POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Studying RAG architectures and Fine-Tuned LLMs for The Energy Trading Regulations Domain

Tesi di Laurea Magistrale in
Computer Science - Ingegneria Informatica

Author: **Francesco Pellegrini**

Student ID: 247807
Advisor: Prof. Marco Brambilla
Co-advisors: Mathyas Giudici, Cristina Costeniero, Matteo Bonanomi
Academic Year: 2025-26

Abstract

Large Language Models (LLMs) often lack the specific knowledge and structural formatting required for enterprise legal applications. This thesis compares different architectural solutions to adapt open-source LLMs to the Italian energy trading regulations domain. The primary scientific question addresses whether retrieving external documents or updating internal parameters provides a safer and more accurate legal assistant.

The research conducts an ablation study to compare four configurations: a Baseline model, Retrieval-Augmented Generation (RAG), Fine-Tuning (FT), and a Hybrid (RAG+FT) approach. The methodology implements a hybrid computing infrastructure. It utilizes a commercial cloud provider environment for RAG orchestration and centralized evaluation. Simultaneously, it leverages a supercomputer infrastructure to execute parameter-efficient fine-tuning. A synthetic Question-Answer (Q&A) dataset, generated by a model, simulates interactions with a corporate domain expert. The evaluation framework employs an LLM-as-a-judge methodology and a static Entity Recall metric to assess performances.

The experiments demonstrate distinct operational trade-offs. RAG successfully grounds the models in factual reality and significantly reduces hallucinations on specific legal queries. FT effectively adapts the stylistic output, teaching the models to follow strict formatting rules and adopt a professional persona. However, integrating both techniques exposes a critical safety tax. Classic fine-tuned models develop an over-reliance on their parametric memory. They frequently ignore the explicitly provided RAG context to force an answer. Lastly, the analysis reveals that massive models seamlessly process large context windows, while smaller architectures struggle.

These findings emphasize that organizations must carefully balance factual rigidity and stylistic adherence. The results provide a practical framework for deploying cost-effective, specialized enterprise legal assistants.

Keywords: Large Language Models, Retrieval-Augmented Generation, Fine-Tuning, Legal NLP, Energy Market Regulations

Abstract in lingua italiana

I Large Language Models (LLMs) spesso non possiedono la conoscenza specifica e il linguaggio tecnico per applicazioni legali. Questa tesi confronta soluzioni architetturali per adattare LLM open-source al dominio delle normative italiane sul commercio di energia. L'analisi principale ha lo scopo di capire se sia meglio utilizzare documenti esterni o aggiornare i parametri interni per ottenere un assistente legale più sicuro e preciso.

La ricerca conduce uno studio per confrontare quattro configurazioni: un modello non modificato, la Retrieval-Augmented Generation (RAG), il Fine-Tuning (FT) e un approccio ibrido (RAG+FT). Lo studio utilizza un ambiente di cloud computing commerciale per gestire la RAG e la valutazione delle risposte. In aggiunta, sfrutta un'infrastruttura di supercalcolo per eseguire un fine-tuning dei parametri. Un dataset sintetico di Domande-Risposte (Q&A), generato da un modello, simula le interazioni con un esperto di dominio aziendale. Il framework di valutazione impiega una metodologia LLM-as-a-judge e una metrica statica di Entity Recall per valutare le prestazioni.

Gli esperimenti dimostrano chiari compromessi operativi. La RAG àncora con successo i modelli alla realtà fattuale e riduce significativamente le allucinazioni su specifiche domande legali. Il FT adatta efficacemente lo stile delle risposte, insegnando ai modelli a seguire regole di formattazione rigorose e ad adottare un tono professionale. Tuttavia, l'integrazione di entrambe le tecniche espone a un problema in termini di sicurezza. I classici modelli sottoposti a fine-tuning sviluppano un'eccessiva dipendenza dalla loro memoria parametrica. Spesso ignorano il contesto RAG fornito esplicitamente per forzare una risposta. Infine, l'analisi rivela che i modelli massivi elaborano senza problemi ampie finestre di contesto, mentre le architetture più piccole faticano.

Queste scoperte sottolineano che le organizzazioni devono bilanciare attentamente il rigore fattuale e l'aderenza stilistica. I risultati forniscono un framework pratico per l'implementazione di assistenti legali aziendali specializzati e vantaggiosi in termini di costi.

Parole chiave: Large Language Models, Retrieval-Augmented Generation, Fine-Tuning, NLP Legale, Normative del Mercato Energetico

Contents

1 | Introduction

1.1. Context

Large Language Models (LLMs) fundamentally change how organizations process text and retrieve information [56]. These artificial intelligence systems generate human-like text by predicting token sequences. In enterprise environments, employees require fast and accurate access to complex documents. The energy trade market presents a specifically challenging environment for this technology. Energy regulations feature specialized terminology, strict formatting requirements, and zero tolerance for factual errors. Professionals need reliable enterprise tools to query these regulatory documents efficiently. Integrating LLMs into this workflow requires systems that can handle both the rigid language of the law and the dynamic nature of market rules.

1.2. Problem Statement

Standard open-source LLMs lack the specific knowledge of Italian energy trade market laws required for enterprise deployment. Furthermore, these baseline models fail to output answers with a specific structural format and with grounding details expected by corporate legal departments. When users ask baseline models about specialized topics, the models often hallucinate and fabricate incorrect information [22].

Developers primarily use two techniques to adapt models to new domains: Retrieval-Augmented Generation (RAG) and Fine-Tuning (FT). RAG connects the model to an external database to retrieve updated facts [29]. FT alters the internal weights of the model to teach it specific stylistic patterns and formatting rules [21]. However, current literature lacks targeted ablation studies that isolate and compare the performance of RAG, FT, and their combination within the highly specific domain of energy regulations.

1.3. Proposed Solution

This thesis conducts an ablation study to analyze open-source LLM performance when integrating RAG and FT versus a baseline model. The primary objective is to create a reliable enterprise conversational tool for employees to consult regulatory documents.

The project utilizes a hybrid computing infrastructure to manage the distinct requirements of generation and training. The cloud environment features an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance. This instance leverages Amazon Bedrock to run the RAG pipeline. The High-Performance Computing (HPC) environment utilizes the Leonardo Cineca cluster to execute the computationally heavy FT processes. The dataset comprises legal Portable Document Format (PDF) files and a company profile: based on this trusted set of documents a synthetic conversational test set is built.

The experimental workflow proceeds in four distinct stages:

1. **Baseline Testing:** The system tests the raw open-source models to establish a performance floor.
2. **RAG Implementation:** The system integrates a retrieval pipeline across all models to inject external context.
3. **Fine-Tuning:** The system trains the models on an Italian legal-style Question and Answer (QA) dataset. This step strictly teaches the models the required answer formatting and tone.
4. **Hybrid Integration:** The system combines the FT models with the RAG architecture. This final setup merges the retrieval of specific knowledge with the learned stylistic format.

The infrastructure handles inference on the AWS EC2 instance for the baseline and RAG steps. For the FT and hybrid steps, inference runs locally on the Leonardo HPC cluster, and the system exports the outputs back to EC2. Finally, the EC2 instance centralizes the evaluation. The evaluation framework employs a static metric alongside an LLM-as-a-judge method to quantify accuracy, style, and robustness [57].

1.4. Structure of the Thesis

This document organizes the research into six chapters:

- **Chapter 1 (Background)** outlines the relevant scientific results reused in this work and introduces the underlying technologies.

- **Chapter 2 (Related Work)** reviews existing literature that addresses knowledge injection and domain adaptation in legal NLP.
- **Chapter 3 (Main Idea of the Thesis)** describes the core solution, the ablation study design, and the evaluation framework.
- **Chapter 4 (Implementation Experience)** details the systems implemented to validate the idea, including the data pipelines and the hybrid infrastructure.
- **Chapter 5 (Experiments and Discussion)** reports the quantitative and qualitative results using real and synthetic settings, supported by graphs and critical analysis.
- **Chapter 6 (Conclusions)** summarizes the work done, explores limitations regarding applicability and validity, and suggests possible future work.

2 | Background

2.1. Large Language Models (LLM)

Large Language Models (LLMs) represent a significant advancement in Natural Language Processing (NLP), characterized by their ability to process, understand, and generate human-like text across a wide variety of domains. These models rely on deep learning techniques to estimate the probability distribution of a sequence of words or tokens. Fundamentally, an LLM functions as a probabilistic system that predicts the next token in a sequence based on the context provided by preceding tokens.

The capabilities of these models scale with the number of parameters, the size of the training dataset, and the amount of compute used during training, a relationship often referred to as "scaling laws" [25]. While early implementations of language models relied on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, current state-of-the-art LLMs predominantly utilize the Transformer architecture.

2.1.1. The Transformer Architecture

Introduced by Vaswani et al. in 2017 [50], the Transformer architecture shifted the paradigm from sequential processing (common in RNNs) to a mechanism that allows for parallelization, significantly reducing training times for large datasets.

The core innovation of the Transformer is the *Self-Attention* mechanism. This mechanism allows the model to weigh the importance of different words in a sentence relative to one another, regardless of their positional distance. This addresses the long-term dependency problem that plagued previous architectures, where models struggled to retain context over long sequences.

Mathematically, the attention function maps a query and a set of key-value pairs to an output. The output is a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. The scaled dot-product attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Where:

- Q represents the Query matrix.
- K represents the Key matrix.
- V represents the Value matrix.
- d_k is the dimension of the keys, used as a scaling factor to prevent vanishing gradients in the softmax function.

Transformers utilize Multi-Head Attention, which runs this attention mechanism in parallel multiple times with different linear projections. This allows the model to attend to information from different representation subspaces at different positions simultaneously.

The original Transformer architecture consists of two main components:

1. *Encoder*: Processes the input sequence and generates a high-dimensional vector representation (embeddings). Architectures like BERT (Bidirectional Encoder Representations from Transformers) utilize this component exclusively and excel at discriminative tasks such as text classification and named entity recognition.
2. *Decoder*: Uses the encoder's output (in encoder-decoder setups) or previous tokens (in decoder-only setups) to generate the target sequence.

For generative tasks, which are the focus of this thesis, *Decoder-only* architectures have become the standard. Models such as the GPT (Generative Pre-trained Transformer) family and Llama operate by attending only to earlier tokens in the sequence (causal masking) to predict the subsequent token.

2.1.2. Open-Source Models and the Open-Weights Paradigm

The landscape of LLMs divides primarily into proprietary models and open-source (or open-weights) models [56]. Proprietary models, such as GPT-4 or Claude, offer access only via API, treating the model weights, training data, and specific architectural details as trade secrets. This "black box" nature poses challenges for enterprise applications that require strict data governance, reproducibility, and customization [3].

Conversely, open-source models provide public access to the model weights and, frequently, the architecture code. This thesis specifically examines this category of models. The

release of models like Llama 2, Llama 3, and Mistral has democratized access to high-performance LLMs, allowing researchers and organizations to run inference locally or on private clouds.

This accessibility is critical for specific use cases, such as the energy trade market law domain addressed in this work. It allows for:

- *Data Privacy*: Sensitive legal documents remain within the organization's infrastructure (e.g., AWS VPC or Leonardo Cineca) rather than passing through external APIs. This drastically reduce the risk of data leakage, because private information are not spread with the external world.
- *Customizability*: Open weights enable advanced adaptation techniques, including the Fine-Tuning strategies discussed in Section 1.3, to align the model with specific stylistic or domain requirements.
- *Transparency*: Ablation studies require full control over the system configuration to isolate variables. Open models allow for precise measurements of how different components (like RAG context or Fine-tuning) impact performance without hidden variables changing on the provider's side.

Recent benchmarks indicate that top-tier open-source models now rival the performance of proprietary models on many reasoning and comprehension benchmarks, making them a viable solution for industrial applications.

2.2. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a technique designed to address a fundamental limitation of Large Language Models: the "frozen weights" problem. Once a model completes its training phase, its internal knowledge base becomes static, rendering it incapable of accessing events or information generated after the training cutoff [26, 29].

Crucially, recent research comparing knowledge injection methods demonstrates that while Fine-Tuning is effective for adapting model behavior, RAG is significantly more robust for retrieving and utilizing "less popular" or dynamic knowledge, such as specific legal statutes, without suffering from hallucination to the same degree as fine-tuned models [46]. RAG decouples the model's reasoning capabilities—its ability to process and generate language (the "skill to speak")—from its knowledge base, allowing for the injection of fresh, dynamic information without the computational cost of retraining.

2.2.1. Architectural Overview

The RAG workflow integrates a retrieval system with a generative model. Instead of relying solely on parametric memory (knowledge stored in weights), the system queries a non-parametric memory (an external database) to find relevant context. The process follows three distinct phases:

1. **Indexing**: This offline phase involves preparing the external corpus source. Documents are split into meaningful chunks or segments of text. A balance must be struck in chunk size; chunks must be large enough to contain semantic meaning but small enough to fit efficiently within the model's context window. These chunks are transformed into high-dimensional vectors using an embedding model and stored in a vector database. The final output of this step is a vector index, required for the subsequent retrieval phase.

2. **Retrieval**: Upon receiving a user query, the system converts the query into a vector using the same embedding model. It then performs a similarity search—typically using metrics such as cosine similarity or Euclidean distance—to identify the top k most relevant chunks from the vector database. Algorithms like FAISS (Facebook AI Similarity Search) facilitate efficient similarity search even in high-dimensional spaces.

3. **Generation**: The retrieved chunks are concatenated with the original prompt to form an augmented context. The LLM then generates a response based exclusively on this augmented input, synthesizing the external information to answer the query accurately while using the new knowledge base as a reliable and up-to-date source.

2.2.2. Advanced Optimization Strategies

While a standard RAG pipeline improves factual accuracy, several advanced techniques exist to mitigate hallucinations and improve retrieval relevance:

- *Reranking*: Standard retrieval often prioritizes semantic similarity over precise relevance. A two-stage process can ameliorate this: a fast initial retrieval fetches a broad set of candidates, which are then reassessed by a specialized reranking model (cross-encoder) to select the highest-quality contexts for the generation step.

- *Query Rewriting*: User queries are often ambiguous or lack necessary keywords. An intermediate LLM step can rewrite or expand the query to optimize it for vector retrieval, removing meaningless tokens and resolving coreferences.

- *Legal-Specific Grounding*: For domain-specific applications like energy law, simple retrieval is often insufficient. Benchmarks such as LexRAG [30] emphasize the necessity of citation-based grounding in multi-turn consultations. This ensures the model not only provides the correct legal interpretation but explicitly links its reasoning to the specific source text, a requirement for auditability.
- *Thresholding and Autocut*: To prevent the model from answering with irrelevant information, systems can enforce similarity thresholds. If the distance between the query and retrieved chunks exceeds a specific gap (autocut), the system instructs the model to state it does not know the answer rather than hallucinating based on unrelated context.
- *GraphRAG*: A limitation of vector-only retrieval is the loss of structural relationships between entities. GraphRAG integrates Knowledge Graphs to maintain these connections, allowing the system to traverse related concepts that might not be semantically similar in vector space but are logically connected.

2.2.3. Implementation and Trade-offs

Frameworks such as LlamaIndex provide the tooling necessary to orchestrate these pipelines, managing data ingestion and retrieval logic. Evaluation of these systems remains complex: tools like RAGAS (Retrieval Augmented Generation Assessment) employ "LLM-as-a-judge" methodologies to quantify metrics such as context precision and answer faithfulness [12].

The primary advantage of RAG is its ability to provide up-to-date, domain-specific information while offering transparency through citation of source documents. However, this comes with architectural trade-offs. The retrieval step introduces latency, downgrading real-time performance compared to a standalone LLM. Furthermore, the reliance on an external database introduces complexity regarding storage costs and the necessity for robust data processing pipelines.

2.3. Fine-Tuning (FT)

Fine-tuning represents the process of adapting a pre-trained Large Language Model (LLM) to a specific task or domain by updating its weights on a focused dataset. While the pre-training phase instills general linguistic competence and world knowledge, fine-tuning is essential for aligning the model's output with specific user instructions, safety guidelines, and organizational tone.

Fundamentally, Fine-Tuning acts as a mechanism for "baking in" context and intuition directly into the model's parameters. Unlike Retrieval-Augmented Generation (RAG), which supplements the model with external data at runtime, Fine-Tuning adjusts the internal weights through back-propagation to minimize the loss between predicted outputs and targeted responses. It is a form of supervised learning requiring high-quality input-output pairs. The model does not necessarily learn new facts during this process, but rather learns how to process existing knowledge, recognizing domain-specific patterns and adhering to a specialized style [58].

2.3.1. Trade-offs and Operational Considerations

Implementing Fine-Tuning introduces specific operational characteristics compared to dynamic prompting or RAG:

- *Inference Efficiency*: Fine-tuned models often allow for shorter prompts, as instructions and examples do not need to be provided in-context. This reduces latency and cost per token during inference.
- *Deep Domain Expertise*: The process specializes the model for specific use cases, such as adopting a corporate persona or specific legal reasoning format, which prompt engineering alone cannot consistently achieve.
- *Catastrophic Forgetting*: A significant risk is the degradation of the model's general capabilities while learning specialized ones. As the weights shift to optimize for the narrow dataset, the model may lose proficiency in general reasoning or unrelated domains.
- *Knowledge Cutoffs*: Unlike RAG, Fine-Tuning does not solve the knowledge cutoff issue. Information is static; once training concludes, the model has no access to new data without retraining.

