Politecnico di Torino

Politecnico di Torino

Master's Degree in Ingegneria Gestionale

A.A. 2025/2026

Graduation Session March 2026

An Open-Source Multilingual RAG Pipeline for Policy Analysis

Supervisor:
Alessandro Aliberti

Candidate:
Alessandro Buriasco
s334261

Abstract

The analysis of environmental policies usually requires the exploration of a massive multilingual corpus of heterogeneous formats, jurisdictions, and institutional portals with no unified access point. Despite recent rapid progress in Retrieval-Augmented Generation (RAG) pipelines and LLM-assisted document processing, current methods do not completely address multilingual content while ensuring legal compliance and maintaining structured domain taxonomies over an incrementally growing database.

This thesis presents a two-stage, open-source system for the automated data ingestion and natural language querying of 819 climate policy documents in twenty-one different languages, generating a vector knowledge base of around 280,000 semantically structured chunks.

The first stage integrates a traffic-light license validation model, a combination of multilingual heuristic pattern matching, trusted domain boosting, and LLM-based classification via Mistral-Nemo (Ollama), together with recursive directory exploration, targeted file filtering across institutional portals, automated PDF quality gating, and LLM-assisted metadata extraction into a predefined glossary of hazard types, resilience criteria, and other characterizing fields.

The second stage implements a RAG architecture employing hierarchical parent-child chunking, where each child is encoded alongside the textual prefix from its parent segment, allowing the encoder to retain broader document context within local embeddings. The process is driven by a sliding-window semantic boundary detection strategy, context-weighted embeddings generated through the Sentence Transformers framework using *paraphrase-multilingual-mpnet-base-v2* as the multilingual encoder, and ChromaDB-backed retrieval with document diversity enforcement and metadata filtering. Cosine similarity is adopted as the primary retrieval metric due to its direct correspondence with the normalized embedding space, supported by source attribution verification and qualitative assessment on domain-specific policy queries.

The ingestion pipeline achieved a processing success rate exceeding 99%, yielding an average of approximately 350 chunks per document. Results indicate good retrieval effectiveness, although performance varies across document formats and source languages, underlining the intrinsic complexity of multilingual policy retrieval. Performance is evaluated using the RAGAs framework against a curated gold set of domain-specific queries, measuring faithfulness, answer relevancy, context precision, and context recall.

While fine-tuning and periodic retraining cycles remain necessary to achieve greater consistency across the full document spectrum, this model demonstrates

that a fully open-source, locally deployed pipeline can effectively bridge the gap between fragmented policy corpora and structured, queryable domain knowledge.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context: fragmentation of climate policies and the need for structured data

1.1.1 The regulatory landscape of climate resilience

Over the past two decades the international community has built a dense regulatory architecture to address climate change and disaster risk reduction, producing a proliferation of frameworks, strategies, action plans, and guidelines at every level of governance [1, 2, 3, 4]. At the national and sub-national levels this multilateral effort has generated documentation in formats, languages, and structures that vary enormously: National Adaptation Plans (NAPs), national strategies for disaster risk reduction, civil-protection legislation, regional regulations on flood management, local seismic-resilience programmes, vulnerability assessments, and many more. The Intergovernmental Panel on Climate Change, in its Sixth Assessment Report (AR6), noted that this documentary production, while representing a positive signal of institutional commitment, suffers from significant problems of fragmentation, duplication, and inconsistency across governance levels [5]. The 2025 report of the UN Secretary-General on the implementation of the Sendai Framework further highlighted persistent challenges in data gaps, limited inclusivity, and inadequate integration between disaster risk reduction policies and broader sustainable development planning [6].

1.1.2 The problem of information fragmentation

The challenge facing researchers, policy-makers, and practitioners is therefore not a scarcity of information but, on the contrary, its disorganized overabundance. A policy analyst seeking to understand, for example, how different European countries

address flood risk in their respective national plans would need to manually navigate dozens of institutional portals — each with its own structure, publication formats, and access policies — download documents in different languages, extract the relevant information from PDFs that often lack structured metadata, and compare them side by side. This process is slow, error-prone, and fundamentally unscalable: the volume of documentation produced each year far exceeds the capacity for manual analysis of any research team.

Compounding this documentary fragmentation is a problem of *computational accessibility*. The vast majority of normative documentation is not available in machine-readable form. Government portals predominantly publish PDF files — frequently scanned paper documents, reports laid out for print, or protected files that prevent direct text extraction. Many institutional websites use modern JavaScript frameworks (React, Angular, Vue.js, Svelte, Nuxt) that render content dynamically on the client side, making it invisible to traditional web crawlers based on static HTTP requests and requiring browser-automation tools for content retrieval. Even when the text is technically extractable, the absence of a shared taxonomy across institutions from different countries and legal traditions makes systematic comparison extremely laborious. A concept such as *disaster risk reduction* may be rendered differently not only across languages but also across documents in the same language produced by institutions with different terminological conventions — consider, for instance, the vocabulary differences between a United Nations report, a European Commission directive, and a Brazilian federal decree.

