevance. The deployment of a user web application also validated the system’s practical usability within Djezzy’s legal environment. Overall, the results confirm that our objectives were achieved and the proposed solution is both technically sound and contextually appropriate.

General conclusion

Synthesis of Contributions

This thesis presented the design and implementation of an intelligent legal assistant based on the Retrieval-Augmented Generation (RAG) architecture, tailored to the internal needs of Djezzy’s legal department. The system aimed to improve employees’ access to legal and regulatory information in a multilingual setting, with a focus on clarity, compliance, and relevance.

Throughout this project, we successfully built a working RAG pipeline using advanced open-source language models such as Qwen, LLaMA, and Mistral. The system was evaluated using both automatic metrics (e.g., BERTScore, Lexical Overlap) and a human evaluation conducted by a legal expert from the company. The results confirmed the effectiveness of the approach, highlighting the quality of responses in terms of accuracy, fluency, and usefulness in a corporate legal context.

In addition to achieving the initial objectives, the project introduced various components such as chunking strategies, vector-based retrieval using embeddings, and a user-friendly interface. These efforts collectively demonstrate the feasibility and benefits of adopting generative AI systems to support legal departments in large organizations.

However, certain limitations remain. The dataset used was limited in scope and did not fully cover all legal subdomains within the company. As a result, the evaluation may not reflect performance across all possible legal questions. Expanding the system to handle more diverse legal documents and internal policies would improve its generalizability and robustness.

Future Perspectives

In conclusion, this work provides a solid foundation for integrating generative AI into legal information systems. It demonstrates how language models, when combined with retrieval mechanisms, can support legal departments in offering fast, accurate, and context-aware responses. The system developed herein opens up future directions for more adaptive,

explainable, and legally aligned AI tools within enterprise environments.

Looking ahead, several perspectives can be explored to enhance the system:

- Allowing users to upload legal PDF documents and interact with them via a chat-with-PDF functionality;
- Fine-tuning the underlying models using reinforcement learning from human feedback (RLHF) to further improve the quality of responses;
- Implementing an analytics dashboard for administrators to monitor and analyze common employee questions, enabling better legal communication and training strategies;
- Enabling legal experts to download the source documents retrieved by the system, thus increasing transparency and legal traceability.

Bibliography

- [Adamopoulou and Moussiades, 2020] Adamopoulou, E. and Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2.
- [AI, 2024a] AI, M. (2024a). Meta llama 3: Open foundation and instruction models. <https://ai.meta.com/blog/meta-llama-3/>. Accessed: 2025-06.
- [AI, 2024b] AI, M. (2024b). Mistral 7b and mixtral: High-performance open language models. <https://mistral.ai/news/mistral-7b/>. Accessed: 2025-06.
- [Banu Prakash and Saravanan, 2023] Banu Prakash, R. and Saravanan, S. (2023). MED-BOT: An ai doctor using rule-based chatbot for healthcare recommendations. *Research Square*. Preprint.
- [Bhat et al., 2024] Bhat, V., Sree, D., Cheerla, J., Mathew, N., Liu, G., and Gao, J. (2024). Retrieval augmented generation (rag) based restaurant chatbot with ai testability. In *Proceedings of [Conference Name if available]*.
- [Bhatt and Vaghela, 2023] Bhatt, A. and Vaghela, N. (2023). Med-bot: An ai-powered assistant to provide accurate and reliable medical information.
- [Chiang et al., 2023] Chiang, P.-C., Chen, A., Singh, A., et al. (2023). Can llms judge? rethinking benchmarking in the era of generative ai. *arXiv preprint arXiv:2305.10403*.
- [ChromaDB Developers, 2024] ChromaDB Developers (2024). Chromadb documentation. Technical documentation. Accessed: 2025-05.
- [Cloud, 2025] Cloud, A. (2025). Qwen2 and qwen3: High-performance open-source language models. <https://github.com/QwenLM/Qwen>. Accessed: 2025-06-.
- [Coordination Nationale pour l’IA et al., 2025] Coordination Nationale pour l’IA, Ministère de l’Éducation nationale, Inria, LNE, GENCI, and Hugging Face