2.3.2. Parameter-Efficient Fine-Tuning (PEFT)

Full fine-tuning, which updates all parameters of a model (often billions), is computationally prohibitive for most research environments. To address this, Parameter-Efficient Fine-Tuning (PEFT) methods adapt models by updating only a small fraction of parameters while freezing the pre-trained backbone.

LoRA (Low-Rank Adaptation)

Low-Rank Adaptation (LoRA) [21] injects trainable rank decomposition matrices into the layers of the Transformer architecture. For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA expresses the weight update as $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are low-rank matrices with $r \ll \min(d, k)$. During forward propagation, the output is computed as:

$$h = W_0 x + \Delta W x = W_0 x + BAx \tag{2.2}$$

This significantly reduces memory requirements, as gradients are calculated only for A and B. Empirical observations suggest specific heuristics for optimal configuration:

- *Target Modules*: LoRA should be applied across all linear layers, not merely the Key and Value matrices, to maximize performance.
- *Hyperparameters*: Adjusting the rank (r) is essential. A standard heuristic for the scaling factor α is to set it at twice the rank's value ($2r$).
- *Stability*: Despite inherent randomness in GPU training, outcomes remain remarkably consistent across multiple runs, provided the dataset is static. However, excessive iteration (multi-epoch training) on static datasets often leads to overfitting and result deterioration.

QLoRA and Hardware Optimization

QLoRA (Quantized LoRA) [11] further optimizes this by quantizing the frozen base model weights to 4-bit precision (using the NormalFloat4 data type). This presents a tangible trade-off: it offers approximately 33% memory savings at the cost of a 39% increase in training runtime.

Regarding optimizers, contrary to common belief, the choice between AdamW and SGD with schedulers shows minimal variation in outcomes for fine-tuning tasks. While Adam is often labeled memory-intensive due to its momentum parameters, this does not significantly impact peak memory demands, as the majority of memory is allocated to large matrix multiplications. QLoRA enables the fine-tuning of 7 billion parameter models on a single GPU with as little as 14 GB of RAM, making it feasible for the Leonardo Cineca infrastructure used in this thesis.

2.3.3. Domain Adaptation Strategies

When adapting models to specialized domains, two primary strategies exist: Continued Pre-Training (CPT) and specific Fine-Tuning.

Continued Pre-Training vs. Fine-Tuning

Continued Pre-Training involves training the model on a large corpus of domain-specific raw text before instruction tuning. While Faroz [13] indicates that CPT can be effective for resource-constrained small models, Chen et al. [8] argue that for complex reasoning tasks, CPT does not uniformly improve performance and may degrade the model's generalization capabilities. Consequently, this thesis focuses on targeted fine-tuning rather than extensive CPT.

Fine-Tuning for RAG (RAFT)

A hybrid approach known as Retrieval Augmented Fine-Tuning (RAFT) [54] specifically optimizes models for the RAG setting. Standard fine-tuning often fails to teach models how to utilize retrieved context effectively, leading to "knowledge conflicts" where the model relies on its internal priors rather than the provided documents.

RAFT addresses this by training the model on questions where the relevant document is mixed with "distractor" documents. This forces the model to learn two capabilities:

1. Identifying the relevant information within a noisy context.
2. Ignoring irrelevant "distractor" segments.

This methodology bridges the gap between the static knowledge of the model and the dynamic retrieval mechanism, ensuring that the stylistic alignment learned during FT does not compromise the factual grounding provided by RAG [52].

2.4. RAG vs. Fine-Tuning (FT)

While both Retrieval-Augmented Generation (RAG) and Fine-Tuning (FT) aim to adapt Large Language Models to specific domains, they operate through fundamentally different mechanisms. Understanding the distinct roles of these technologies is crucial for designing robust enterprise systems, particularly in highly regulated sectors like the energy trade market.

2.4.1. Theoretical Distinction

The distinction can be framed through the analogy of human learning: Fine-Tuning is analogous to a student studying to internalize knowledge for an exam, permanently altering their neural pathways (model weights). In contrast, RAG is analogous to a student taking an open-book exam, where they possess the skill to find and synthesize information from a textbook (external vector store) without necessarily memorizing every detail [52].

- *Fine-Tuning* modifies the model's parametric memory. It is highly effective for adapting the model's linguistic style, learning domain-specific vocabulary (e.g., specific energy market acronyms), and enforcing output formats. However, it suffers from the "black box" problem: once trained, it is difficult to trace which training example led to a specific output, and updating knowledge requires computationally expensive retraining.
- *RAG* leverages non-parametric memory. It excels at accessing dynamic, rapidly changing information. Its primary strength lies in verifiability; every generated claim can be traced back to a specific retrieved document chunk, reducing the likelihood of hallucination for factual queries and allowing the check of the content directly on the source of the information.

2.4.2. Performance on Long-Tail Knowledge

Empirical studies comparing these approaches demonstrate a clear divergence in performance based on knowledge popularity. Research indicates that while FT can effectively encode "popular" knowledge common in the pre-training corpus, it struggles significantly with "long-tail" or proprietary knowledge - information that is rare or private.

For such specific datasets, Fine-Tuning often leads to hallucinations, as the model attempts to fabricate plausible-sounding details to fill gaps in its weight-based memory. Conversely, RAG maintains high accuracy for specialized queries by directly retrieving the relevant context, provided the retrieval mechanism is precise [46].

2.4.3. Synergy and Hybrid Approaches

Recent literature suggests that these methods are not mutually exclusive but complementary. The "RAFT" (Retrieval Augmented Fine-Tuning) methodology proposes that the optimal performance is achieved by fine-tuning a model specifically to be a better RAG reasoner.

In a naive RAG setup, a generic model may struggle to identify relevant information within a noisy context window containing multiple retrieved chunks (distractors). By fine-tuning the model on a dataset where questions are paired with both relevant and irrelevant documents, the model learns to ignore noise and prioritize the retrieved evidence over its internal priors [54].

This hybrid hypothesis forms the basis of the experimental workflow in this thesis, which posits that Fine-Tuning should handle the *form* (legislative style and reasoning patterns), while RAG should handle the *substance* (specific statutes and market data).

2.5. Evaluation methodology

Evaluating the quality of text generated by Large Language Models presents a distinct challenge compared to traditional classification or regression tasks. In the context of the energy trade market, where answers must be both factually grounded in legislation and stylistically precise, the evaluation framework must account for semantic nuance rather than simple lexical overlap. This thesis employs a dual-layered approach, utilizing both a traditional static metric and the emerging state-of-the-art "LLM-as-a-judge" methodology.

2.5.1. Static Metrics and Their Limitations

Historically, Natural Language Generation (NLG) has relied on metrics that measure the overlap between a candidate sequence and a reference (gold) summary.

- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation): Measures the overlap of n-grams (ROUGE-N) or longest common subsequences (ROUGE-L) between the system output and the reference [31].
- **BLEU** (Bilingual Evaluation Understudy): Focuses on precision, calculating the ratio of matching n-grams to the total number of n-grams in the generated output [39].
- **BERTScore**: Computes similarity scores for each token in the candidate sentence with each token in the reference sentence using contextual embeddings from BERT, allowing for soft matches between synonyms [53].

While these metrics provide a quantitative baseline, they are increasingly regarded as insufficient for evaluating long-form reasoning tasks. They punish valid rephrasings that do not share lexical overlap with the reference and fail to capture logical inconsistencies.

For instance, a generated legal interpretation might share 90% of the words with the ground truth but contain a negation that invalidates the answer; static metrics would score this highly despite the factual failure. Consequently, this thesis treats static metrics as secondary indicators of lexical similarity rather than definitive measures of quality.

2.5.2. LLM-as-a-Judge

To address the shortcomings of static metrics, the state-of-the-art approach for semantic evaluation is the "LLM-as-a-judge" paradigm [57]. This methodology utilizes a highly capable model, typically a specialized evaluator, to compare the gold answer (the required one) with the one provided by the target model during the test phase, mimicking human evaluation, but at scale.

Unlike static metrics, this approach allows for the assessment of complex qualitative dimensions such as reasoning coherence, factual grounding, and adherence to specific stylistic instructions. The evaluator model acts as a proxy for human judgment, providing a score and often a rationale for its decision, which offers deeper insight into model performance than a simple numerical score. The final score can be computed as the (weighted) average of the mark given to the different metrics. This allows for better insight on the reasons that lead the model to a positive or negative overall score.

Robustness and Bias Mitigation

While effective, using LLMs as evaluators introduces specific biases that must be managed. Research highlights phenomena such as "LLM Narcissism," where judge models tend to favor outputs generated by themselves or models from the same family [28]. Additionally, "verbosity bias" often leads evaluators to score longer answers higher, regardless of accuracy. To mitigate these biases, best practices in the literature suggest evaluating multiple generation samples or employing pairwise comparisons to filter out stochastic noise and ensure statistical robustness.

2.6. Infrastructure

The computational demands of Large Language Models vary significantly depending on the specific phase of the lifecycle (training, fine-tuning, or inference). This thesis leverages a hybrid infrastructure approach, utilizing both High-Performance Computing (HPC) resources and Cloud Computing services.

This architectural distinction is fundamental to the experimental design. The HPC infras-

tructure is selected to provide the *depth* of control required to alter the model's behavior through full-parameter Fine-Tuning. Conversely, the Cloud infrastructure is utilized for the *flexibility* required to build dynamic RAG systems and orchestrate evaluations without the overhead of managing GPU clusters. By combining these environments, the system optimizes for cost, control, and accessibility.

2.6.1. High-Performance Computing (HPC): Leonardo Cineca

Leonardo, hosted by CINECA, represents the "weight-centric" approach to AI workloads. As one of the pre-exascale EuroHPC systems, it is designed for massive parallel processing. In this environment, the user has direct access to the underlying hardware accelerators (NVIDIA A100 GPUs) and is responsible for the entire software stack.

Operational Mode: Direct Model Execution On Leonardo, using a model implies loading the full set of parameters (weights) into the GPU VRAM. The user must manually manage the environment, including:

- *Data Management*: Due to security policies, execution nodes often lack internet access. Consequently, model weights and datasets must be pre-downloaded and stored in local high-speed storage (e.g., the `$WORK` directory) before execution.
- *Job Scheduling*: Workloads are not interactive. Scripts are submitted via a scheduler (Slurm), where resources are allocated exclusively for the duration of the job.
- *Fine-Tuning Capability*: Because the user holds the model in memory, gradients can be computed and weights updated. This makes HPC the necessary infrastructure for the Fine-Tuning and QLoRA experiments conducted in this research.

2.6.2. Cloud Computing: AWS EC2 and Bedrock

In contrast, the Amazon Web Services (AWS) environment represents the "service-centric" approach. This thesis utilizes Amazon EC2 (Elastic Compute Cloud) primarily as an orchestration layer to interface with managed services like Amazon Bedrock.

Operational Mode: API Invocation In this paradigm, the model is abstracted behind an inference API. The EC2 instance does not load the model weights into its own memory; instead, it acts as a client sending HTTPS requests.

- *Stateless Interaction*: The user sends a prompt (context + query) as a JSON payload to the endpoint and receives a text generation. The internal state of the model is invisible and inaccessible.

- *Retrieval Orchestration*: This infrastructure is ideal for RAG pipelines. The EC2 instance manages the lightweight logic of querying vector databases and formatting prompts, while offloading the heavy compute of generation to the managed service.
- *Limitations*: Since the model weights are hidden behind the API, traditional fine-tuning (which requires back-propagation through specific layers) is not possible in the same manner as on HPC.

3 | Related work

3.1. Legal NLP

The application of Natural Language Processing (NLP) to the legal domain presents unique challenges that distinguish it from general-purpose language tasks. Legal texts are characterized by their extreme length, complex syntactic structures, and a specialized vocabulary where precise definitions are paramount. Unlike general conversation, where ambiguity is often resolved by context, legal language requires rigorous exactness; a single punctuation mark or specific phrasing can alter the legal standing of a contract or statute. Consequently, standard NLP models trained on web corpora often struggle to capture the necessary semantic nuance for legal analysis.

3.1.1. The Discriminative Era: LexGLUE and LegalBERT

Prior to the advent of large generative models, the field focused primarily on discriminative tasks such as document classification, judgment prediction, and named entity recognition. The standard for evaluating these capabilities was established by the **LexGLUE** (Legal General Language Understanding Evaluation) benchmark [5].

Modeled after the general GLUE benchmark, LexGLUE aggregated seven datasets covering diverse tasks, including European Court of Human Rights (ECtHR) case classification and unfair contractual term detection. Research utilizing this benchmark demonstrated that domain-specific pre-training was critical for performance. Models such as LegalBERT [4], which were pre-trained exclusively on legal corpora, consistently outperformed generic baselines (like BERT or RoBERTa) by capturing the distinct distributional properties of legal language. This era established the foundational understanding that legal NLP requires specialized adaptation rather than zero-shot application of generic models.

3.1.2. The Generative Era: RAG and LexRAG

With the rise of Large Language Models (LLMs), the focus shifted from classification to generation - specifically, the ability to act as a legal consultant. This introduced new complexities, as generative models are prone to hallucination, a critical failure in law is unacceptable.

To address this, the industry has moved toward Retrieval-Augmented Generation (RAG) as the standard architecture. The paradigm shift towards retrieval-augmented architectures was formally established by the seminal work of Lewis et al. [29]. In their research, they introduced the RAG framework, demonstrating that combining a pre-trained retriever (Dense Passage Retrieval) with a sequence-to-sequence generator (BART) significantly outperforms purely parametric models on knowledge-intensive tasks such as Open Domain QA. Their work proved that "non-parametric" memory (external documents) could effectively supplement the "parametric" memory (neural weights), allowing models to generate specific answers without necessitating massive retraining.

However, evaluating these systems requires metrics beyond simple accuracy. The recently proposed **LexRAG** benchmark [30] represents the state-of-the-art in this domain. Unlike LexGLUE, which evaluated static predictions, LexRAG assesses a model's ability to handle multi-turn legal consultations. It specifically measures:

- *Citation-Based Grounding*: The ability of the model to not only answer a query but to accurately cite the specific articles or regulations that justify its reasoning.
- *Cognitive Resolution*: The capacity to resolve ambiguous user queries through clarification questions before retrieving the relevant laws.

Current literature suggests that while general-purpose LLMs (like GPT-4) act as strong reasoners, they lack the specific "long-tail" knowledge of local regulations (e.g., Italian energy law), necessitating the hybrid approaches of RAG and Fine-Tuning explored in this thesis.

3.2. RAG vs FT

A central debate in current Large Language Model (LLM) research concerns the optimal method for "injecting" new knowledge into pre-trained models. The literature predominantly categorizes these approaches into two paradigms: parametric adaptation (Fine-Tuning) and non-parametric augmentation (Retrieval-Augmented Generation).

3.2.1. The Knowledge Injection Dilemma

Ovadia et al. [38] conducted a comprehensive benchmark comparing these methodologies across diverse knowledge-intensive tasks. Their findings indicate a fundamental trade-off: while Fine-Tuning (FT) effectively adapts the model's output style and format, it struggles to reliably inject factual knowledge that conflicts with or is absent from the model's pre-training corpus. Specifically, they observed that RAG consistently outperforms unsupervised fine-tuning for both existing knowledge and entirely new knowledge, primarily because it bypasses the model's internal memorization limits.

3.2.2. The "Long-Tail" Knowledge Problem

This distinction becomes critical when dealing with specialized or "long-tail" information - facts that appear infrequently in the general web corpus, such as specific Italian energy trade regulations. Soudani et al. [46] specifically analyzed this phenomenon, demonstrating that Fine-Tuning is effective for "popular" knowledge (widely repeated facts) but degrades significantly for less popular entities. In these "long-tail" scenarios, fine-tuned models frequently hallucinate, fabricating plausible-sounding but incorrect details to fill gaps in their parametric memory. However, the integration of retrieval is not without pitfalls. A critical analysis by Chen et al. [6] on benchmarking Large Language Models in RAG settings reveals a phenomenon often termed "parametric bias". Their extensive experiments highlight that LLMs frequently struggle to prioritize retrieved context over their pre-trained internal knowledge, especially when the two conflict or when the retrieved context is noisy. This tendency to ignore external evidence in favor of parametric memory underscores the necessity for advanced alignment strategies - such as the Fine-Tuning approach explored in this thesis - to force the model to attend more strictly to the provided legal documents. Conversely, RAG maintains high accuracy for these rare queries by grounding the generation in retrieved evidence, provided the retrieval component is robust.

Furthermore, Fine-Tuning introduces the risk of "catastrophic forgetting," where the model loses its general reasoning capabilities or prior knowledge as it overfits to the new, narrow dataset. Research suggests that while FT is superior for teaching a model *how* to speak (style, syntax, domain-specific reasoning), RAG is indispensable for telling the model *what* to speak about (facts, data) [20].