Furthermore, many policy documents are published under licensing terms that vary from fully open (Creative Commons, public domain) to restrictive (all rights reserved) or simply unstated. A system that automatically ingests documents from heterogeneous web sources must therefore be able to assess — for each individual document — whether its licence permits reuse, before incorporating it into a downstream dataset. This legal-compliance dimension is almost entirely absent from existing retrieval and document-processing pipelines, which typically assume that the user has already curated a corpus of permissively licensed material.

1.1.3 The opportunity offered by Large Language Models

Recent advances in natural language processing, and in particular the development of Large Language Models (LLMs), open a significant research opportunity to address the challenges described above. LLMs are deep neural networks based on the Transformer architecture [7], trained on internet-scale text corpora, capable of understanding, analysing, and generating natural-language text with a level of competence that, for many tasks, approaches human performance [8]. Their ability to operate multilingually, follow complex instructions, and produce structured output (JSON, tables, classifications) makes them particularly suited to automating

tasks that traditionally required expert human intervention: extracting metadata from unstructured text, thematically classifying documents, answering open-ended questions over specialist documentary corpora.

However, LLMs have a fundamental limitation: the knowledge they memorise in their parameters during pre-training is static, potentially outdated, and not verifiable. When queried on specific facts, models can generate plausible but unsupported assertions — a phenomenon known in the literature as *hallucination* [9]. The Retrieval-Augmented Generation (RAG) paradigm, introduced by Lewis et al. [10], addresses this problem by augmenting the LLM with an external non-parametric memory: before generating a response, the system retrieves from a document index the passages most relevant to the user's question and injects them into the prompt as factual context. The response is thereby grounded in verifiable documentary sources, with the possibility of explicitly attributing each claim to its origin.

This architecture offers significant advantages over traditional model fine-tuning. The knowledge base can be updated incrementally without retraining the model; the sources of answers are traceable and auditable; and the system can operate on specialised domains even with comparatively small models, because domain knowledge resides in the document index rather than in the model parameters. Moreover, RAG naturally supports provenance tracking: each answer can cite the specific documents and passages from which information was drawn, a property of critical importance in policy analysis where every claim must be traceable to an authoritative source.

Nevertheless, applying RAG to the specific domain of climate policies introduces non-trivial requirements that generalist frameworks do not explicitly address. These include robust multilingual support across a corpus that spans documents in English, French, Spanish, and Portuguese as well as less-represented languages such as Turkish, Greek, Norwegian, and German; legal compliance in automated document acquisition, requiring licence verification before every download; the ability to maintain coherent domain-specific taxonomies over an incrementally growing database; and the necessity of handling documents with highly variable levels of structuring, quality, and completeness — from well-formatted official reports to OCR-scanned legacy documents with garbled text.

1.2 Objectives of the thesis

This thesis sets out to design, implement, and evaluate an end-to-end system, entirely based on open-source technologies and executable locally on consumer hardware, for the automated ingestion and natural-language querying of documentation related to climate resilience and disaster risk reduction.

The system operates in two distinct but integrated stages. The first stage — the *data-ingestion pipeline* — is responsible for acquiring documents from heterogeneous web sources, automatically verifying licence and reuse conditions, performing quality control, extracting metadata conforming to a predefined domain taxonomy, and constructing an integrated dataset. The second stage — the *RAG system* — takes the processed documents, splits them into semantically coherent units, indexes them in a vector database, and enables natural-language querying with explicit source attribution.

A deliberate scoping decision shapes the acquisition strategy: rather than operating as an autonomous web crawler, the system works from a *curated seed list* of 249 institutional document references compiled by the research team in collaboration with the project correlatrice Sofia Darbesio. This list collects URLs pointing to government portals, international-organisation repositories, and open-access archives known to host climate and disaster risk reduction policy documentation, and encompasses both direct file links and multi-document portals. Starting from this seed, the pipeline ingested **819 documents** spanning **21 languages**, constructing a vector knowledge base of around 280,000 semantically structured chunks. The choice of a curated seed over unconstrained crawling reflects a practical trade-off: an autonomous crawler would retrieve a large volume of off-domain material, whereas a curated seed list ensures that every document entering the pipeline is at least plausibly relevant to the domain. The system is designed to support future extension with automated seed discovery, as discussed in Section 7.3.2.

The metadata taxonomy underpinning the system is not generic. It has been co-designed with the research team to reflect the conceptual framework of the Sendai Framework for Disaster Risk Reduction and related international agreements, and classifies each document along five dimensions: the type of natural hazard addressed; the geographical scale of application; the key resilience criteria promoted; additional thematic tags; and the document type. This structured classification enables the system to answer not only broad questions but also highly specific queries — for example, identifying which national strategies address flood risk through community-based governance mechanisms, or which guidelines on heatwaves have been issued at the supranational level.