- (2025). French language model leaderboard. https://huggingface.co/spaces/fr-gouv-coordination-ia/llm_leaderboard_fr. Accessed: 2025-06.
- [Dam et al., 2024] Dam, S. K., Hong, C. S., Qiao, Y., and Zhang, C. (2024). A complete survey on llm-based ai chatbots.
- [Hugging Face, 2019] Hugging Face (2019). Bertscore: Evaluating text generation with bert. Model repository. Accessed: 2025-05.
- [Hugging Face, 2021] Hugging Face (2021). dangvantuan/sentence-camembert-base. Model repository. Accessed: 2025-05.
- [Hussain et al., 2019] Hussain, S., Sianaki, O., and Ababneh, N. (2019). A survey on conversational agents/chatbots classification and design techniques. *Lecture Notes in Computer Science*, pages 946–956.
- [IBM, 2024] IBM (2024). Text embedding generation. IBM Documentation, Version 2.0.0. Accessed: 2025-05.
- [Johnson et al., 2019a] Johnson, J., Douze, M., and Jégou, H. (2019a). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.
- [Johnson et al., 2019b] Johnson, J., Douze, M., and Jégou, H. (2019b). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.
- [Kshirsagar, 2024] Kshirsagar, A. (2024). Enhancing rag performance through chunking and text splitting techniques. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 10:151–158.
- [Kumar S and Sheshadri, 2024] Kumar S, V. and Sheshadri, K. (2024). The voice assistants that connect you to your library, whether it is alexa, google, or siri. *Annals of Library and Information Studies*, 71:272–278.
- [Malick and Marone, 2024] Malick, F. and Marone, R. M. (2024). Conception d’un « chatbot » pour soutenir les services d’information dans les bibliothèques universitaires design of a "chatbot" to support information services in university libraries.
- [Malkov and Yashunin, 2020] Malkov, Y. A. and Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

- [Ollama, 2024] Ollama (2024). Ollama run large language models locally. Accessed: 2025-05.
- [OpenAI, 2025] OpenAI (2025). Openai completion parameters. Accessed: 2025-05-08.
- [Panchal et al., 2024] Panchal, D., Gole, A., Narute, V., and Joshi, R. (2024). Lawpal: A retrieval augmented generation based system for enhanced legal accessibility in india. *arXiv preprint*. To appear.
- [pgvector Contributors, 2024] pgvector Contributors (2024). pgvector documentation. <https://github.com/pgvector/pgvector>. Accessed: 2025-05.
- [Pinecone Systems Inc., 2024] Pinecone Systems Inc. (2024). Pinecone documentation. Technical documentation. Accessed: 2025-05.
- [Qdrant Team, 2024] Qdrant Team (2024). Qdrant documentation. Technical documentation. Accessed: 2025-05.
- [Sharma et al., 2021] Sharma, T., Rathore, T. S., and Rathore, N. S. (2021). Retrieval based chatbot system. Impact Factor: 5.354.
- [UnfoldAI, 2024] UnfoldAI (2024). Ivfflat vs hnsw: Vector search techniques compared. Accessed: 2025-05-07.
- [Wang, 2024] Wang, K. (2024). From eliza to chatgpt: A brief history of chatbots and their evolution. *Applied and Computational Engineering*, 39:57–62.
- [Wang et al., 2024] Wang, Z., Chu, Z., Doan, T. V., Ni, S., Yang, M., and Zhang, W. (2024). History, development, and principles of large language models: An introductory survey.
- [Weaviate Team, 2024] Weaviate Team (2024). Weaviate documentation. Developer documentation. Accessed: 2025-05.
- [Zheng et al., 2023] Zheng, S., Guo, Y., Zhao, W., Zhou, M., Wang, H., and Tang, J. (2023). Judging llm-as-a-judge: Benchmarking chatgpt and gpt-4 with legal human annotations. *arXiv preprint arXiv:2305.14688*.
- [Zilliz, 2024] Zilliz (2024). Hierarchical navigable small worlds (hnsw). Accessed: 2025-05-07.
- [Zilliz, 2024] Zilliz (2024). Milvus documentation. Technical documentation. Accessed: 2025-05.