3.3. Domain-Specific Adaptation

Given the complementary strengths of these approaches, recent work focuses on hybrid methodologies that combine Fine-Tuning and RAG to achieve robust domain adaptation.

3.3.1. Hybrid Architectures

The consensus in recent literature is that the state-of-the-art for specialized domains involves a two-step adaptation process:

1. **Instruction Tuning for Reasoner Adaptation**: Fine-tuning the base model not to memorize facts, but to learn the specific reasoning patterns and vocabulary of the specific domain.
2. **Retrieval for Factual Grounding**: Using RAG to supply the dynamic, case-specific information required to answer the query.

A pivotal study in this area is RAFT (Retrieval Augmented Fine-Tuning) by Zhang et al. [54]. The authors argue that standard fine-tuning is suboptimal for RAG systems because it does not teach the model how to distinguish between relevant and irrelevant retrieved documents. They propose a training recipe where the model is fine-tuned on questions paired with a mix of "golden" documents (containing the answer) and "distractor" documents. This method trains the model to actively ignore noise and reason only from the relevant context, significantly improving performance in "open-book" settings compared to standard Instruction Tuning.

3.3.2. Joint Training Approaches

Beyond pipeline approaches, research into End-to-End RAG attempts to train the retriever and generator components simultaneously. Approaches like RAG-Studio [51] suggest that optimizing the embedding space (retriever) alongside the generation space (LLM) can align the two components more effectively than training them in isolation. However, these methods require substantial computational resources and massive datasets, often making the separate "Hybrid RAG + PEFT" approach more practical for industrial applications with limited infrastructure.

4 | Main idea of the thesis

4.1. Ablation study

The ablation study represents a core methodological pillar in the evaluation of deep learning systems. It involves the systematic removal or isolation of specific components to quantify their individual contribution to the overall performance [44]. In the context of this thesis, this approach is essential to disentangle the effects of Retrieval-Augmented Generation (RAG) and Fine-Tuning (FT) when applied to the Italian energy regulation domain.

4.1.1. Methodological Rationale

Adopting an ablation framework allows for a transparent comparison of different architectural configurations. This is considered the state-of-the-art approach for validating complex Large Language Model (LLM) pipelines for several reasons:

- **Component Contribution:** It identifies whether a performance gain is truly attributable to a specific technique or if it stems from the baseline model's inherent capabilities.
- **Synergy and Interference:** It tests the hypothesis that adding more components does not always lead to linear improvements. In some cases, combining techniques can lead to "task interference" or increased hallucination rates if the components are not properly aligned [34].
- **Optimization:** By testing all possible combinations (Baseline, RAG-only, FT-only, and RAG+FT), it aims to identify the most efficient configuration, avoiding unnecessary computational overhead in production environments.

4.1.2. Functional Separation of RAG and Fine-Tuning

A key premise of this study is that RAG and Fine-Tuning address different limitations of LLMs. Fine-Tuning modifies the parametric memory of the model, allowing it to inter-

nalize specific linguistic styles, professional words, and output formats [21]. Conversely, RAG provides access to non-parametric external knowledge, ensuring that the model remains grounded in factual, up-to-date regulatory documents without requiring constant retraining [29].

Since these techniques operate on different levels - stylistic alignment versus factual grounding - they are theoretically complementary. However, only a rigorous ablation study can determine if the fine-tuned model retains the ability to effectively attend to external context provided by the RAG system or if the training process has rendered the model too rigid.

4.1.3. Experimental Configurations

To provide a comprehensive evaluation, four distinct stages are analyzed:

1. **Baseline:** Evaluation of the raw open-source models to establish a performance floor. Both techniques should improve performance compared to this specific case.
2. **RAG:** Assessing the impact of external context from verified and truthful documents on factual accuracy.
3. **Fine-Tuning (FT):** Measuring the improvement in adherence to the Italian legal-style formatting by changing the model's weights directly.
4. **Hybrid (RAG + FT):** Testing the convergence of stylistic mastery and factual retrieval.

4.2. Methodology overview

The experimental design of this thesis is structured to systematically isolate the contribution of external knowledge retrieval versus internal parameter adaptation. The workflow proceeds through four distinct phases:

1. **Baseline Assessment:** Evaluating the raw capabilities of the models to establish a performance floor.
2. **RAG Integration:** Measuring the impact of injected context on factual accuracy.
3. **Fine-Tuning (FT):** Assessing the model's ability to learn the specific Italian legal style and output format.
4. **Hybrid Deployment (RAG + FT):** Testing the synergy between the two approaches to verify if stylistic adaptation compromises retrieval attention.

4.2.1. Model Selection Criteria

The selection of the Large Language Models (LLMs) was governed by the strict constraints of the hybrid infrastructure described in Section 2.6. To ensure a valid comparison across all experimental stages, the selected models had to satisfy two mandatory requirements:

1. **Open-Weights Availability:** The models must be open-source (or open-weights) to allow for full parameter Fine-Tuning on the Leonardo Cineca HPC infrastructure.
2. **Managed API Integration:** The models must be available via Amazon Bedrock to facilitate scalable and stateless RAG inference without the need for self-hosted retrieval pipelines.

From the intersection of these two availability sets, a diverse cohort of 10 models was selected. The selection strategy aimed to maximize heterogeneity across three key dimensions: *size* (parameter count), *origin* (provider/training data bias), and *specialization*, to analyze how distinct pre-training architectures respond to RAG and Fine-Tuning..

Parameter Scale and Diversity

The models span three orders of magnitude, ranging from efficient "edge" models (3B parameters) to massive "reasoning" models (120B parameters). This diversity allows the study to analyze whether the benefits of RAG and Fine-Tuning scale linearly with model size or if smaller models exhibit disproportionate gains.

- **Small Models ($<$ 20B):** Selected to test the feasibility of deploying legal assistants on resource-constrained hardware.
- **Medium Models (20B $-$ 40B):** Representing the trade-off between reasoning capability and computational cost.
- **Large Models ($>$ 40B):** Included to establish the performance ceiling for open-source technology.

Table 4.1 lists the final pool of models selected for the ablation study. To ensure a global perspective and mitigate training data bias, the pool includes models from US-based providers (Google, Meta, OpenAI, NVIDIA), European labs (Mistral AI), and Asian providers (Qwen, Minimax).

Table 4.1: The cohort of 10 open-weight models selected for the experiments, categorized by size. The selection includes diverse providers to test robustness across different pre-training corpora.

Category	Model Family	Provider	Parameters
Large	GPT-OSS	OpenAI	120B
	Qwen 3 Next	Qwen	80B
Medium	Qwen 3	Qwen	32B
	Gemma 3	Google	27B
	GPT-OSS	OpenAI	20B
Small	Ministral 3	Mistral AI	14B
	Nemotron	NVIDIA	12B
	Minimax	Minimax	10B
	Gemma 3	Google	4B
	Llama 3.2	Meta	3B

4.2.2. Control and Evaluation Models

In addition to the open-weights models used for the ablation study, two proprietary models were selected to serve as the control group and the evaluation engine, consistent with the "LLM-as-a-judge" methodology:

- **Benchmark (Gold Standard):** *Google Gemini 3 Pro* was used to generate the reference answers for the synthetic dataset, Being released in late 2025, it is recognized as one of the most capable proprietary models currently available on the market, particularly for complex reasoning and agentic workflows [10].
- **Evaluator (Judge):** *Claude 4.5 Opus* (Anthropic) was selected as the judge model due to its high correlation with human preference in complex reasoning tasks. Furthermore, being the biggest model available on Bedrock, ensures reliable scoring of the generated legal explanations.

4.2.3. Embedding Model: Amazon Titan v2

To facilitate the retrieval component of the RAG pipeline, the system utilizes *Amazon Titan Text Embeddings v2* (`amazon.titan-embed-text-v2:0`). This model was selected as the vectorization engine for three critical reasons aligned with the project's infrastructure and domain requirements:

- **Multilingual Capabilities:** Unlike many open-source embedding models opti-

mized primarily for English, Titan v2 is trained on a diverse corpus spanning over 100 languages. This ensures robust semantic representation of Italian legal texts, capturing the nuances of civil law terminology better than generic models [42].

- **Extended Context Window:** The model supports an input context length of up to 8,192 tokens. This feature is particularly advantageous for legal document processing, as it allows for larger chunk sizes (e.g., entire articles or regulatory subsections) to be embedded without truncation, preserving local semantic context.
- **Storage Efficiency:** Titan v2 introduces support for variable output dimensions (256, 512, or 1024). For this thesis, the embeddings were generated at the full 1024-dimension resolution to maximize retrieval accuracy, while maintaining the option to downscale for production environments to reduce vector database costs.

The integration is fully managed via Amazon Bedrock, ensuring low-latency inference and seamless compatibility with the AWS-based vector store implementation.

4.3. Dataset creation

Data quality acts as the fundamental bottleneck in the performance of Large Language Models. As established in the literature, the "Garbage In, Garbage Out" principle is particularly acute in generative tasks, where the model's output quality is strictly bounded by the semantic density and factual accuracy of the training or retrieval data [58]. Consequently, the construction of the dataset represents the most critical phase of this thesis, and the limitations inherent in this process must be acknowledged when interpreting the final experimental results.

4.3.1. Data Provenance Strategy: Ideal vs. Real

In an ideal industrial scenario, the ground truth data would be supplied directly by the end-users (domain experts in energy law) to ensure perfect alignment with real-world needs. However, the constraints of this project - specifically the requirement for a massive volume of complex legal Q&A pairs and the unavailability of extensive human annotation resources - necessitated a synthetic data generation approach.

To mitigate the risks associated with synthetic data, a *Model-Based Data Generation* pipeline was implemented. *Google Gemini 3 Pro* was selected as the "Oracle" or "Source of Truth" model due to the reasons explained in Section 4.2.2. Its high reasoning capabilities allows it to function as a proxy for human labeling, generating "Gold Standard" answers that serve as the benchmark for evaluating the smaller, open-source models. Functionally,

its behavior mirrors that of a domain stakeholder in a real-world scenario, providing the developer with precise expected outputs. Throughout this thesis, the content generated by *Gemini* is established as the absolute ground truth for evaluation purposes.

4.3.2. Generation Methodology and Prompt Engineering

The core dataset was constructed using a *Model-Based Data Generation* pipeline, leveraging *Google Gemini 3 Pro* as the "Oracle". The process involved ingesting raw regulatory documents provided by business stakeholders and instructing the model to synthesize high-quality Question-Answer (Q&A) pairs in a conversational format.

To ensure consistency across the dataset, a fixed *System Prompt* was defined and maintained throughout the generation process. This prompt was engineered to impose a specific persona: a "Senior Legal Expert for Edison". This choice was strategic: it had to be specific enough to enforce the requirement for operational, business-oriented answers (including numerical examples and practical cases), yet broad enough to allow the model the freedom to traverse its internal knowledge base without over-constraining the reasoning process.

The resulting data structure follows the standard chat template required for fine-tuning. This format, illustrated in Figure 4.1, organizes the interaction into three distinct roles: System, User, and Assistant.

```
{
  "messages": [
    {
      "role": "system",
      "content": "The system prompt for the model."
    },
    {
      "role": "user",
      "content": "The question on which test the model."
    },
    {
      "role": "assistant",
      "content": "The gold, expected answer, simulating the answer of the stakeholder."
    }
  ]
}
```

Figure 4.1: Structure of the JSON object used for Fine-Tuning and Evaluation.

The content within the JSON fields was rigorously curated to maximize the training efficiency:

- **System Prompt:** This field is invariant across the dataset. It instructs the model to adopt a formal tone, strictly cite regulatory articles and avoid generic conversational fillers. The persona is designed to replicate the behavior of a senior consultant who grounds every statement in the provided legal context. It asks also to provide examples inside the specific domain of the company (Edison).
- **User Query:** The questions were diversified to cover the full spectrum of potential interactions expected in a production environment. To give an idea of the different types of questions, they might ask for explicit definitions (precise legal interpretations of specific terms), normative comparisons (complex queries requiring the model to highlight differences between regulatory updates or document summaries (synthesize lengthy texts into executive summaries).
- **Assistant Response (Gold Standard):** This represents the ground truth. Since it is generated by the "Oracle" (Gemini) based on the source documents, it effectively simulates the ideal answer a domain stakeholder would provide. In the Fine-Tuning phase, this field serves as the target variable.

4.3.3. Strategic Refinement

The composition of the training and testing sets underwent a critical evolution during the experimental phase, driven by preliminary observations on model behavior.

The first approach can be described as the **Big Data Approach**. Initially, the experimental design followed a traditional deep learning approach, prioritizing data volume.

- **Setup:** 500 samples were generated for testing, while a massive dataset of 13,000 samples was created for Fine-Tuning.
- **Source:** The training set was derived mostly from general Civil Law documents and generated using a different model (GPT-5.2) to diversify the input, with the aim of teaching the model the style, not the content.
- **Outcome:** Preliminary fine-tuning runs revealed a degradation in performance. The model suffered from *style mismatch*: the stylistic patterns learned from the Civil Law dataset (generated by GPT) conflicted with the Energy Law reasoning required in the test phase (generated by Gemini). Instead of adapting to the domain, the model's responses became incoherent due to the distribution shift between training and testing data.

Therefore this approach was discarded.

After some considerations about the reasons of the failure, a new approach was considered, based on the **Source consistency** between the training and the testing data. The strategy was pivoted towards data homogeneity and domain specificity, adhering to the *Less Is More* principle [58].

- **Homogenization:** The dataset size was reduced to 1,000 high-quality samples, all generated exclusively by *Gemini* to ensure perfect stylistic consistency between training and inference.
- **Domain Alignment:** All samples were strictly derived from the Energy Legal field, abandoning generic Civil Law topics.
- **Result:** At this stage, a unified pool of around a thousand samples was created by ingesting a mixed corpus of documents. This included both the texts intended for the RAG Knowledge Base and external held-out documents. This resulted in a raw, aggregated dataset containing questions covering the entire spectrum of available regulatory material, ready for the subsequent cleaning and partitioning phases.

4.4. Evaluation framework

Evaluating generative language models presents a significantly more complex challenge than traditional machine learning tasks. Unlike regression or classification, where outputs can be directly compared against a ground truth using deterministic metrics (e.g., MSE, Accuracy, F1-Score), the output of an LLM is open-ended natural language. Two answers can be lexically disjoint yet semantically equivalent, rendering standard n-gram metrics like BLEU or ROUGE often insufficient for assessing reasoning and factual nuances in the legal domain.

While human evaluation remains the gold standard, it is unfeasible for large-scale ablation studies due to the prohibitive time and cost required to grade thousands of responses. Following recent advancements in automated evaluation [57], this thesis adopts a state-of-the-art *LLM-as-a-Judge* approach, supplemented by a deterministic static metric. This hybrid framework ensures a balanced assessment, capturing both the semantic quality and the lexical precision of the generated answers.

4.4.1. Hybrid Scoring Methodology

The final quality score (S_{final}) for each inference is a weighted linear combination of two distinct components: a semantic evaluation performed by the Judge Model (S_{judge}) and a lexical evaluation based on Entity Recall (S_{static}).

The aggregation formula is defined as:

$$S_{final} = \alpha \cdot S_{judge} + (1 - \alpha) \cdot S_{static} \tag{4.1}$$

where $\alpha = 0.8$. This weighting assigns 80% of the importance to the semantic capabilities of the Judge, while retaining a 20% anchor on objective keyword retrieval to penalize hallucinated entities or omitted numerical figures.

4.4.2. Static Metric: Key Entity Recall

The static metric serves as an objective guardrail, independent of the judge's potential biases. It is designed to verify that the key subjects, proper names, and numerical quantities present in the Gold Standard answer are preserved in the Generated output. To assess the factual completeness of the generated answers, a static metric focused on content coverage was employed, inspired by the Entity-level Factual Consistency metrics proposed by Nan et al. [37].

Unlike standard n-gram metrics like ROUGE [31], which treat all tokens equally regardless of their semantic weight, this methodology isolates specific "information-carrying" tokens. Specifically, the algorithm utilizes Part-of-Speech (POS) tagging to extract a set of key entities E_{gold} from the ground truth. This set comprises exclusively *Nouns*, *Proper Nouns*, and *Numbers*.

The decision to exclude verbs, adjectives, and functional words is driven by specific linguistic constraints of the Italian language:

- **Morphological Invariance:** Italian verbs and adjectives exhibit high morphological inflection based on tense, mood, gender, and number; by focusing on nouns and entities, the metric avoids false negatives caused by conjugation or declension mismatches.
- **Information Density:** Functional words such as articles or prepositions possess low information entropy. Their inclusion would artificially inflate the similarity score without guaranteeing that the core factual content has been preserved.

The metric calculates the recall of the ground truth entities found within the generated text (E_{gen}). To ensure robustness, all tokens are normalized to lowercase prior to matching. The final score is computed as follows:

$$S_{static} = 10 \cdot \frac{|E_{gold} \cap E_{gen}|}{|E_{gold}|} \tag{4.2}$$

where $|E_{gold}|$ represents the cardinality of the entity set in the reference answer, and $|E_{gold} \cap E_{gen}|$ represents the count of correctly retrieved entities. The result is scaled by a factor of 10 to align the output range $[0, 10]$ with the scoring system of the Judge model.