More specifically, the system is designed to satisfy the following six requirements:

R1 Acquisition and legal validation. The pipeline must collect documents from heterogeneous institutional portals, handling both static HTML content and dynamic web pages based on JavaScript frameworks that require client-side rendering. Before each download, the system must automatically verify licence and reuse conditions by combining heuristics based on multilingual regular expressions, a database of trusted domains (government portals, international organisations, open-access archives), and LLM-assisted classification

for ambiguous cases.

R2 Quality control and filtering. Not all downloaded documents are relevant or of sufficient quality for analysis. The system must implement an automated quality gate that filters out posters, press releases, navigation pages, overly short documents, and documents not pertinent to the domain, based on dimensional heuristics (minimum file size, minimum page count), title analysis against a multilingual blacklist, and LLM-based validation of thematic relevance.

R3 Metadata structuring. For each document that passes the quality gate, the system must extract and normalise metadata conforming to the predefined taxonomy. This taxonomy comprises: the official document title, the issuing authority, the document type, the natural-hazard type (Flood, Earthquake, Landslide, Wildfire, Heatwaves, Climate change, Multi-hazard), the geographical level (from Global/Supranational to National), key resilience criteria (Adaptive governance, Health and wellbeing, Active memory, Social interaction and inclusiveness, Socio-economic resilience), and additional free-text tags.

R4 Semantic chunking and vector indexing. Documents must be split into semantically coherent text units, organised in a two-level hierarchy (parent chunks providing broad context, child chunks serving as retrieval units), and indexed in a vector database that supports both semantic-similarity search and filtering on structured metadata.

R5 Natural-language querying with source attribution. The system must allow a user to pose questions in natural language and receive synthesised answers based on the ingested documents, with explicit citation of the sources used. The generative model must produce answers in the language of the question, synthesising information from multiple documents when appropriate.

R6 Reproducibility, updatability, and cloud independence. The entire pipeline must be reproducible on consumer hardware (a single 8 GB GPU or CPU-only machine), without dependencies on proprietary cloud APIs, and must support incremental updating of the documentary database as new documents become available.

The goal is not to build a system that replaces expert analysis, but to provide a tool that drastically reduces the time needed to access specific information within a large, multilingual corpus. A researcher or policy-maker should be able to formulate a question — for example, *"What early-warning strategies for flood risk have been*

implemented in southern European countries?" — and receive within seconds a structured answer with references to the source documents.

1.3 Contributions of this thesis

The main original contributions of this work with respect to the existing literature are the following:

1. **Traffic-light licence validation model.** A three-layer classification component — combining multilingual heuristic pattern matching, a curated trusted-domain database, and LLM-based disambiguation via Mistral-Nemo — that automatically assigns a reuse status (green / amber / red) to each candidate document before download. To the best of our knowledge, no prior RAG pipeline for policy documents addresses the legal-compliance dimension at ingestion time.

2. **End-to-end open-source RAG pipeline for multilingual policy analysis.** A complete, locally deployable pipeline that integrates automated document ingestion, LLM-assisted metadata extraction, hierarchical semantic chunking, hybrid dense–sparse retrieval, and grounded natural-language answer generation into a single reproducible system. Starting from 249 curated seed URLs, the pipeline ingested 819 documents in 21 languages with a processing success rate exceeding 99%, and produced more than 270,000 semantically structured chunks without any dependency on proprietary cloud APIs.

3. **Domain-specific taxonomy co-designed with domain experts.** A metadata schema tailored to the Sendai Framework conceptual vocabulary, covering hazard type, geographical level, resilience criteria, and document type, extracted automatically for each document via LLM-assisted structured output and validated through hybrid merging and fallback repair mechanisms.

4. **Hierarchical parent–child chunking with context-weighted embeddings.** A chunking strategy that pairs sliding-window semantic boundary detection with a two-level parent–child architecture, where each child chunk is encoded alongside a textual prefix from its parent segment, preserving global document context within local embeddings without increasing retrieval latency.

5. **Empirical evaluation on a curated domain-specific gold set.** A RAGAs-based evaluation on a set of domain-specific policy queries, measuring faithfulness, answer relevancy, context precision, and context recall,

providing a reproducible benchmark for future work on multilingual policy RAG systems.

The work was carried out in the context of a research collaboration with the group of Sofia Darbesio (PoliTO), which provided the initial curated dataset of documentary references and the specifications of the domain taxonomy.

1.4 Thesis structure

This thesis is organised into seven chapters whose progression follows the logical architecture of the proposed system, moving from theoretical background through design and implementation to experimental evaluation and conclusions.

Chapter 2 — Background and State of the Art provides the theoretical foundations: the Transformer architecture [7], open-source LLMs (LLaMA, Mistral, Gemma), the RAG paradigm [10], chunking strategies, embedding models, vector databases, dense and sparse retrieval methods, and a survey of related work including GraphRAG [11] and LlamaIndex [12].

Chapter 3 — System Design and Architecture presents the two-stage architecture, quantified requirements analysis, technology stack with design rationale, and the agent-based orchestrator with intent routing.

Chapter 4 — Stage I: Data-Ingestion Pipeline details document acquisition from static and dynamic web sources, traffic-light licence validation, PDF quality gating and corpus deduplication, LLM-assisted metadata extraction, and the incremental dataset update mechanism.

Chapter 5 — Stage II: RAG System describes hierarchical parent–child chunking, semantic boundary detection, context-weighted embedding generation, ChromaDB indexing, retrieval with per-document diversity enforcement, and response generation via Mistral-Nemo running locally under Ollama.

Chapter 6 — Experimental Evaluation reports the quantitative and qualitative assessment of both pipeline stages: ingestion success rate, licence-classification accuracy, metadata-extraction precision, RAGAs retrieval and generation metrics on the gold set, and a case-study analysis of representative policy queries.

Chapter 7 — Conclusions and Future Work summarises the contributions, discusses limitations, and outlines future directions including domain-specific fine-tuning, automated seed discovery, and agentic multi-hop reasoning.

Chapter 2

Background and State of the Art

This chapter provides the theoretical and empirical foundations on which the proposed system rests. Section 2.1 covers the Transformer architecture, the landscape of open-source and locally deployable large language models, and the inference toolchain adopted in this work. Section 2.2 presents the Retrieval-Augmented Generation paradigm in depth, examining chunking strategies, embedding models, vector databases, and retrieval methods. Section 2.3 analyses the distinction between single-step and iterative retrieval, discussing the trade-offs between the two paradigms. Section 2.4 discusses domain-adaptation techniques, focusing on the trade-off between RAG and fine-tuning and on LLM-assisted metadata extraction. Section 2.5 presents the evaluation frameworks for RAG systems, with a focus on the RAGAs suite. Section 2.6 surveys the most relevant related work.

2.1 Large Language Models

2.1.1 From Sequence Models to the Transformer

The ability to process and generate natural language has been a central ambition of artificial intelligence since its earliest formalisation. For most of the history of the field, sequence modelling relied on recurrent neural architectures — Long Short-Term Memory networks (LSTMs) [13] and Gated Recurrent Units (GRUs) [14] — which process tokens one by one in strict left-to-right order. This design imposed two significant limitations. First, the sequential nature of computation prevented parallelisation during training, making large-scale pre-training prohibitively slow. Second, gradient signals propagated across long token spans were systematically attenuated, making it difficult for models to capture long-range syntactic and

semantic dependencies.

Attention mechanisms were introduced as a partial remedy: by allowing the decoder at each step to compute a weighted average over all encoder states, models could selectively focus on distant parts of the input sequence [15]. The decisive step came with [7], who showed in *Attention Is All You Need* that recurrence is not necessary: a model built *entirely* from attention mechanisms not only matches but substantially surpasses recurrent baselines on machine translation benchmarks. This work established the Transformer as the dominant computational substrate for language modelling.

2.1.2 The Transformer Architecture

The Transformer follows an encoder–decoder structure in which both components consist of a stack of $N = 6$ identical layers. Each encoder layer contains two sub-layers: a multi-head self-attention mechanism and a position-wise fully-connected feed-forward network, both equipped with residual connections and layer normalisation [16].

The core computational primitive is *scaled dot-product attention*:

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

where Q, K, and V denote the query, key, and value matrices respectively, and d_k is the dimensionality of the key vectors. The scaling factor $\sqrt{d_k}$ prevents the softmax from entering saturation regions for large values of the dot product. *Multi-head attention* extends this by projecting Q, K, and V onto h different learned subspaces and performing attention in parallel across each:

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\, W^{O}. \tag{2.2}$$

In the original model $h = 8$ attention heads are used with $d_k = d_v = d_{\mathrm{model}}/h = 64$. This formulation allows each head to specialise on different aspects of inter-token relationships: [17] demonstrated through probing experiments that individual heads consistently attend to specific syntactic relations.

Since the architecture contains no recurrence, position information must be injected explicitly through sinusoidal positional encodings:

$$\mathrm{PE}_{(\mathrm{pos},\,2i)} = \sin\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right), \tag{2.3}$$

$$\mathrm{PE}_{(\mathrm{pos},\,2i+1)} = \cos\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right). \tag{2.4}$$

The original Transformer achieved a BLEU score of 28.4 on the WMT 2014 English-to-German translation benchmark, surpassing the best previous ensembles

at a fraction of the training cost [7]. Its intrinsic parallelisability made it the foundation of virtually all subsequent large-scale language models: from encoder-only models such as BERT [18], to decoder-only autoregressive models such as the GPT family [8], to the current generation of open-source instruction-tuned models.