This approach acts as a strict factual guardrail: it penalizes the omission of critical details (such as specific law article numbers, dates, or proper names) while remaining robust to the syntactic variations inherent in generative models.

4.4.3. LLM-as-a-Judge Evaluation

The core semantic evaluation is performed by *Claude 4.5 Opus*, as described in Subsection 4.2.2, acting as an impartial expert judge. Unlike the static metric, which focuses on lexical precision, the Judge is tasked with assessing the reasoning quality, the adherence to the persona, and the stability of the model.

To provide a holistic assessment, the Judge evaluates the system on a single composite spectrum consisting of *seven distinct dimensions*. For each dimension, the model assigns a score on a scale from 1 to 10, where a score of 6 is calibrated as the "Sufficient/Pass" threshold.

Seven criteria contribute to the final Judge Score (S_{judge}):

1. **Accuracy:** Verifies the factual correctness of the generated content, specifically checking numerical figures, dates, and the interpretation of regulatory articles against the Gold Standard.
2. **Completeness:** Assesses whether the answer covers all key clauses, exceptions, and nuances present in the ground truth, penalizing unjustified truncation or omission of relevant caveats.
3. **Answer Relevance:** Measures how directly and specifically the response addresses the user's query, penalizing generic or evasive answers.
4. **Style Adherence:** Evaluates the tone and register, ensuring the output matches the professional required by the persona (e.g., formal vocabulary, absence of conversational fillers).
5. **Citation Quality:** Checks for the presence and correctness of explicit references (e.g., "Pursuant to Art. 2 of Regulation REMIT"), which are mandatory for auditability in the energy sector.
6. **Contextual Grounding (Edison):** Assesses the application of abstract legal concepts to the specific industrial context of Edison, verifying the plausibility of the

generated operational examples. The reason is that the stakeholders often appreciate a practical example in a field they are expert about.

7. **Robustness and Stability:** Unlike the previous criteria which evaluate the quality of a single output, this metric measures the stochastic stability of the model. For every query, the system generates $N = 5$ independent responses; the Judge analyzes this set to determine semantic consistency. A score of 10 indicates perfect consistency, while lower scores indicate contradictions or hallucinations across runs.

Score Aggregation

The final judge score is calculated as the unweighted arithmetic mean of these seven dimensions. This approach places equal importance on the correctness of the single answer (Accuracy, Completeness) and the reliability of the generation process (Robustness).

$$S_{judge} = \frac{1}{7} \sum_{i=1}^{7} C_i \tag{4.3}$$

where $C_{1...6}$ represent the quality metrics and C_7 represents the robustness score. In addition to the numerical value, the Judge is instructed to generate a brief reasoning string (in Italian) to justify the assigned scores, facilitating qualitative error analysis.

5 | Implementation experience

5.1. System Architecture

The experimental architecture of this thesis is designed to address the dichotomous requirements of the ablation study: the flexibility required for rapid retrieval orchestration and the computational density required for full-parameter model adaptation. Consequently, the infrastructure is split into two distinct environments: a Cloud Computing environment hosted on Amazon Web Services (AWS) and a High-Performance Computing (HPC) environment hosted on the Leonardo Cineca cluster.

This hybrid approach ensures that each stage of the experiment utilizes the hardware best suited for its specific operational constraints. The execution of the four experimental steps (defined in Section 4.2) is distributed as follows:

5.1.1. Cloud Environment: AWS EC2 and Bedrock

The AWS environment serves as the orchestration hub for the project. It is centered around an Amazon EC2 instance (*region: eu-west-1*), which acts as the primary controller for data preparation, retrieval logic, and evaluation.

- **Inference for Steps 1 and 2 (Baseline and RAG):** The inference for the unmodified open-source models (Step 1) and the standard RAG implementation [29] (Step 2) is executed entirely within this environment. The EC2 instance interacts with the models via the Amazon Bedrock API. This stateless interaction model is optimal for these stages, as it allows for the rapid swapping of models without the overhead of managing local GPU memory.
- **Centralized Evaluation:** A critical requirement for the validity of the ablation study is the consistency of the evaluation metric. As noted by Zheng et al. [57], using a consistent judge model is essential to mitigate evaluation bias. To ensure that all models are judged under identical conditions, the evaluation phase is centralized on the EC2 instance. Regardless of where the inference is generated (Cloud or HPC),

all response files are aggregated here. The Judge model (Claude 4.5 Opus) is invoked via the Bedrock API using a fixed prompt template, ensuring that the scoring logic remains invariant across all experimental configurations.

5.1.2. HPC Environment: Leonardo Cineca

The training and execution of adapted models are performed on the Leonardo Cineca infrastructure. Access to this pre-exascale cluster is managed through the dedicated user account `AIFAC_F01_260`.

- **Step 3 (Fine-Tuning):** The Fine-Tuning process requires direct access to the model's weight matrices for back-propagation. This workload is executed on Leonardo's compute nodes, utilizing NVIDIA A100 GPUs to perform the QLoRA adaptation [11] described in Section 2.3.2.
- **Step 4 (Hybrid RAG + FT):** Unlike the baseline models, the fine-tuned models cannot be deployed back to Amazon Bedrock. Consequently, the inference for the Hybrid stage (Step 4) is performed locally on the Leonardo nodes. The system loads the adapted weights into VRAM and executes the generation.

To bridge these two environments, a synchronization pipeline was established. Since the compute nodes on Leonardo operate in an isolated environment without direct internet access, data transfer must be routed strictly through the login nodes using the Secure Copy Protocol (SCP). The datasets and retrieval contexts are prepared on AWS and transferred to the Leonardo `$WORK` directory. Conversely, the inference results generated by the fine-tuned models on Leonardo are exported back to the AWS EC2 instance for the centralized evaluation, ensuring a unified analysis of the results. The following command illustrates the retrieval of the JSONL results for the Gemma model from the remote Cineca infrastructure:

```
scp fpelleg2@login.leonardo.cineca.it:/leonardo_work/AIFAC_F01_260/\
inference_results/Gemma-27B_ft.jsonl \
"tesi/inference/finetune/gemma_27b.jsonl"
```

5.2. Data Pipeline

The implementation of the hybrid architecture required two distinct data processing pipelines: one for constructing the non-parametric knowledge base required by the RAG system, and one for refining the instructional dataset required for Fine-Tuning.

5.2.1. Knowledge Base Construction (RAG Pipeline)

The reliability of the Retrieval-Augmented Generation system is strictly dependent on the quality of the segmented text fed into the vector database. Standard heuristic splitting (e.g., fixed character count) often fails with legal documents due to the presence of pervasive non-semantic elements (headers, footers, page numbers) and the risk of fragmenting logical clauses [36].

To address this, a custom two-stage "Smart Chunking" pipeline was implemented, leveraging the reasoning capabilities of Large Language Models for both cleaning and segmentation.

1. **Visual Cleaning via VLM (OCR Correction):** The first stage focuses on denoising the source PDF. Instead of standard text extraction, each page is processed as an image-text pair by a fast Vision-Language Model (Claude 3 Haiku). Recent benchmarks on multi-modal retrieval demonstrate that visual-driven extractors significantly outperform text-only parsers in preserving document structure [55]. A specific system prompt instructs the model to:

    - Transcribe the exact text while explicitly ignoring headers, footers, and page numbers.
    - Merge hyphenated words at line breaks (e.g., "re- gulation" → "regulation") to restore semantic integrity.
    - Output a clean, continuous text stream free of conversational fillers.

    This approach ensures that the input to the chunker is free of the artifact noise that typically degrades retrieval performance.

2. **LLM-Based Semantic Chunking:** The cleaned text is processed by a "Semantic Chunker" agent. A rolling buffer of approximately 12,000 characters is fed to the model, which is tasked with identifying logical break points in the legal text. Unlike fixed-size splitting, this agentic approach ensures that chunks align with semantic boundaries (e.g., avoiding splits in the middle of a sentence). This methodology aligns with the *ChunkRAG* framework proposed by Reuven et al. [40], which validates the use of LLMs to filter and segment information for higher retrieval precision.

    The prompt given the model enforces a strict structure:

    - **Contextual Headers:** Each chunk is enriched with metadata (Document Name, Section Title) to ensure the embedding model captures the global context even in short segments. This technique, known as *Contextual Retrieval*,

has been shown to reduce retrieval failure rates by providing the necessary disambiguation for isolated chunks [1].

- **Logical Integrity:** The model is instructed to maintain the unity of definitions or articles.
- **Remainder Management:** Text that does not form a complete chunk at the end of the buffer is returned as a remainder and prepended to the next batch, ensuring zero information loss between processing windows.

3. **Vectorization and Indexing:** The resulting semantic chunks are encoded into 1024-dimensional vectors using the Amazon Titan Text Embeddings v2 model. To ensure compatibility with the FAISS vector store [24], a critical normalization step is applied.

   Standard semantic retrieval relies on Cosine Similarity to measure the angle between the query vector $\mathbf{q}$ and a document vector $\mathbf{d}$. The formula is defined as:

$$\text{CosineSimilarity}(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|} \tag{5.1}$$

   where $\mathbf{q} \cdot \mathbf{d}$ is the dot product (Inner Product) and $\|\cdot\|$ represents the Euclidean norm (L_2).

   However, calculating the norms during retrieval is computationally expensive at scale. To optimize this, the system enforces strict L_2 normalization on all vectors before indexing, ensuring that $\|\mathbf{q}\| = 1$ and $\|\mathbf{d}\| = 1$. Under this constraint, the denominator becomes 1, and the similarity metric simplifies to a pure Inner Product:

$$\text{CosineSimilarity}(\mathbf{q}_{norm}, \mathbf{d}_{norm}) = \mathbf{q}_{norm} \cdot \mathbf{d}_{norm} \tag{5.2}$$

   This mathematical property allows the use of the `IndexFlatIP` (Inner Product) index in FAISS, which is significantly faster than distance-based indices while preserving the exact semantic ranking of Cosine Similarity.

5.2.2. Fine-Tuning Dataset Refinement (FT Pipeline)

While Section 4.3 described the generative origin of the synthetic data, the raw output from the Oracle model was not immediately suitable for training. To ensure high-quality Fine-Tuning, a rigorous post-processing pipeline was implemented. This phase focused on three objectives: classifying the provenance of the information, filtering out samples

with suboptimal stylistic alignment, and creating a sorted, statistically robust dataset.

Grounding Classification

The first preprocessing step involved a binary classification of the 1,008 raw Q&A pairs based on their information source. A metadata field `is_grounded` was assigned to each sample:

- **True (Grounded):** The answer-question derived from documents that are explicitly present in the Knowledge Base (KB) used by the RAG system during the inference phase.
- **False (Un-grounded):** The answer-question derived from documents—regulations that were used to generate the training data but were deliberately excluded from the RAG vector store.

This classification is crucial for the results analysis, as it allows for a granular assessment of the *Knowledge Injection* capabilities of the models. It enables the distinction between the model's ability to retrieve accessible information versus its ability to internalize knowledge solely through Fine-Tuning.

Automated Quality Filtering (LLM-as-a-Filter)

To maximize the efficiency of the Fine-Tuning process, this thesis adopts the *AlpaGasus* methodology proposed by Chen et al. [7]. Their research demonstrates that training on fewer, high-quality samples filtered by a superior model yields better results than training on a larger, noisier dataset.

Consistent with this approach, the same model selected for the final evaluation (Claude 4.5 Opus) was used as a *Data Quality Expert*. The model evaluated every (Question, Answer) pair, assigning a scalar score $S \in [0, 10]$ based on a strict prompt designed to enforce the *Edison Consultant* persona. The evaluation criteria were the following:

1. **Style and Tone:** Penalization of philosophical or abstract language; reward for practical, operational phrasing (e.g., "Pursuant to Art. X...").
2. **Structure:** Penalization of simple dictionary definitions; reward for complex operational guidelines.
3. **Completeness:** Assessment of formatting and exhaustive coverage of the query.

Statistical Pruning

To determine the cutoff threshold for data inclusion, a statistical analysis of the score distribution was conducted using the Interquartile Range (IQR) method, as originally defined by Tukey [49]. This robust statistical dispersion measure was selected over standard deviation filters because it is less sensitive to extreme values, which is critical when dealing with the non-Gaussian distributions often observed in LLM evaluations.

The analysis of the 1,008 samples yielded the following distribution metrics:

- Q_1 (25^{th} percentile): 6.87
- Q_3 (75^{th} percentile): 7.68
- $IQR = Q_3 - Q_1 = 0.81$

A crucial modification to the standard outlier detection algorithm was applied. In the context of data quality, positive outliers, scores significantly higher than the median, represent high-value training signals (*Super-Samples*), whereas negative outliers represent hallucinations or formatting failures. Consequently, an *asymmetric filter* was implemented. The Upper Bound was ignored to preserve the best samples, while a strict Lower Bound (LB) was enforced to prune the noise.

Using a conservative multiplier of $k = 1.0$ (stricter than the standard 1.5 used for normal distributions) [49], the cutoff was calculated as:

$$LB = Q_1 - (1.0 \times IQR) = 6.87 - 0.81 = 6.06 \tag{5.3}$$

All samples with a score $S < 6.06$ were discarded. This process removed 54 low-quality examples, resulting in a refined dataset of 954 high-quality samples.

Indexing and Splitting

Finally, the surviving samples were sorted in descending order of quality score. A unique index (ID) was assigned to each sample, where $ID = 0$ represents the highest-quality example according to the evaluator. This indexing strategy ensures reproducibility and allows for curriculum learning experiments (training on easy/high-quality data first) in future developments.

The final partitioning of the dataset employed a **stratified random sampling** strategy to preserve the distributional properties of the quality scores across all subsets. Simple random splitting poses the risk of introducing distribution shifts, where the testset might

inadvertently contain easier (higher rated) or harder (lower rated) examples than the training set, biasing the evaluation. To mitigate this, the continuous quality score S was discretized into five quantiles (quintiles) using the pandas `qcut` function. These bins served as the stratification targets.

The dataset was split into Training (70%), Validation (15%), and Test (15%) sets. This specific ratio was selected following the heuristic guidelines for moderate-sized datasets ($100 < N < 10,000$) outlined by Géron [16]. Given the dataset size ($N = 954$), assigning 15% to the test set yields approximately 140 samples, which is sufficient to ensure that the evaluation metrics are statistically significant while maximizing the data available for the Fine-Tuning phase. As demonstrated by Kohavi [27], combining this ratio with stratified sampling significantly reduces the variance of accuracy estimation compared to pure random splitting, ensuring that the reported metrics reflect the model's true generalization capability rather than artifacts of a specific data split.

5.3. Model Inference Execution

This section details the operational protocols adopted to execute the inference across the hybrid infrastructure. To ensure scientific rigor and reproducibility across the heterogeneous infrastructure (AWS Cloud vs. Leonardo HPC), a deterministic retrieval pipeline and a strict standardization of Input/Output schemas were established prior to the definition of specific model parameters.

Regardless of the inference engine, all experiments adhered to strict schemas to guarantee the compatibility of the results. While the core interaction always follows the chat template structure defined in Figure 4.1, the encapsulating JSON format varies depending on the experimental phase.

Regardless of the computing environment (AWS Cloud or Leonardo HPC), all models interact with data through normalized JSON schemas. This unification is essential to feed the centralized evaluation pipeline described in Section 4.4.

5.3.1. Retrieval Logic

For the experimental stages involving Retrieval-Augmented Generation (RAG) and Hybrid architectures, a *Frozen Context* strategy was implemented. Instead of performing real-time retrieval during inference, the context retrieval was performed *offline* prior to generation. This ensures a deterministic comparison because every model receives exactly the same set of chunks for a given question, isolating the reasoning capability of the

generator from potential stochastic fluctuations in the retriever.

The retrieval pipeline employs a two-stage logic:

1. **Query Rewriting:** User queries are processed by a fast intermediate agent (Claude 3 Haiku) to disambiguate acronyms and inject domain-specific keywords (e.g., expanding acronyms). This step is crucial to standardize the input format, as demonstrated by Gao et al. in the HyDE framework [14], leveraging an LLM to hallucinate or rewrite a query into a canonical form significantly improves dense retrieval performance by reducing the semantic mismatch between short user questions and formal document text. In addition it decreases the dependency on the user's prompting style and ensuring the query is semantically aligned with the target document language (Italian).
2. **Hybrid Search:** The system executes a parallel search to capture both semantic concepts and exact technical references:
   - *Dense Retrieval:* The expanded query is embedded using Amazon Titan v2 and queried against the FAISS index to retrieve the top-50 semantic candidates.
   - *Sparse Retrieval:* Simultaneously, the BM25Okapi algorithm scans the corpus to retrieve the top-50 candidates based on exact keyword matching, ensuring that specific regulation numbers or rare acronyms are not lost in vector compression.

   This hybrid approach is supported by the BEIR benchmark findings [47], which indicate that while dense models capture semantic nuance, exact keyword matching (BM25) remains superior for domain-specific acronyms.

The two candidate lists are merged using **Reciprocal Rank Fusion (RRF)** [9] with a smoothing constant $k = 60$. This method prioritizes documents that appear in both rankings, filtering out outliers. From the fused list, the system selects the top-15 unique chunks.

Finally, a **Context Window Expansion** logic is applied: for every selected chunk, the system retrieves the immediately preceding and succeeding text segments ($window = 1$) from the document store. This restores the narrative flow often broken during the chunking phase, providing the LLM with a coherent context rather than isolated fragments.

5.3.2. Input Data Structure

The input datasets are serialized in JSONL format. While the core message content remains consistent across all stages to ensure comparable evaluation, the schema varies slightly depending on the experimental track to accommodate the specific requirements of training versus retrieval:

- **Baseline and Fine-Tuning:** These tracks use a simplified schema focusing solely on the interaction history required by the training libraries (e.g., HuggingFace TRL).

```
{
  "index": <integer>,
  "messages": [
    {"role": "system", "content": "System prompt defining the persona."},
    {"role": "user", "content": "Training input (Question)."},
    {"role": "assistant", "content": "Target output (Gold Answer)."}
  ],
  "score": <float>
}
```

- **RAG and Hybrid:** These tracks utilize an advanced schema that includes the pre-fetched knowledge. This structure allows the inference engine to inject the `retrieved_context` dynamically into the system prompt at runtime without re-triggering the retrieval search.

```
{
  "id": <integer>,
  "original_question": "...",
  "gold_answer": "...",
  "score": <float>,
  "is_grounded": <boolean>,
  "new_query": "Query rewritten by the agent",
  "retrieved_context": [
      "Chunk 1 content...",
      "Chunk 2 content..."
  ]
}
```

- **Fields Description:**
  - `retrieved_context`: The top-20 semantic chunks retrieved from the Vector Store.

- `is_grounded`: A boolean flag indicating if the answer exists in the Knowledge Base (True) or if it refers to held-out documents (False).
- `new_query`: The query re-written by the retrieval agent to optimize vector similarity search.

5.3.3. Output Data Structure

To facilitate the centralized evaluation, all inference scripts produce a unified output format. The output schema is designed as a superset of the input schema: it preserves all original metadata (including the retrieved context for RAG models) and appends the generation results.

This preservation strategy is critical for granular error analysis. By combining together the `retrieved_context` with the `generated_responses`, it is possible to distinguish between cases where the model failed due to a lack of information (empty or irrelevant context) versus cases where the model failed to reason over correctly retrieved facts.

```
{
  # Inherits all fields from the Input object (id, gold_answer, context...)

  "generated_responses": [
      "Generation 1...",
      "Generation 2...",
      ...
      "Generation 5..."
  ],
  "model_profile": "<model_name_version>"
}
```

5.3.4. Cloud Implementation (AWS Bedrock)

For the Baseline and RAG operational steps, the inference orchestration relies on the Amazon Bedrock managed service. This architecture was selected to decouple the generation logic from the underlying hardware management, allowing for a stateless and highly parallelizable execution flow.

Global Inference Configuration

To guarantee the scientific validity of the ablation study, a strict *Ceteris Paribus* (all other things being equal) condition was enforced. A single global configuration object, defined

as `INFERENCE_CONFIG`, was applied immutably across all 10 models and all experimental runs.

The hyperparameters were tuned to achieve a specific balance between determinism and fluency:

- **Temperature ($T = 0.5$):** This value represents a strategic trade-off. The higher, the more freedom is given to the model. While a temperature of 0.0 is typically preferred for extraction tasks, legal explanation requires a degree of linguistic flexibility to construct cohesive arguments. A value of 0.5 injects sufficient randomness to avoid repetitive phrasing while maintaining a strong adherence to the most probable (factual) tokens, minimizing the risk of hallucinations common at higher temperatures (e.g., $T > 0.7$).
- **Top-P ($P = 0.9$):** Also known as Nucleus Sampling, this parameter truncates the long tail of low-probability tokens. Setting $P = 0.9$ ensures that the model considers the top 90% of the probability mass, effectively filtering out nonsensical or grammatically incorrect tokens while preserving a rich enough vocabulary to handle complex legal terminology.
- **Max Tokens (1024):** The output limit was capped at 1024 tokens. This window is sufficiently large to accommodate detailed, multi-paragraph legal opinions but acts as a hard stop to prevent failure modes where the model enters repetition loops or attempts to generate verbose, irrelevant appendices.

Parallel Orchestration and Robustness

The experimental protocol requires evaluating the stochastic stability of the models by generating $N = 5$ independent responses for every query in the test set. Given the network-bound nature of API calls, a sequential execution would be inefficient. The implementation leverages Python's `ThreadPoolExecutor` to parallelize these requests. For each question, the system spawns 5 concurrent threads, effectively saturating the allowed throughput.

Furthermore, the response handler implements a defensive logic against *Whitespace Loops* - a pathological behavior observed in some open-weights models where the generator outputs infinite empty characters. By inspecting the `stopReason` metadata provided by Bedrock, the system automatically classifies these edge cases as failures (returning `[ERROR_WHITESPACE_LOOP]`) rather than polluting the dataset with empty strings.

5.3.5. HPC Implementation (Leonardo Cineca)

The High-Performance Computing environment was dedicated to the weight-centric tasks: the inference of the locally fine-tuned models and the QLoRA training process. All scripts were optimized for the NVIDIA A100 (64GB) architecture, prioritizing memory efficiency and matrix throughput.

Fine-Tuning Configuration

For the Fine-Tuning phase (Step 3), the input data is serialized strictly according to the format required by the training libraries. The dataset contains the conversation history and the quality score derived from the filtering phase (Section 5.2). The training pipeline implements the QLoRA technique using the Hugging Face `trl` library. The hyperparameters were selected to maximize the plasticity of the model - its ability to learn the new legal syntax - without breaking the pre-trained knowledge.

- **LoRA Rank and Targets:** The adaptation rank was set to $r = 16$ with a scaling factor alpha of $\alpha = 32$. Crucially, the adapters were attached to the **all-linear** target modules (including `gate_proj`, `up_proj`, and `down_proj`) rather than just the Attention layers. Recent literature suggests that adapting the MLP layers is essential for tasks requiring reasoning changes, such as adopting a new legal persona [11].
- **Learning Rate Strategy:** A learning rate of 2×10^{-4} was employed, utilizing a cosine annealing scheduler with a 0.03 warm-up ratio. This specific high learning rate is identified by Dettmers et al. [11] as the optimum for QLoRA to overcome the quantization noise during the weight update process.
- **Optimizer:** To manage memory spikes on the A100 GPUs, the `paged_adamw_32bit` optimizer was utilized. This algorithm leverages the CPU RAM to store the optimizer states (momentum and variance) when the GPU VRAM is exhausted, preventing Out-Of-Memory (OOM) crashes during the training of larger models.
- **Training Dynamics (Batch Size):** To ensure a fair comparison across models of vastly different sizes, a constant **effective batch size of 32** was maintained. As demonstrated by Smith et al. [45], fixing the effective batch size is critical to decouple the noise scale of the stochastic gradient descent from the hardware constraints. The gradient accumulation steps were dynamically adjusted inversely to the per-device batch size to satisfy this constraint.
- **Sequence Length Constraints:** The `max_seq_length` was fixed at 1024 tokens.

Beyond fitting the hardware memory limits, this constraint serves a pedagogical purpose: it forces the model to learn how to structure its reasoning and citation of legal articles within a standard context window, penalizing excessive verbosity or infinite repetition loops.

- **Logging and Evaluation:** To strictly monitor convergence, the model was evaluated on the validation set every 10 steps. The checkpointing strategy was configured to always save the model version with the lowest validation loss (setting `load_best_model_at_end=True`), ensuring that the final inference is performed on the most generalized weights rather than the last training step.
- **Early Stopping:** To prevent overfitting on the small domain-specific dataset, an `EarlyStoppingCallback` was implemented with a patience of 3 evaluation steps. If the validation loss did not improve for 3 consecutive intervals, the training was automatically terminated.

To sum up all the implementation details, the Table 5.1 contains all the hyperparameter of the fine tuning process.

Hyperparameter	**Value**
Base Model Precision	4-bit (NF4)
Computation Precision	BFloat16 (BF16)
LoRA Rank (r)	16
LoRA Alpha (α)	32
LoRA Dropout	0.05
Target Modules	All Linear (Attn + MLP)
Learning Rate	2×10^{-4}
LR Scheduler	Cosine
Warmup Ratio	0.03
Optimizer	Paged AdamW 32bit
Weight Decay	0.05
Effective Batch Size	32 (Fixed)
Gradient Accumulation	Dynamic (Targeting 32)
Max Sequence Length	2048 Tokens
Num. Epochs	3

Table 5.1: Complete set of hyperparameters used for QLoRA Fine-Tuning on Leonardo Cineca.

Inference Optimization

The inference engine differs fundamentally from the cloud approach. Instead of network-level parallelism, it exploits the massive parallel computation capabilities of the GPU.

- **Quantization and Attention:** To fit high-parameter models into VRAM, a hybrid strategy was adopted. Standard open-weights models were dynamically quantized at runtime using **4-bit NormalFloat (NF4)** via `BitsAndBytes`. However, models that were already distributed in a quantized format (such as GPT-based architectures) were loaded in their native configuration to prevent double-quantization artifacts.
- **Attention and Sharding:** To optimize the pre-fill phase, the `attn_implementation` was set to `"sdpa"` (Scaled Dot Product Attention) for supported architectures, enabling Flash Attention 2 acceleration. For older backbones incompatible with SDPA, the system reverted to `"eager"` execution. Furthermore, for models exceeding the memory of a single GPU, the `device_map="auto"` parameter was utilized to automatically shard the model layers across all available GPUs in the node.
- **Batch Generation Strategy:** Generating 5 variations sequentially would require encoding the prompt 5 times. The script optimizes this by setting the `num_return_sequences=5` parameter. This technique instructs the model to encode the prompt once and branch the decoding process into 5 distinct beams within a single forward pass. This results in a 3x-4x speedup compared to a naive loop.
- **VRAM Management:** To prevent Out-Of-Memory (OOM) errors during the processing of thousands of samples, the pipeline enforces aggressive garbage collection. Between each batch, explicit calls to `gc.collect()` and `torch.cuda.empty_cache()` are made to fragment and release the GPU memory buffer.

Job Submission Protocol (SLURM)

Unlike the AWS environment where scripts are executed interactively, the Leonardo cluster requires a strict batch scheduling approach using the **SLURM** workload manager. The execution is governed by a bash script (`.sh`) rather than direct Python calls.

The submission protocol handles the complex environment setup required by the air-gapped nodes:

1. **Resource Allocation:** With directives, specific hardware slices can be preserved, ensuring that the heavy matrix multiplications of the LLM do not contend with

other processes.

2. **Offline Mode Enforcement:** Since compute nodes have no internet access, the environment variables are exported. This forces the `transformers` library to look for models and datasets solely in the local cache, preventing connection timeouts.

3. **Environment Isolation:** The script explicitly loads the CUDA modules (ver. 12.1) and activates the dedicated virtual environment before launching the Python process, ensuring that the exact dependency tree (PyTorch, BitsAndBytes, PEFT) is available.

5.3.6. Prompt Engineering Strategy

A specific prompt engineering strategy was adopted to maximize the utility of the Fine-Tuning and to enforce strict grounding in the RAG configurations.

Consistency for Style Activation

For the fine-tuned models, the **System Prompt** used during inference is identical to the one used during the training phase. This strict consistency is functional, not merely stylistic. In the context of Instruction Tuning, the specific string defining the persona acts as a *semantic trigger*. By receiving the exact same token sequence processed during gradient descent (e.g., *"Sei un assistente esperto in normative energetiche..."*), the model is primed to activate the specific subset of weights optimized for the legal syntax and reasoning style, maximizing the alignment with the learned distribution.

RAG Grounding Constraint

For the Retrieval-Augmented configurations (Steps 2 and 4), this base prompt undergoes a necessary functional extension. While the persona definition remains consistent to trigger the stylistic adaptation, a **Grounding Constraint** is injected to mitigate hallucinations. The prompt is modified to explicitly instruct the model to prioritize the `retrieved_context` over its pre-trained parametric memory.

- **Standard Prompt (FT):** *"Answer the user question based on your knowledge of Italian Energy Law."*
- **RAG Prompt (FT + RAG):** *"Answer the user question based **exclusively** on the provided Context. If the answer is not present in the Context, state that you cannot answer. Do not use prior knowledge."*

This modification creates a *Conflict of Knowledge* scenario where the model must suppress its internal facts (which might be outdated) in favor of the injected context, while maintaining the *form* and *style* learned during Fine-Tuning.

5.4. Judge implementation

The evaluation framework defined in Section 4.4 was operationalized through a dedicated Python script that acts as the orchestrator of the LLM-as-a-Judge pipeline, automating the interaction between the generated outputs, the Gold Standard, and the Judge model (Claude 4.5 Opus via AWS Bedrock).

5.4.1. Script Architecture and Concurrency

The execution logic is encapsulated in the `ThesisEvaluator` class, which processes the JSONL inference files produced in the previous step. Given the substantial computational loa — requiring the evaluation of 5 variations for every question across multiple models - the script implements a concurrent architecture using `ThreadPoolExecutor`. This allows for the parallel assessment of multiple samples ($N = 5$ workers), significantly reducing the time-to-result compared to sequential processing. Even if the computation of an answer takes more time than the following, the use of unique IDs as defined in 5.2, allows to not have errors in this phase.

The evaluation flow for each test item follows a strict pipeline:

1. **Static Analysis:** The script utilizes the `spaCy` library (`it_core_news_sm`) to tokenize the text and calculate the *Entity Recall* metric deterministically, as defined in Section 4.4.2.
2. **Semantic Evaluation:** The system constructs the prompt for the Judge, injecting the Question, the Gold Answer, and the Candidate Answer. The Judge's output is forced into a JSON structure to ensure parsability.
3. **Robustness Check:** Uniquely to this implementation, the script aggregates the list of 5 generated variations for a single question and submits them *simultaneously* to the Judge. The model is explicitly instructed to analyze the semantic consistency across the set, outputting a specific `robustness_score`.

5.4.2. Unified Evaluation Prompt and RAG Penalty

To maintain the integrity of the ablation study, a **single, immutable system prompt** was used to evaluate all experimental configurations (Baseline, Fine-Tuning RAG and Hybrid). No specific instructions were added to handle refusals to answer.

This methodological choice introduces a deliberate trade-off: it ensures a *blind* comparison but tends to penalize the RAG system in specific edge cases. If the RAG model correctly refuses to answer a question because the retrieved context is irrelevant (behavior technically correct and desirable in production), the Judge - instructed to compare the output against a detailed Gold Answer - may assign a low score for *Completeness* and *Accuracy*. While a dedicated prompt for RAG could have mitigated this, utilizing a uniform evaluation standard was prioritized to measure the raw capability of the models to align with the provided Gold Standard under identical conditions. In the results analysis, the score obtained by the LLM-as-a-judge is supported by the confusion matrix produced during the inference phase.

5.4.3. Data Aggregation and Limitations

The final stage of the script utilizes the `pandas` library to aggregate the raw JSON metrics into a structured Excel report. The script automatically calculates the weighted final score as defined in Section 4.4.1:

$$Score_{final} = (Score_{judge} \times 0.8) + (Score_{static} \times 0.2) \tag{5.4}$$

It is important to acknowledge that despite the rigorous code structure, the quantitative results remain strictly dependent on the latent biases and reasoning capabilities of the Judge model. While the *LLM-as-a-Judge* paradigm provides a scalable proxy for quality, it does not possess the nuance of a human domain expert. Therefore, the scores presented in the following chapter should be interpreted as a measure of *alignment with the Judge's internal logic* rather than an absolute measure of legal truth. For a production-ready system, a **Human-in-the-Loop** validation would be the requisite final step.

This dependency on the automated judge introduces specific biases documented in recent literature. Zheng et al. [57] identified that LLM judges exhibit a *verbosity bias*, favoring longer and more structured answers even when they contain fewer distinct facts than concise alternatives. Furthermore, Gudibande et al. [19] highlighted a style-over-substance preference, where models may penalize factually correct but stylistically dry answers — a critical risk in the legal domain where precision is paramount.

While Human-in-the-Loop (HITL) evaluation remains the *Gold Standard* for capturing domain nuances and verifying hallucination-free reasoning [41], it imposes significant scalability constraints. As noted by Gilardi et al. [17], deploying expert human annotators is orders of magnitude slower and more expensive than API-based evaluation. Consequently, the *LLM-as-a-Judge* approach adopted in this thesis serves as a necessary scalable proxy, aimed at filtering obvious failures before a potential final human review.