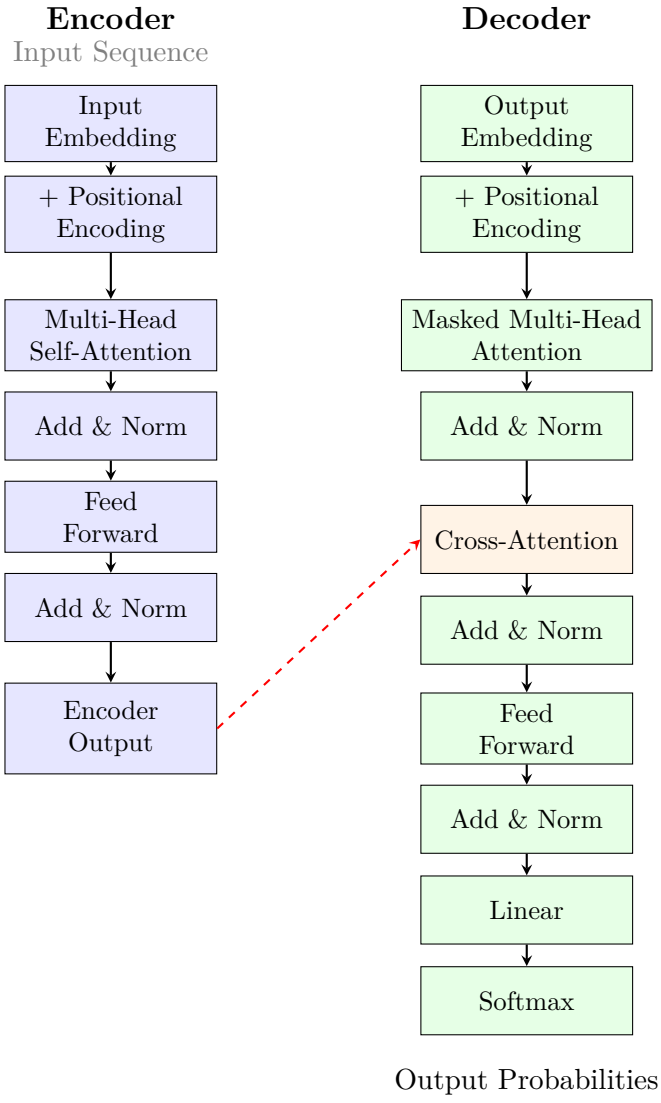

Figure 2.1: High-level architecture of the original Transformer [7]. The encoder (left) maps an input sequence to a sequence of continuous representations; the decoder (right) generates the output sequence autoregressively, attending to both its own previous outputs and the encoder representations through cross-attention.

2.1.3 Open-Source Language Models

The evolution of large language models has proceeded along two trajectories: proprietary models accessible only via cloud API (GPT-4o, Claude 3, Gemini 1.5) and a growing family of *open-weight* models that can be downloaded, run locally, fine-tuned, and redistributed without dependence on external services. This distinction has significant implications for systems that handle sensitive or legally constrained data: a locally deployed model eliminates the need to transmit document content to third-party cloud endpoints, which in the context of institutional policy documentation raises non-trivial privacy and compliance questions.

Pre-training and alignment. The capabilities of open-weight models are the product of a two-phase training regime. In the first phase — *pre-training* — the model is trained on internet-scale text corpora (typically hundreds of billions to several trillion tokens) with a next-token prediction objective. This phase instils broad linguistic competence, factual knowledge, and the ability to generalise across domains and languages. The quality of the pre-training corpus is now recognised as at least as important as raw parameter count: careful data curation, deduplication, and quality filtering allow smaller models to match the performance of larger ones trained on noisier data [19].

The second phase — *alignment* — adapts the pre-trained model to follow instructions, decline harmful requests, and produce responses aligned with human preferences. The dominant technique is *Reinforcement Learning from Human Feedback* (RLHF) [8]: human annotators rank model outputs, a reward model is trained on these preferences, and the language model is fine-tuned via proximal policy optimisation to maximise the learned reward. More recently, *Direct Preference Optimisation* (DPO) has emerged as a simpler alternative that optimises preference alignment directly from annotated pairs without a separate reward model, achieving comparable results with substantially lower training complexity.

Architectural convergence and efficiency innovations. The open-weight landscape has converged on decoder-only Transformer architectures [7] trained autoregressively. Across model families, a set of architectural choices has emerged as the practical standard for efficient inference at scale.

Grouped-Query Attention (GQA), introduced in Mistral 7B [20], reduces the memory footprint of the key–value cache by sharing projections across groups of query heads. This allows a model to maintain most of the representational quality of full Multi-Head Attention while achieving inference throughput approaching the cheaper Multi-Query Attention, making it the default choice in decoder-only models targeting consumer hardware.