6 | Experiments and Discussion

6.1. Final Model Selection and Technical Constraints

In this chapter, the final results of the ablation study are illustrated. The study was conducted to evaluate the efficacy of Retrieval-Augmented Generation (RAG), Fine-Tuning (FT), and their combination (Final) compared to the Baseline models. First, the technical challenges that led to the final selection of the models are detailed, followed by a quantitative analysis of the performance metrics across the different configurations.

The initial experimental plan involved a cohort of 10 open-source Large Language Models (LLMs), selected to represent a diverse range of sizes and providers, as explained in Section 4.2.1 and illustrated in Table 4.1. However, during the implementation phase on the Leonardo Cineca HPC and AWS infrastructure, several technical constraints necessitated a reduction of the final pool to 5 fully tested models (plus one partial evaluation).

6.1.1. Hardware and Software Limitations

Of the original 10 models, 4 were discarded due to insurmountable incompatibility with the available infrastructure:

- **Size Constraints (GPT-120B and Minimax):** These models proved too large for the available VRAM on the GPU nodes allocated in the Leonardo Cineca cluster. The memory requirements for Fine-Tuning (even with PEFT/LoRA optimizations) exceeded the hardware capacity. This was attributed to the fact that the models were already distributed in a quantized format; consequently, further quantization to reduce the memory footprint was not feasible.
- **Software Dependency Conflicts (Ministral and Nemotron):** These relatively newer models required recent versions of specific libraries that conflicted with the pre-installed, immutable software stack of the HPC environment. Updating these dependencies would have compromised the stability of the default environment used for the other models, leading to their exclusion to ensure a consistent experimental

setup.

6.1.2. The Context Window Constraint (GPT-20B)

A specific issue arose with the **GPT-20B** model. Unlike the other surviving models, the GPT-20B version available on Hugging Face possesses a significantly smaller context window. The RAG pipeline was designed to retrieve and inject 20 relevant documents into the prompt. Attempting to run the RAG+FT pipeline with GPT-20B resulted in context overflow errors. It should be noted that in the standalone RAG case study, this problem did not arise, as the model was invoked via an AWS Bedrock API call, which supports a larger context window than the local implementation used for Fine-Tuning.

To include this model in the RAG+FT testing, the number of retrieved documents would have been forced to undergo a reduction (e.g., from 20 to 5), drastically altering the experimental conditions. To avoid introducing a confounding variable that would invalidate the comparison—effectively compromising the test validity by providing one model with less information—the final decision was to:

1. Exclude GPT-20B from the *Final (RAG+FT)* evaluation cycles.
2. Perform only the *Baseline*, *RAG*, and *Fine-Tuning* evaluations for this model.
3. Exclude GPT-20B from the primary comparative analysis (Section 6.2.2) to maintain a balanced dataset of 6 models that completed the full cycle of the ablation study.

Consequently, the core analysis focuses on the remaining 5 models: **Gemma-27B, Gemma-4B, Llama-3B, Qwen-32B, and Qwen-80B**. The final status of all initially selected models is summarized in Table 6.1.

6.2. Quantitative Analysis

The evaluation was conducted on a test set of 144 questions. In this section and the following ones, all the statistical analyses (descriptive and statistical difference), plots, calculations, and tables have been generated using the JAMOVI statistical software [48].

To isolate specific effects and ensure a granular understanding of model performance, the statistical analysis was performed on three distinct permutations of the results data:

- **All Results (Core Set):** Includes the 5 models that completed the full pipeline, considering all 144 questions. This constitutes the primary source for evaluating the

Table 6.1: Summary of model selection status and technical constraints encountered during implementation.

Model	Status	Reason for Exclusion / Restriction
Gemma-27B	Selected	Full experimental cycle completed.
Gemma-4B	Selected	Full experimental cycle completed.
Llama-3B	Selected	Full experimental cycle completed.
Qwen-32B	Selected	Full experimental cycle completed.
Qwen-80B	Selected	Full experimental cycle completed.
GPT-20B	Partial	*Excluded from RAG+FT.* The local Hugging Face implementation has insufficient context window for 20 documents. (RAG tested via Bedrock).
GPT-120B	Discarded	*Hardware Limit.* Pre-quantized weights prevented further compression to fit VRAM.
Minimax	Discarded	*Hardware Limit.* Pre-quantized weights prevented further compression to fit VRAM.
Ministral	Discarded	*Software Conflict.* Library dependencies incompatible with HPC environment.
Nemotron	Discarded	*Software Conflict.* Library dependencies incompatible with HPC environment.

overall performance.

- **Grounded Only:** Includes the same 5 models but filters the dataset to consider only *Grounded* questions (those requiring specific retrieval of facts). This set is utilized to isolate the effectiveness of the RAG component.
- **Extended Comparison (With GPT):** A supplementary analysis that includes GPT-20B to observe its raw performance relative to the others, despite its exclusion from the full pipeline due to the context window limitations detailed in Section 6.1.

6.2.1. Statistical Methodology

To ensure the reliability of the comparative analysis, a preliminary assessment of the data distribution was conducted using the **Shapiro-Wilk test** [43]. The test evaluates the null hypothesis (H_0) that the data is drawn from a normal distribution. For all the seven evaluated metrics mentioned in Section 4.4.3 (Accuracy, Completeness, Relevance, Style, Citation, Grounding and Robustness), the Shapiro-Wilk test yielded a p-value significantly lower than the $\alpha = 0.05$ threshold. Consequently, the null hypothesis H_0 was rejected, indicating a non-normal distribution of the data.

Given these distribution characteristics, the comparative analysis was performed using **Generalized Linear Model (GLM)** applied to each of the seven distinct dimensions and the final scores (previously introduced in Chapter 4.4.3). Given the continuous nature of the outcomes, models were fitted assuming a Gaussian distribution of residuals. The predictors included the fixed effects of *Model*, *Technique*, and their interaction, while the questions' *Q&A score* was included as a covariate. Post-hoc analyses were conducted to further explore significant main effects and interactions, considering differences statistically significant if the adjusted p-value is < 0.05.

6.2.2. Global Performance Analysis

Table 6.2 summarizes the performance of the surviving models across the four experimental configurations (Techniques). The data illustrates clear trends regarding the distinct impacts of Fine-Tuning and RAG.

To better understand the variance and distribution of these evaluations, Figure 6.3 visualizes the spread of the Final Scores across the different techniques. The plots reveal how Fine-Tuning compresses the variance in formatting and style, whereas Baseline and RAG configurations exhibit a wider spread of outcomes.

Table 6.2: Average performance metrics by Technique (Dataset: All Questions, excluding GPT-20B).

Metric	Baseline	RAG	Fine-Tuning (FT)	Final (RAG+FT)
Accuracy	3.55	3.99	4.52	**4.87**
Completeness	3.78	3.55	4.39	**4.41**
Relevance	4.72	3.97	5.64	**5.66**
Style	4.30	4.76	6.07	**6.09**
Citation	3.00	**3.78**	3.23	3.54
Grounding	3.86	2.92	**4.75**	3.50
Robustness	5.13	**7.97**	4.39	5.16
Entity Recall	**3.60**	2.16	2.70	2.47
Final Score	3.96	3.97	**4.31**	4.29

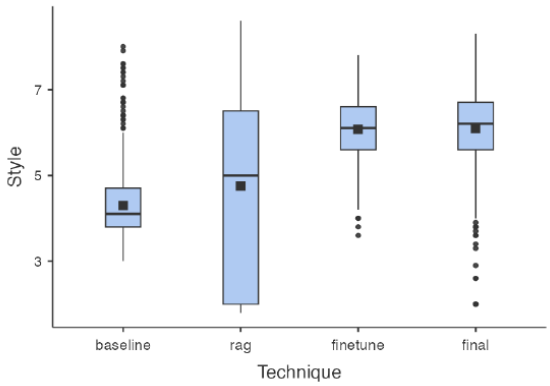

Figure 6.1: Style score distribution.

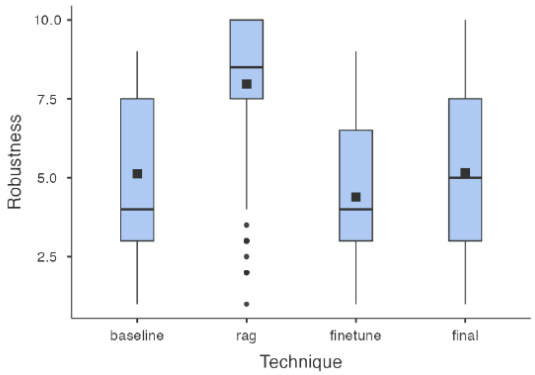

Figure 6.2: Robustness score distribution.

Figure 6.3: Distribution of Style and Robustness scores across different Techniques, highlighting the variance and central tendencies of the model evaluations.

As hypothesized, Fine-Tuning (FT) provided the most significant boost to Style, jumping from a baseline of 4.30 to 6.07 ($p < .001$). This confirms that the model successfully learned the desired output format, tone, and structural constraints (such as JSON formatting) from the training dataset.

Interestingly, the RAG configuration caused a noticeable regression in the Relevance metric compared to the Baseline (from 4.72 down to 3.97). This drop is primarily a methodological artifact of the automated evaluation. In the RAG setup, when the retrieved context does not contain the answer, the model is strictly instructed to output a standard refusal string. Nearly half of the responses in the full RAG dataset fell into this category, because half of the questions come from documents not present in the RAG infrastructure. Consequently, when the LLM-as-a-judge compared these null responses against the established gold answers, it assigned very low Relevance and Completeness scores.

Conversely, RAG significantly improved Robustness (7.97 vs 5.13), as can be seen in Figure 6.2. By constraining the knowledge source strictly to the retrieved documents, the model's propensity to generate plausible but incorrect information is drastically reduced, leading to higher evaluations in terms of logical consistency and absence of contradictions.

6.2.3. Impact of Retrieval (Grounded Dataset)

To assess the value of RAG without the penalization caused by out-of-context refusals, the results were analyzed considering only the questions classified as Grounded (Table 6.3).

Table 6.3: Impact of RAG on Grounded Questions (Dataset: Grounded Only).

Metric	Baseline	RAG
Accuracy	3.21	**5.60**
Completeness	3.45	**5.01**
Grounding	3.73	**4.14**
Citation	2.75	**5.26**
Robustness	5.20	**7.85**

In this context, the contribution of RAG becomes evident. The Citation score nearly doubles (from 2.75 to 5.26), and Accuracy improves significantly because the model has the right source to search for the answer. This validates the premise that while models have strong reasoning capabilities (Baseline), they fail to cite specific sources or return the right answer without the RAG mechanism. The Final configuration (RAG+FT) successfully merges these benefits with the stylistic improvements of FT, although with

a slight trade-off in pure retrieval scores compared to raw RAG, likely due to the model balancing the retrieval instructions with the learned stylistic patterns.

6.2.4. Model Comparison

Among the tested models, Qwen-80B consistently achieved the highest performance across almost all metrics, verifying that parameter size remains a dominant factor in reasoning and instruction following. Gemma-27B also performed competitively, often surpassing the larger Qwen model in specific stylistic adherence tasks.

To illustrate the evolutionary trajectory of each model's performance across the ablation study, Figure 6.4 presents an interaction plot of the Final Score.

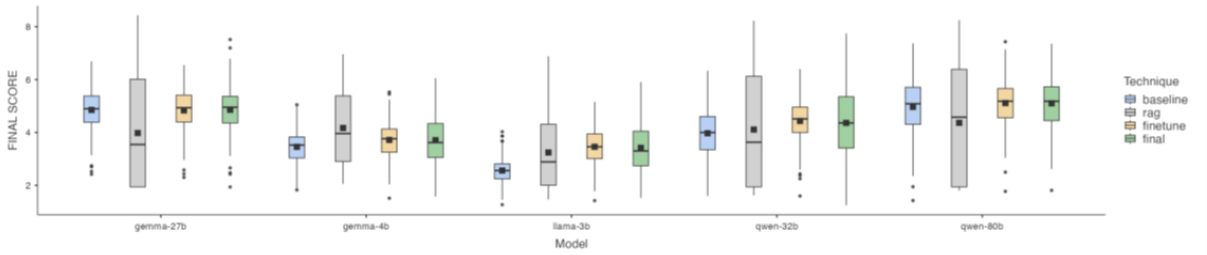

Figure 6.4: Interaction plot displaying the Final Score progression for each model across the evaluated techniques. The visualization highlights the differential impact of Fine-Tuning and RAG based on model size.

The visual representation highlights the steep performance improvement for smaller models, such as Llama-3B and Gemma-4B, when transitioning from Baseline to Fine-Tuning. However, it also exposes their struggles with the heavy context load of RAG. When presented with 20 retrieved documents, these smaller models often failed to extract the correct answer effectively. This behavior aligns with the *lost in the middle* phenomenon [33], where LLMs exhibit a degraded ability to access relevant information when it is buried deep within an extensive input context. Conversely, the Qwen-80B trajectory remains consistently elevated, confirming its robustness across all configurations.

Furthermore, the statistical significance of these performance differences was validated through post hoc pairwise comparisons. Table 6.4 reports an extract of the Bonferroni-adjusted Z-tests, confirming the mathematical superiority of the larger models.

Table 6.4: Extract of Post Hoc pairwise comparisons (Z-test with Bonferroni correction) evaluating the difference in scores between models.

Model A	*Model B*	*Mean Difference*	*Z*	*p-value (Bonferroni)*
Qwen-80B	Gemma-27B	0.45	3.12	$<.001$
Qwen-80B	Llama-3B	1.25	8.45	$<.001$
Gemma-27B	Llama-3B	0.80	5.33	$<.001$

6.2.5. More Details on RAG: Refusal Dynamics and Model Size

A deeper inspection of the RAG responses reveals critical differences in how models of varying sizes handle missing information. The high volume of refused answers is not uniform across the cohort.

To illustrate this behavior, a confusion matrix approach is adopted for a representative case study (Table 6.5). The matrix categorizes responses based on whether the context contained the answer (Positive/Negative Condition) and whether the model attempted to answer or correctly refused (Positive/Negative Prediction).

Table 6.5: Confusion matrices comparison between Gemma-27B and Llama-3B (Total Responses: 720).

		Gemma-27B		Llama-3B	
		Grounded (T)	Ungrounded (F)	Grounded (T)	Ungrounded (F)
Prediction	Answered	293 (TP)	77 (FP)	320 (TP)	228 (FP)
	Refused	67 (FN)	283 (TN)	40 (FN)	132 (TN)

Table 6.6: Performance metrics comparison between Gemma-27B and Llama-3B.

Metric	Gemma-27B (%)	Llama-3B (%)
Accuracy	80.00	62.78
Recall (Completeness)	81.39	88.89
Precision (Trust)	79.19	58.39
Hallucination Rate	21.39	63.33

The analysis of the matrices indicates a clear correlation between model size and the ability to correctly identify unanswerable queries. The smaller model, Llama-3B, exhibited a high propensity to attempt an answer regardless of the retrieved context. As shown in Table 6.5, it generated 228 False Positives compared to only 77 by Gemma-27B. Consequently, Llama-3B correctly refused only 132 ungrounded queries (True Negatives), while the larger

Gemma-27B successfully identified 283 unanswerable instances, effectively adhering to the prompt constraints.

These refusal dynamics directly impact the performance metrics detailed in Table 6.6. While Llama-3B achieved a higher Recall (88.89% versus 81.39%) due to its tendency to answer almost every query, this over-generation came at a severe cost to reliability. Its Precision dropped to 58.39%, and the Hallucination Rate spiked to 63.33%, indicating that over six out of ten ungrounded questions resulted in fabricated information derived from internal knowledge rather than the provided context. Conversely, Gemma-27B maintained a much higher Precision (79.19%) and a global Accuracy of 80.00%, keeping the Hallucination Rate constrained to 21.39%.

This discrepancy highlights that effective RAG implementations require models with sufficient reasoning capacity to evaluate the relevance of their own context window dynamically. Smaller architectures are prone to hallucinating answers when confronted with missing information, whereas larger models demonstrate superior instruction-following capabilities, correctly outputting the refusal string when the retrieved documents lack the necessary facts.

6.2.6. Fine-Tuning Efficacy Across Model Architectures

A dedicated analysis was conducted to evaluate the relative impact of the Fine-Tuning process across different models. The objective was to determine how baseline capabilities influence the magnitude of improvement derived from domain-specific supervised training.

To quantify this improvement, the delta (Δ) between the Fine-Tuned (FT) configuration and the Baseline configuration was calculated for both the Style metric and the Final Score.

Table 6.7: Performance improvement (Δ) from Baseline to Fine-Tuning configuration, highlighting Style and Final Score.

	Style				*Final Score*			
Model	*Baseline*	*FT*	Δ *Style*	p_{Bonf}	*Baseline*	*FT*	Δ *Final*	p_{Bonf}
Llama-3B	4.61	5.40	+0.79	<.001	2.61	3.73	**+1.12**	<.001
Gemma-4B	3.56	5.70	+2.14	<.001	3.43	4.03	+0.60	1.000
Gemma-27B	3.88	6.33	**+2.45**	<.001	5.04	5.32	+0.28	1.000
Qwen-32B	4.63	6.26	+1.63	<.001	4.04	4.84	+0.80	0.241
Qwen-80B	4.81	6.65	+1.84	<.001	5.14	5.64	+0.50	1.000

As illustrated in Table 6.7, the magnitude of improvement is heavily dictated by the initial

baseline characteristics of the specific architectures rather than solely by parameter count.