Sliding Window Attention (SWA) constrains each token's attention to a fixed

local window of W tokens, reducing per-layer computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \cdot W)$. Information propagates beyond the window through stacked layers with overlapping receptive fields, effectively enabling long-range dependencies without quadratic scaling. This mechanism is particularly relevant for long-document processing, where the effective context length would otherwise become a hard bottleneck.

Rotary Positional Embeddings (RoPE) encode position information by rotating the key and query vectors in the attention computation, rather than adding fixed positional encodings to the input. RoPE allows context-window extension at inference time (through techniques such as `rope_scaling`) without retraining, which is why most recent open-weight models report context windows of 32k to 128k tokens.

Open-weight model families. The LLaMA family [19, 21] established that compute-efficient pre-training on curated data could produce models competitive with much larger counterparts, catalysing a wave of community fine-tunes and domain-specific derivatives. Mistral [20] demonstrated that architectural efficiency gains — specifically GQA and SWA — could close the gap with larger models while maintaining practicality on consumer hardware. The Gemma family [22] contributed further evidence that vocabulary size (256k tokens for Gemma versus the typical 32k of earlier models) and training-data quality are as impactful as parameter count on downstream task performance.

A cross-cutting trend across all these families is the systematic optimisation for consumer-hardware deployment through post-training quantisation in formats such as GGUF, GPTQ, and AWQ, discussed in Section 2.1.4. Mistral-Nemo, the specific model adopted as the generative backbone of this system, builds on the architectural foundation described here; its configuration and the rationale for its selection are detailed in Section 3.2.1.

2.1.4 Local Inference: Quantisation and Ollama

Running large language models on consumer hardware requires reducing the memory footprint of model weights. *Post-training quantisation* (PTQ) is the dominant approach: it converts the original 32-bit floating-point weights to more compact representations — typically 4-bit or 8-bit integers — by mapping each weight to the nearest value in a discrete set. For a model such as Mistral-Nemo (12.2B parameters), 4-bit quantisation reduces memory occupancy from approximately 24 GB to roughly 7 GB, making execution feasible on a single 8 GB consumer GPU or on CPU alone. Quality degradation is generally limited: perplexity increases of 1–5% are typical for 4-bit formats, and 8-bit quantisation narrows this gap further [23]. The GGUF format (GPT-Generated Unified Format) is the current standard for

distributing quantised Transformer models, encoding weights, vocabulary, and inference configuration in a single self-contained binary.

Ollama [24] is an open-source inference manager built on `llama.cpp` [25], a highly optimised C++ library for executing quantised Transformer models on consumer CPUs and GPUs. It exposes a command-line interface for model lifecycle management and an HTTP REST API compatible with the OpenAI standard, allowing existing LLM client code to communicate with local models without modification. Serving models locally eliminates privacy concerns associated with sending potentially sensitive documentation to third-party cloud services, ensures reproducibility independently of external service availability, and allows the complete system to operate in air-gapped environments — properties of direct relevance to pipelines processing institutional documents under varied licensing conditions.

2.2 Retrieval-Augmented Generation

2.2.1 The RAG Pipeline: Retrieve-then-Generate

The Retrieval-Augmented Generation (RAG) paradigm was introduced by [10] to address a fundamental limitation of parametric language models: despite memorising factual knowledge in their weights during pre-training, models can produce unsupported or factually incorrect claims — the phenomenon of *hallucination* [9] — when queried on specific, up-to-date, or domain-specific information.

RAG addresses this by combining a *parametric memory* (the generative model) with a *non-parametric memory* (an external document index) consulted dynamically at inference time. In the original formulation, Lewis et al. propose two variants: RAG-Sequence, where the entire output is conditioned on a single retrieved document marginalised over the top-K candidates; and RAG-Token, where each output token can be conditioned on a different document. Both variants employ a DPR-based retriever [26] and a BART generator [27]; evaluation on three open-domain question-answering benchmarks showed that RAG models produce more specific, diverse, and factually grounded responses than purely parametric baselines.

In current engineering practice, the RAG pipeline has standardised into four sequential phases, illustrated in Figure 2.2:

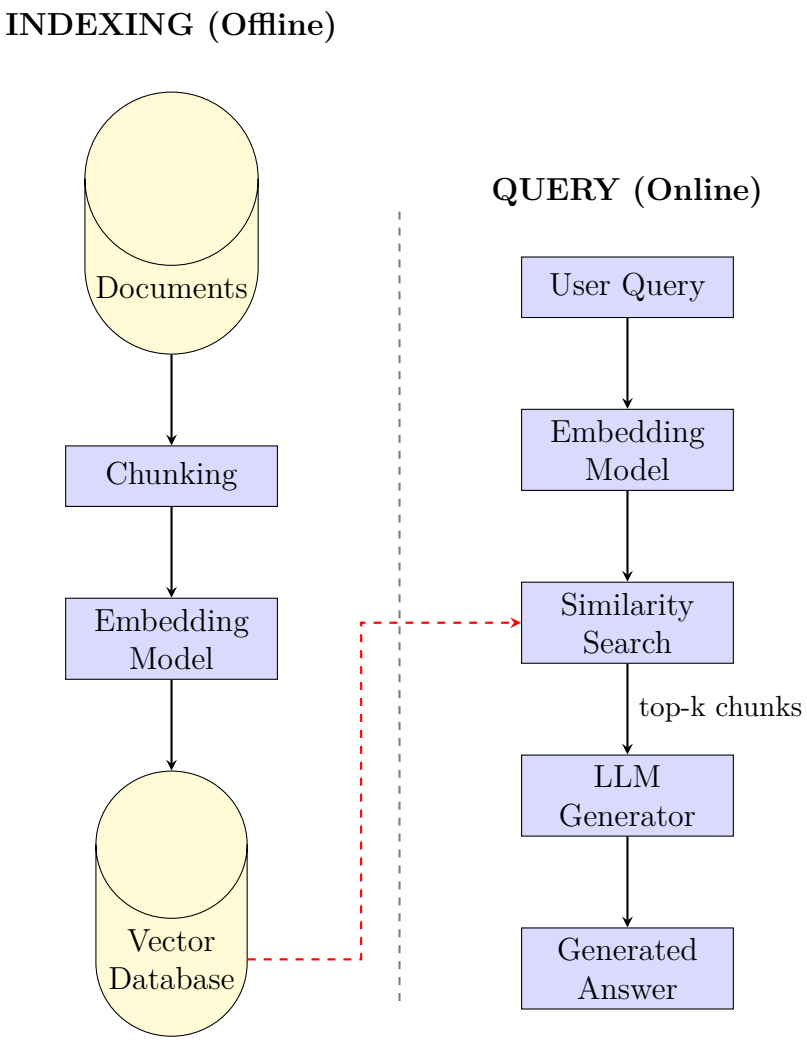

Figure 2.2: The standard RAG pipeline. The *indexing* phase (offline, left) preprocesses documents into chunks, embeds them, and stores vectors in a database. The *query* phase (online, right) embeds the user question, retrieves the top-k most similar chunks, and passes them as context to the LLM for answer generation.

This architecture decouples knowledge from model parameters: the knowledge base can be updated without any retraining simply by updating the document index. Subsequent work has extended the basic paradigm: Self-RAG [28] introduces self-reflection tokens that allow the model to decide when retrieval is necessary and to critique its own output for consistency; CRAG [29] adds a corrective step that triggers re-retrieval when passages are insufficiently relevant; Adaptive RAG [30] dynamically selects the retrieval strategy based on estimated query complexity.

2.2.2 Chunking Strategies: Fixed, Semantic, Hierarchical, and Late Chunking

The granularity and coherence of text chunks is among the most consequential design decisions in a RAG system. The literature distinguishes several main approaches.

Fixed-size chunking divides text into segments of fixed length — typically 200–1 000 tokens — with configurable overlap between consecutive chunks to mitigate information loss at boundaries. Implementations such as `RecursiveCharacterTextSplitter` from LangChain [31] apply a hierarchy of natural split points (paragraphs, sentences, words) before resorting to fixed-character splits. Despite its simplicity, fixed-size chunking remains a strong baseline in settings where document structure is heterogeneous or unpredictable.

Semantic chunking uses embedding models to identify points at which the topic of the text changes significantly. A common implementation computes the cosine similarity between embedding representations of adjacent text windows and places a chunk boundary where similarity drops below a threshold, typically defined relative to the percentile distribution of similarity values across the document [31]. This approach better preserves semantic coherence within chunks at the cost of variable chunk sizes and sensitivity to threshold and window parameters. A practical constraint is that most multilingual embedding models have context windows of 128–512 tokens, making sliding-window variants preferable to single-sentence comparison for detecting macro-level topic transitions.

Hierarchical chunking organises text at two or more levels of granularity. *Parent chunks* (larger, contextually rich) provide the broad context fed to the LLM for response generation, while *child chunks* (smaller, more precise) serve as the actual retrieval unit for similarity matching. This architecture decouples retrieval precision from generation quality, allowing each to be optimised independently. LlamaIndex [12] provides production implementations of this pattern, which has become a standard recommendation for high-quality RAG in specialised domains.

Late chunking [32] is a more recent technique that addresses a fundamental limitation of independently encoded chunks: when a chunk contains a pronoun or elliptical reference, its embedding carries no information about the referent if the referent appears in a different chunk. Rather than encoding each chunk independently, late chunking first processes the *entire document* through the encoder to obtain token-level embeddings carrying global context, and then pools the token embeddings within each pre-determined chunk boundary to obtain chunk-level representations. This substantially improves retrieval for chunks whose meaning depends on surrounding context, without requiring any change to the downstream retrieval pipeline. A practical approximation — injecting a textual prefix drawn from the parent chunk into each child chunk before encoding — achieves similar

contextual enrichment in systems constrained to short-context encoders.