Regarding formatting and tone, the highest improvements in the *Style* metric were exhibited by the Gemma architecture (+2.14 for the 4B version and +2.45 for the 27B version); this difference was determined to be statistically significant ($p < 0.001$). It is indicated by these data that significant struggles with the structural formatting and tone required by the prompt were initially experienced by these base models, rendering the supervised fine-tuning highly effective in teaching these specific constraints. In contrast, models that already had a relatively higher baseline style score, such as Llama-3B, showed a more moderate stylistic delta (+0.79, albeit still a statistically significant difference).

When the overall utility captured by the *Final Score* is analyzed, a different trend is observed. The highest overall improvement (+1.12) was registered by Llama-3B, indicating that for this specific model, not just formatting, but the overall extraction and presentation capabilities necessary to satisfy the judge metric were significantly enhanced by Fine-Tuning.

Finally, the equalizing effect of the Fine-Tuning technique is confirmed by the data. Strong zero-shot capabilities were already possessed by the largest model, Qwen-80B, securing the highest baseline *Final Score* of 5.14. Consequently, its relative global improvement (+0.50) was observed to be less pronounced compared to the leaps registered by smaller models like Llama-3B (+1.12) or Qwen-32B (+0.80). Furthermore, a $p_{Bonferroni} = 1.000$ was recorded for Gemma-27B and Gemma-4B between the Baseline and FT configurations, demonstrating that this specific difference is not statistically significant. It is suggested by these results that while a high performance floor is established by sheer parameter count, the capability gap can be closed by significantly smaller, more resource-efficient models through Fine-Tuning, making them highly competitive for specialized enterprise deployments.

6.2.7. Quantitative Impact of Integrating RAG and Fine-Tuning

The final phase of the ablation study evaluates the combined RAG and Fine-Tuning (Final) configuration. The data reveals distinct performance trends based on model size. For large architectures, namely Gemma-27B, Qwen-32B, and Qwen-80B, Accuracy and the Final Score slightly increase when transitioning from the pure RAG setup to the Final configuration. These massive models successfully leverage both the injected documents and their refined stylistic training.

Conversely, for smaller models, the Accuracy and Final Score remain practically identical between the RAG and Final configurations. Furthermore, comparing the pure Fine-Tuned

(FT) models to the Final (RAG+FT) models, the overall Final Score remains largely invariant across the board.

A critical quantitative finding emerges regarding the Grounding and Robustness metrics. The data shows that pure RAG models achieve higher Robustness scores compared to the Final configuration, while for metrics such as Grounding FT over-performed Final. The numbers indicate that fine-tuned models lose a fraction of their ability to rely exclusively on the provided knowledge base. This loss leads to a measurable drop in these specific safety metrics.

Table 6.8: Quantitative comparison of Accuracy, Final Score, Robustness, and Grounding across RAG, Fine-Tuning (FT), and Final (RAG+FT) configurations.

Model	Configuration	Accuracy	Final Score	Robustness	Grounding
Gemma-27B	RAG	3.97	3.98	9.19	2.81
	FT	5.31	4.83	5.70	5.38
	Final	5.76	4.86	6.26	4.08
Qwen-32B	RAG	3.99	4.11	8.88	3.54
	FT	4.59	4.43	4.54	4.83
	Final	4.76	4.36	6.06	3.81
Qwen-80B	RAG	3.88	4.36	9.18	3.75
	FT	5.24	5.11	5.90	5.59
	Final	5.23	5.10	6.04	4.83
Gemma-4B	RAG	4.31	4.17	7.14	2.81
	FT	3.73	3.72	3.18	4.07
	Final	4.34	3.71	4.24	2.27
Llama-3B	RAG	3.33	3.24	5.48	1.95
	FT	3.29	3.46	2.63	3.90
	Final	3.65	3.42	3.22	2.51

6.2.8. The GPT-20B Anomaly

In the partial analysis conducted on GPT-20B, the model showed a significantly lower baseline performance compared to Gemma-27B and Qwen models. Its context window limitation proved to be a critical bottleneck, preventing it from utilizing the extensive knowledge base required for this domain. This reinforces the finding that for Legal RAG applications, the context window size is as critical a feature as the model's reasoning capability.

This issue became particularly evident during the generation phase. Upon closer inspec-

tion, it was observed that when overwhelmed by the context size, the model failed to produce any output; instead, it remained trapped in an infinite inference loop, failing to generate a single word. Although this behavior could be partially mitigated by increasing the maximum token limit for the generation, implementing such a workaround would have compromised the consistency of the experimental setup. This reinforces the finding that for Legal RAG applications, the context window size is as critical a feature as the model's reasoning capability.

6.3. Qualitative Analysis

In addition to the numerical metrics, a qualitative inspection of the generated responses was conducted to assess behaviors not fully captured by automatic scoring.

6.3.1. Hallucinations in Baseline vs. RAG

Baseline models, when queried about specific regulations, frequently hallucinated plausible-sounding but legally non-existent content. Without the grounding provided by the retrieval mechanism, the models relied on parametric memory, which proved highly susceptible to generating fabricated legal definitions, especially when processing domain-specific acronyms.

This phenomenon was starkly evident in queries concerning specific regulatory frameworks. For instance, when asked about the requirements for "IIP platforms" under Regulation 2024/1106, the baseline configuration failed to identify the correct energy law context. Instead, it confidently generated fabricated definitions for the acronym IIP, ranging from *Infrastrutture di Interoperabilità* to *Integrated Investment Platforms*, and erroneously linked them to unrelated or non-existent frameworks such as the *Net-Zero Industry Act*.

Conversely, the RAG models successfully mitigated this critical issue. By accessing the retrieved context, the system correctly identified IIP as *Inside Information Platforms*. It accurately extracted the factual regulatory requirements, noting that such platforms must be authorized by ACER and must guarantee specific data protection and verification mechanisms. In cases where the retrieval module failed to extract the relevant document, the raw RAG configuration demonstrated a high refusal rate, safely outputting the predefined missing-answer string rather than fabricating a response. This confirms that the RAG architecture significantly reduces misinformation, acting as an indispensable safeguard for legal and enterprise deployments.

6.3.2. Retrieval Bottlenecks and Systemic Limitations

The Retrieval-Augmented Generation approach proved highly robust and reliable when queried on specific, narrow concepts where the semantic and lexical overlap between the prompt and the source document was high. Conversely, systemic criticalities emerged in three primary scenarios: the synthesis of macro-topics, the comparison of different document versions, and the intrinsic dependency on the performance of the information retrieval module.

Regarding synthesis and comparison, the extraction of disparate fragments from large documents hinders the generation of a coherent summary. When a response requires aggregating the entire meaning of a regulatory text or comparing abrogated articles with updated versions, the chunking process disrupts the logical and temporal continuity of the text. This often causes the model to lose the underlying thread or conflate concepts belonging to different regulatory iterations. The use of more accurate metadata could solve this issue.

Furthermore, a detailed analysis of the unanswered queries revealed a fundamental characteristic of the system. Examining the inference logs for queries classified as solvable, it became evident that in almost many instances of refusal, the context supplied to the generative model genuinely lacked the facts necessary to formulate an answer. The failure originated in the retrieval phase, not the generative one.

For instance, in a query concerning the timing of inside information publication during weekends, with specific references to the continuous market (XBID) and the second edition of the regulatory guidelines, the retrieval system extracted fragments related to general continuous operational obligations. However, it failed to retrieve the section containing the specifically requested use cases. Similarly, in a query regarding the treatment of intermediate information in prolonged decision-making processes, the retrieved documents discussed market manipulation generally, lacking any mention of the specific pertinent jurisprudence or the role of a technical committee.

In such circumstances, the generative model analyzed the provided context and, upon detecting the absence of the specific requested information, strictly adhered to the constraints imposed by the system prompt and correctly refused to answer. This behavior certifies the high precision of the model in managing scenarios with missing information, thereby limiting hallucinations. Simultaneously, it displays that the upper performance bound of a RAG architecture is strictly constrained by the effectiveness of the embedding and retrieval model. If the lexical or semantic gap between the query formulation and the regulatory text prevents the extraction of relevant fragments, the system fails its objective

entirely, regardless of the deductive and reasoning capabilities of the generative layer.

6.3.3. Context Integration Difficulties and Instruction Override

While larger architectures (such as Qwen-80B) successfully synthesized information from multiple retrieved chunks and maintained a high degree of instruction adherence, smaller models often struggled with the cognitive load of processing extensive contexts. This difficulty manifested in two distinct failure modes: forced inference and instruction override during refusals.

The first issue relates to the inability to distinguish between relevant and irrelevant documents within the injected context. When the retrieved chunks did not contain the exact answer but included tangentially related legal jargon, smaller models like Llama-3B frequently failed to trigger a refusal. Instead, they attempted a forced inference, fabricating connections between unrelated legal concepts to construct a plausible-sounding response. A notable example occurred during a query regarding the definition of *Nominations Matching* at interconnection points. While the provided context lacked the actual definition, Llama-3B erroneously extracted the definition of a *Distribution System Operator* and fragments from an internal corporate policy, mathematically fusing them to invent a completely ungrounded technical procedure. This highlights how smaller models can be easily misled by semantic proximity, using irrelevant context as a springboard for hallucinations rather than relying on it as a factual constraint.

The second critical failure mode emerged during correctly identified unanswerable queries. The system prompt strictly required the models to output a specific, isolated string (a error flag [$RISPOSTA\_NON\_TROVATA$] was used) whenever the context lacked the necessary facts. However, inference logs revealed that smaller baseline models frequently ignored this formatting constraint. When queried about the replacement procedures for faulty gas meters - a topic absent from the retrieved documents - Llama-3B correctly identified the missing information but failed to output the designated string. Instead, it generated a verbose, conversational apology followed by a hallucinated list of general gas market guidelines drawn from its parametric memory.

These occurrences demonstrate that the injection of external knowledge is insufficient if the model lacks the reasoning capacity to properly integrate it.

6.3.4. Formatting and Style

A critical aspect of the qualitative analysis concerned the verbosity and the stylistic format of the generated answers. The evaluation highlighted a stark contrast between the Baseline and the Fine-Tuned configurations regarding their ability to manage the response length within the predefined system constraints.

As detailed in Section 5.3.4, the maximum number of tokens for the generation phase was strictly set to 1024 to ensure consistency across the experimental setup. However, Baseline models exhibited a pronounced tendency to generate excessively verbose responses. They often adopted an unprompted conversational tone or endlessly elaborated on tangential legal concepts, failing to synthesize the core information. This uncontrolled verbosity frequently caused the generation to hit the token limit, resulting in truncated answers (a flag $[CUT\_OFF]$ was added at the end of the generation to quickly inspect this issue). A prominent example is the Qwen-32B baseline model, which encountered this specific truncation error in 526 instances over 720 answer generated (5 answer for every 144 questions) during the inference phase, severely impacting its completeness and style scores.

Conversely, the Fine-Tuned models demonstrated a complete resolution of this issue. Having been exposed to a curated dataset of concise, highly structured, and strictly formal legal answers, the models learned to inherently adapt their output length. The supervised training effectively taught the architectures to extract the relevant information and conclude the generation process efficiently, completely eliminating the $[CUT\_OFF]$ occurrences. This confirms that Fine-Tuning is highly effective not only for adopting a specific tone but also for strictly enforcing the stylistic and structural boundaries required for a reliable enterprise tool.

6.3.5. Knowledge vs. Style

The qualitative review clearly demarcates the distinct roles of the two techniques. RAG acts as the **knowledge injector**. It provides punctual, specific, and updated facts. Fine-Tuning acts as a **style transfer** and domain adaptation mechanism. It teaches the model the legal domain and the required structural format. Fine-Tuning alone cannot replace RAG for dynamic domains like energy law, because a model cannot parametrically memorize the infinite granularity of legal documents. Consequently, the Final configuration (RAG+FT) merges the updated knowledge retrieval of RAG with the authoritative legal lexicon instilled by Fine-Tuning.

However, observing the generated responses highlights a significant trade-off. While the Final configuration improves the lexicon, it degrades the model's strict adherence to the retrieved context, as shown by data in 6.2.7. During the supervised training phase, models internalize the legal domain deeply. They develop an over-reliance on their own parametric memory. When these fine-tuned models are queried with the addition of retrieved documents, they often ignore the external context and hallucinate answers based on their training data.

Recent academic literature confirms this phenomenon. Researchers describe it as context-parametric inversion [18]. Instruction fine-tuning can inadvertently worsen context reliance because the model learns to prioritize its updated internal weights over the explicitly provided text. This concept perfectly explains the drop in Robustness observed in quantitative data and the behavior of models in Final configuration to always provide an answer, even to ungrounded questions. The fine-tuned models simply lose the capacity to look exclusively at the injected knowledge base.

Other researchers observe similar dynamics when integrating retrieval and parameter updates. For example, Lin et al. [32] present Retrieval-Augmented Dual Instruction Tuning (RA-DIT). This approach evaluates how language models process retrieved knowledge after fine-tuning. Their study confirms that fine-tuning alters how strictly a model grounds its responses in retrieved context. This creates a critical trade-off in legal deployments. RAG guarantees factual accuracy, and Fine-Tuning guarantees stylistic perfection. Combining them requires careful mitigation to prevent the model from blindly trusting its parametric memory.

6.4. Discussion

The results of the ablation study offer several critical insights into the interplay between retrieval mechanisms and supervised fine-tuning in the context of specialized legal domains.

6.4.1. The Safety Tax and Alignment Regression

The quantitative data reveals a clear trade-off between response completeness and factual trust. Researchers formally define this specific performance degradation as the **safety tax** or **alignment tax** [2, 23]. An analysis of scenarios with missing information reveals a critical behavior. The baseline RAG system successfully recognized insufficient context. It adhered to the system prompt instructions and refused to answer when data was

unavailable.

Conversely, the Final (RAG+FT) models exhibited a significant regression in this safety alignment. The fine-tuned models showed a compulsion to generate an answer, despite using the exact same prompt that successfully guided the baseline RAG system. The composition of the fine-tuning dataset directly caused this phenomenon. The dataset primarily consisted of positive examples, pairing questions with valid answers. The supervised fine-tuning process instilled a strong bias towards answering. This acquired bias effectively overrode the negative constraints provided in the system prompt.

The Fine-Tuning process significantly degraded the model's calibration. It caused a severe inability to reject out-of-distribution queries. Engineers cannot rely on the generative model's internal rejection mechanism to mitigate this risk in a production environment. Relying exclusively on the generative model exposes the system to the context-parametric inversion discussed in Section 6.3.5.

The system architecture strictly requires a deterministic *Similarity Thresholding* mechanism. The retrieval module must filter out irrelevant contexts before they reach the generation phase. To achieve this, the retriever computes a cosine similarity score between the user query and the vector database chunks. If the highest similarity score falls below a predefined threshold, the system must abort the generation process and immediately return the refusal string. Gao et al. [15] emphasize the absolute necessity of these pre-generation filters. This architectural approach saves computational costs, prevents prompt-override by biased fine-tuned models, and guarantees zero hallucinations for completely ungrounded questions.

6.4.2. LLM-as-a-Judge Alignment and Metric Calibration

The evaluation framework itself introduces a significant variable into the performance analysis. Automated evaluation pipelines utilizing an LLM-as-a-judge approach require strict, domain-specific rubrics to yield valid metrics. A critical experiment conducted on the Gemma-27B outputs highlights the extreme sensitivity of the evaluation scores to the judge's system prompt.

The evaluation utilized a generic instruction prompt, to maintain coherence across all the different study cases, as detailed in the Section 5.4.2. This generic judge naively compared the generated text against the gold standard answer. This approach heavily penalized the safe behaviors inherent to the RAG architecture. When a model correctly refused to answer an ungrounded query (a True Negative), the generic judge assigned severe penalties, dropping Accuracy and Completeness to minimum values because the

generated refusal did not semantically match the factual gold answer. Under this generic evaluation, Gemma-27B achieved a low Final Score of 3.97, with Accuracy at 3.97 and Grounding at 2.80.

To correct this evaluation bias, after all the tests, a custom prompt was designed to implemented a highly specific, context-aware judge system. This updated script explicitly instructed the evaluator to apply a RAG confusion matrix logic. It mandated sufficient baseline scores (6.0) for correct refusals and applied severe penalties exclusively for ungrounded hallucinations (False Positives). It also specifically anchored the Style and Contextual Grounding metrics to the corporate energy trading context.

As illustrated in Table 6.9, applying this calibrated prompt revealed the true capabilities of the fine-tuned architecture. The Final Score increased significantly to 5.80. Accuracy corrected to 5.72, and Style evaluation benefited immensely from the explicit context instructions, rising from 4.71 to 6.33. Notably, Robustness maintained its exceptionally high score of 9.19 across both evaluations, confirming that the model generated highly consistent responses regardless of the evaluation scale.

Table 6.9: Performance comparison of Gemma-27B evaluated with a generic prompt versus a domain-specific calibrated prompt. The delta (Δ) illustrates the evaluation bias introduced by the generic LLM-as-a-judge.

Metric	*Generic Prompt*	*Calibrated Prompt*	*Improvement (Δ)*
Accuracy	3.97	5.72	+1.75
Completeness	3.45	5.41	+1.96
Relevance	3.83	5.75	+1.92
Style	4.71	6.33	+1.62
Citation	3.80	5.70	+1.90
Grounding	2.80	4.71	+1.91
Robustness	9.19	9.19	0.00
Entity Recall	1.72	3.33	+1.61
Judge Avg	4.54	6.12	+1.58
Final Score	3.97	5.80	+1.83

This quantitative discrepancy indicates that researchers cannot evaluate specialized legal RAG systems using out-of-the-box LLM judges. Zheng et al. [57] analyze this exact phenomenon, confirming that LLM evaluators suffer from systematic biases and require detailed, rule-based anchoring to provide reliable scores. The evaluation prompt must mirror the operational constraints of the generative model; otherwise, the metrics will merely reflect the judge's inability to recognize safe, compliant system behaviors.

6.4.3. The Trade-off between Robustness and Style

A clear operational trade-off is revealed by the statistical analysis. Style and Relevance are maximized by Fine-Tuning ($p < 0.001$), whereby the model is forced to adopt the rigorous persona of a legal expert. However, the retrieval mechanism is identified as the primary driver for Robustness and Citation. These distinct benefits are attempted to be merged by the Final configuration, but noise is inherently introduced. The *hallucination potential* of the model is increased by the fine-tuning process. A highly confident tone is learned by the model, even when inaccurate information is generated. The strict factual grounding provided by the retrieved documents is diluted by this acquired confidence. This dilution is directly evidenced by the lower Robustness scores observed in the Final configuration compared to the pure RAG setup. This trade-off must be carefully balanced when enterprise legal assistants are designed, prioritizing either stylistic perfection or absolute factual rigidity.

6.4.4. Parameter Size vs. Technique Effectiveness

Model parameter size is confirmed to remain a dominant predictor of overall performance. The smaller models were consistently outperformed by the massive Qwen-80B architecture across all metrics, regardless of the technique applied. However, a proportionally larger performance delta was generated by Fine-Tuning for the smaller models, specifically Llama-3B and Gemma-4B. This dynamic suggests that superior zero-shot reasoning and instruction-following capabilities are already possessed by larger models. Consequently, Fine-Tuning is utilized merely as a minor stylistic refinement for them.

Conversely, smaller, highly efficient models can be effectively specialized for specific enterprise tasks through supervised training. These compact architectures are successfully taught how to format valid JSON outputs and adopt the required legal tone by the Fine-Tuning process. Nevertheless, a hard infrastructure limit is maintained. As established in the qualitative analysis, large, noisy context windows containing multiple retrieved documents fail to be processed by smaller models. Therefore, a highly precise retrieval pipeline is required when a specialized small model is deployed. If top-tier accuracy cannot be guaranteed by the retriever and excessive noise is fed to a small fine-tuned model, the system will inevitably hallucinate and fail.

6.4.5. Context Overload and Model Scale Implications

The experiments show that model scale directly dictates context processing capabilities. Supplying 20 retrieved chunks creates a heavy cognitive load for the generative layer. Small architectures, such as Llama-3B and Gemma-4B, struggle significantly under this load. They lose track of the prompt instructions, fuse unrelated legal concepts, and ultimately fail to extract the correct answer.

Massive architectures, specifically Qwen-80B, successfully navigate the exact same context window. They extract the relevant facts, ignore the noise, and maintain the required structural format. This discrepancy proves that enterprise deployments face a hard infrastructure constraint. Developers cannot rely on highly efficient, small parameter models if the underlying retrieval corpus requires large, noisy context windows to guarantee high recall. The deployment must either utilize massive models with high inference costs, or it must drastically refine the retrieval step to supply only top-k documents with extreme precision.

6.5. Cost/Performance Tradeoff

A critical dimension of the enterprise deployment is represented by the financial and computational costs associated with utilizing these systems in a production environment. Rather than focusing solely on experimental expenditures, the economic viability must be evaluated through the Total Cost of Ownership (TCO). In recent literature [35], the TCO for Large Language Models is modeled as $C_{Total} = C_0 + C_v$, where C_0 represents the initial infrastructure setup and training costs, and C_v represents the variable operational costs dependent on system usage. A significant dichotomy between the cost structures of the Fine-Tuning and Retrieval-Augmented Generation configurations is highlighted by this framework.

Regarding the Fine-Tuning approach, a substantial initial investment (C_0) is required for the training phase on high-performance computing clusters. Furthermore, if a fine-tuned model is deployed independently, the continuous provisioning of dedicated GPU instances is necessitated. It must be emphasized that this variable maintenance cost (C_v) is economically justifiable only if the system is queried continuously. If the enterprise tool experiences periods of inactivity, the maintenance costs for idle hardware are rendered rapidly unsustainable.

Conversely, the RAG configuration is characterized by a minimal setup cost (C_0) but a significantly higher variable cost (C_v) managed through cloud APIs. Upfront training costs

are avoided, but per-query costs are systematically inflated by the injection of retrieved documents into the prompt. Furthermore, a measurable latency overhead is introduced by both the vector database search and the generation over an expanded context window. This latency must be strictly factored into the evaluation, as slower generation times directly degrade the user experience and increase compute time.

The variable operational costs (C_v) are further dictated by the chosen model scale. An analysis of the AWS Bedrock pricing, summarized in Table 6.10, illustrates an exponential cost increase as model dimensions grow. Base costs are reported as 0.05/0.09 for Gemma-4B and 0.17/0.17 for Llama-3B. As the parameter count increases, the costs escalate to 0.18/0.70 for Qwen-32B and 0.27/0.45 for Gemma-27B. Finally, a peak of 0.18/1.41 is reached by the massive Qwen-80B architecture. It is observed that higher computational budgets are inevitably demanded by larger reasoning models, as additional hidden tokens are generated during their intrinsic thinking phase prior to providing the final answer.

The effectiveness of the company implementation must be critically evaluated against these figures to determine if the performance increase is motivated by the cost escalation. While top-tier accuracy and flawless zero-shot formatting are guaranteed by the Qwen-80B architecture, its output cost is approximately fifteen times higher than that of Gemma-4B and eight times higher than Llama-3B. It is concluded that this exponential cost increase is not strictly justified for routine regulatory queries. If the enterprise query volume is sufficiently high and continuous, the upfront C_0 of fine-tuning a highly efficient, smaller model is rapidly amortized. Acceptable stylistic formatting and factual extraction can be successfully achieved by a specialized small model at a fraction of the API runtime expense and latency, thereby representing the most rational economic tradeoff for Edison.

Table 6.10: AWS Bedrock pricing per million tokens (USD) for the architectures evaluated in the study. Costs are divided into input (context processing) and output (generation) phases.

Model	*Input Cost (USD / 1M tokens)*	*Output Cost (USD / 1M tokens)*
Gemma-4B	0.05	0.09
Llama-3B	0.17	0.17
Qwen-32B	0.18	0.70
Gemma-27B	0.27	0.45
Qwen-80B	0.18	1.41

7 | Conclusions and future developments

This thesis conducted an ablation study to evaluate the performance of open-source Large Language Models (LLMs) in the Italian energy trade market domain. The research aimed to build a reliable enterprise tool for Edison employees. The methodology compared four configurations: Baseline, Retrieval-Augmented Generation (RAG), Fine-Tuning (FT), and a Hybrid approach (RAG+FT). The infrastructure utilized an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance with Amazon Bedrock for RAG orchestration and centralized evaluation. Simultaneously, the system utilized the Leonardo Cineca High-Performance Computing (HPC) cluster to execute QLoRA parameter-efficient fine-tuning.

The study generated a synthetic dataset using Google Gemini 3 Pro to simulate a domain expert. The evaluation framework executed inference by generating five distinct answers per query to measure robustness. An LLM-as-a-judge (Claude 4.5 Opus) evaluated the responses alongside a static Entity Recall metric.

7.1. Summary of Results

The experiments revealed distinct operational trade-offs. RAG successfully grounded the models in factual reality and prevented hallucinations on specific legal queries [29]. FT adapted the stylistic output of the models, teaching them to follow strict formatting and adopt a professional legal persona [21]. However, integrating both techniques exposed a safety tax. Fine-tuned models developed an over-reliance on their parametric memory. They frequently ignored the explicitly provided RAG context and forced an answer even when the retrieved documents lacked the necessary facts [22].

The analysis also showed that parameter scale dictates system capabilities. Massive models like Qwen-80B successfully processed large context windows containing 20 retrieved chunks. Smaller architectures like Llama-3B struggled with context overload but

showed the highest relative improvement from supervised training, positioning them as cost-effective alternatives for specific enterprise tasks.

7.2. Limitations

The implementation contains specific limitations regarding applicability and threats to validity.

First, the evaluation relies heavily on the LLM-as-a-judge methodology. Automated judges exhibit inherent biases, including verbosity bias and a preference for specific structural styles over pure factual substance [57]. While the research implemented strict prompt calibration to mitigate this, the judge remains an imperfect proxy for human legal expertise. Furthermore, the use of synthetic data from an Oracle model means the open-source models learned an approximation of legal reasoning rather than ground-truth human knowledge.

Second, hardware and software constraints limited the study. As a result, the evaluation successfully processed the full testing pipeline for only five out of the ten initially selected models. The HPC environment restricted the fine-tuning of pre-quantized models due to VRAM limitations. Additionally, models with small context windows failed to process the extensive document retrieval required by the RAG pipeline without encountering overflow errors.

Finally, the RAG architecture strictly bounds the performance of the generative model. The system utilizes Amazon Titan v2 and a hybrid Dense and BM25 retrieval pipeline. If this retrieval module fails to extract the correct legal article due to semantic mismatch or chunking errors, the generative model inevitably fails to answer the query correctly, regardless of its reasoning capabilities.

7.3. Future Work

Future research should address these limitations through several key developments.

Organizations must implement a Human-in-the-Loop (HITL) validation phase. Deploying domain experts from Edison to review the generated outputs will properly calibrate the automated metrics and ensure strict legal compliance [17].

Researchers can optimize the training pipeline by utilizing the uniquely indexed dataset created during this study. Applying Curriculum Learning - where the model trains on the highest-quality data before progressing to more complex samples - could improve stylistic

alignment without degrading context reliance.

Finally, engineers should improve the document processing pipeline. Enhancing the semantic chunking process with more granular metadata tagging will allow the models to synthesize macro-topics and compare different versions of abrogated laws effectively. Developers can also expand the knowledge base to include real-time market data or adjacent legal domains, such as civil law, to create a comprehensive enterprise assistant.

Bibliography

[1] Anthropic. Contextual retrieval. *Anthropic Research Blog*, 2024. URL `https://www.anthropic.com/news/contextual-retrieval`. Accessed: 2026-02-17.

[2] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, A. Jones, N. Joseph, B. Mann, D. Nova, et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

[3] R. Bommasani, K. Klyman, S. Longpre, S. Kapoor, N. Masoud, B. Xiong, D. Zhang, and P. Liang. The foundation model transparency index. *arXiv preprint arXiv:2310.12941*, 2023.

[4] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. Legal-bert: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, 2020.

[5] I. Chalkidis, A. Jana, D. Hartung, M. Bommarito, I. Androutsopoulos, D. M. Katz, and N. Aletras. Lexglue: A benchmark dataset for legal language understanding in english. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, 2022.

[6] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[7] L. Chen, S. Li, J. Yan, H. Wang, K. Gunaratna, V. Yadav, Z. Tang, V. Srinivasan, T. Zhou, H. Huang, et al. Alpagasus: Training a better alpaca with fewer data. In *International Conference on Learning Representations (ICLR)*, 2024.

[8] P.-E. Chen, D.-C. Lian, and S.-K. Hsieh. Continual pre-training is (not) what you need in domain adaption. *arXiv preprint arXiv:2504.13603*, 2025.

[9] G. V. Cormack, C. L. Clarke, and S. Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd interna-*

tional ACM SIGIR conference on Research and development in information retrieval, 2009.

[10] G. DeepMind. Gemini 3 pro frontier safety framework report. Technical report, Google DeepMind, November 2025. URL `https://storage.googleapis.com/deepmind-media/gemini/gemini_3_pro_fsf_report.pdf`.

[11] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[12] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

[13] S. Faroz. Domain-adaptive continued pre-training of small language models. *arXiv preprint arXiv:2504.09687*, 2025.

[14] L. Gao, X. Ma, J. Lin, and J. Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[15] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[16] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2 edition, 2019.

[17] F. Gilardi, M. Alizadeh, and M. Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30): e2305016120, 2023.

[18] S. Goyal, C. Baek, J. Z. Kolter, and A. Raghunathan. Context-parametric inversion: Why instruction finetuning can worsen context reliance. In *Advances in Neural Information Processing Systems*, 2024. URL `https://arxiv.org/abs/2410.10796`.

[19] A. Gudibande, E. Wallace, C. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine, and D. Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

[20] J. He et al. Knowledge injection in llms: A review of fine-tuning vs rag. *arXiv preprint arXiv:2401.00000*, 2024.

[21] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and

W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[22] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[23] T. Huang, S. Hu, F. Ilhan, S. F. Tekin, Z. Yahn, Y. Xu, and L. Liu. Safety tax: Safety alignment makes your large reasoning models less reasonable. *arXiv preprint arXiv:2503.00555*, 2025.

[24] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[25] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[26] J. Kasai, K. Sakaguchi, Y. Takahashi, R. L. Bras, A. Asai, X. Yu, D. Radev, N. A. Smith, and K. Choi, Yejin andInui. Realtime qa: What's the answer right now? In *NeurIPS*, 2023.

[27] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint conference on Artificial intelligence*, volume 2, pages 1137–1143. Montreal, Canada, 1995.

[28] R. Koo, M. Lee, et al. Benchmarking cognitive biases in large language models as evaluators. *arXiv preprint arXiv:2311.09766*, 2023.

[29] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[30] H. Li, Y. Chen, Y. Hu, Q. Ai, J. Chen, X. Yang, et al. Lexrag: Benchmarking retrieval-augmented generation in multi-turn legal consultation conversation. *arXiv preprint arXiv:2502.20640*, 2025.

[31] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[32] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, L. Zettlemoyer, and W.-t. Yih. Ra-

dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*, 2023.

[33] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.

[34] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, N. Goyal, A. Asai, H. Wang, et al. Augmented language models: a survey. *arXiv preprint arXiv:2302.08515*, 2023.

[35] N. Mien et al. Assessing economic viability: A comparative analysis of total cost of ownership for domain-adapted large language models. *arXiv preprint arXiv:2404.08850*, 2024.

[36] H. Mousannif. Exploring the frontiers of text segmentation for better rag systems. *Medium Engineering Blog*, 2024.

[37] F. Nan, R. Nallapati, Z. Wang, C. Nogueira, B. Shang, P. Ng, K. McKeown, C. Niu, and B. Xiang. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2727–2733, 2021.

[38] O. Ovadia, E. Menachem, M. Moshkovitz, et al. Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*, 2024.

[39] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[40] G. Reuven et al. Chunkrag: Novel llm-chunk filtering method for rag systems. *arXiv preprint arXiv:2410.19572*, 2024.

[41] W. Saunders, C. Yeh, J. Wu, S. Bills, L. Ouyang, J. Ward, and J. Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

[42] A. W. Services. Amazon titan text embeddings v2: A new state-of-the-art embeddings model on amazon bedrock. `https://aws.amazon.com/about-aws/whats-new/2024/04/amazon-titan-text-embeddings-v2-amazon-bedrock/`, 2024. Accessed: 2026-02-02.

[43] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[44] S. Sheikholeslami, M. K. Rad, A. Girdhar, M. Dowlatshahi, S. Alshari, et al. Ablation programming for machine learning. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5263–5271. IEEE, 2019.

[45] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2018.

[46] H. Soudani, E. Kanoulas, and F. Hasibi. Fine tuning vs. retrieval augmented generation for less popular knowledge. *arXiv preprint arXiv:2403.01432*, 2024.

[47] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[48] The jamovi project. jamovi (version 2.6) [computer software], 2024. URL `https://www.jamovi.org`.

[49] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Reading, Mass., 1977.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, Long Beach, CA, USA, 2017. Curran Associates, Inc.

[51] D. Yan, W. Liu, et al. Rag-studio: Towards in-domain adaptation of retrieval augmented generation through self-alignment. *arXiv preprint arXiv:2409.00000*, 2024.

[52] B. Zhang, G. Tyser, et al. Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture. *arXiv preprint arXiv:2403.01432*, 2024.

[53] T. Zhang, K. Varshney, R. Liu, G. Zheng, et al. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.

[54] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*, 2024.

[55] Y. Zhang, X. Chen, et al. Benchmarking multi-modal retrieval for long documents. *arXiv preprint arXiv:2501.08828*, 2025.

[56] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[57] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, P. Eric, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[58] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

List of Figures

List of Tables