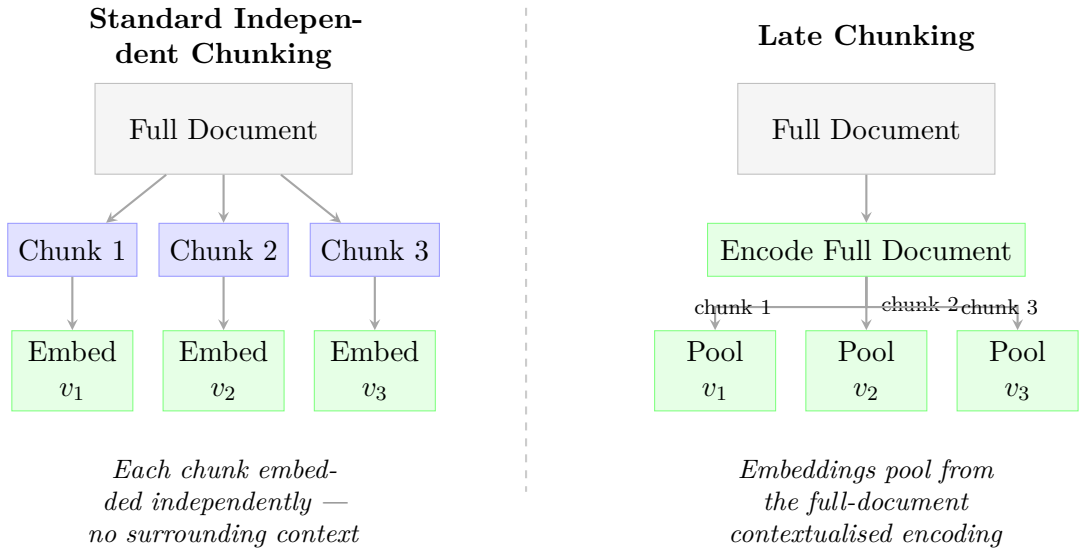

Figure 2.3: Standard independent chunking (left) vs. late chunking [32] (right). In standard chunking each chunk is encoded in isolation, losing cross-chunk context. Late chunking first encodes the full document to produce contextualised token embeddings, then pools the token representations within each chunk boundary so that every chunk vector reflects the surrounding passage.

2.2.3 Embedding Models and Multilingual Representations

Dense text representations and the contrastive paradigm. A text embedding is a fixed-length real-valued vector that encodes the semantic content of a piece of text in a continuous space. The core requirement for retrieval is that vectors of semantically related texts be geometrically proximate, enabling nearest-neighbour search as a proxy for semantic relevance.

Early approaches produced contextual token representations (e.g., BERT [18]) that required task-specific fine-tuning to produce sentence-level vectors. The Sentence Transformers framework [33] established the now-dominant paradigm: fine-tuning a pre-trained Transformer with a *contrastive* objective on sentence pairs. In the Siamese architecture, two identical encoders process a pair of sentences independently; a similarity loss (typically cosine similarity or a margin-based objective) trains the network to place semantically equivalent pairs close in the embedding space and dissimilar pairs far apart. The result is a model that maps

any sentence to a fixed-length vector without task-specific adaptation, enabling similarity search directly in the embedding space.

Multilingual alignment through knowledge distillation. Extending contrastive training to multiple languages requires the model to align semantically equivalent sentences in different languages to nearby vectors, even when no cross-lingual supervision is available for most language pairs. The dominant solution, introduced by [34], is *cross-lingual knowledge distillation*: a high-quality monolingual teacher model (typically trained on English) is used to generate reference embeddings for a large parallel corpus. A multilingual student model is then trained to reproduce the teacher's embedding geometry — placing the French, Spanish, or Turkish translation of an English sentence at the same vector location as the original. Because the student shares its encoder weights across all languages, this alignment generalises to language pairs not seen during distillation, enabling genuinely language-agnostic retrieval from a single model and a single vector index.

The quality of this alignment depends on the coverage and quality of the parallel training data. Benchmarks such as MIRACL [35] and mMARCO [36] have consistently shown that retrieval quality degrades for low-resource languages relative to high-resource ones such as English, French, or Spanish, even for state-of-the-art multilingual models. The degradation is systematic: languages with less parallel training data exhibit weaker cross-lingual alignment, and the gap is not fully closed by scaling the model.

Task-specific adaptation and architectural variants. Standard contrastive training produces a single-role encoder that treats all inputs identically. In retrieval, however, queries and passages have structurally different roles: queries are typically short and underspecified, while passages are long and context-rich. Several architectural strategies address this asymmetry.

Asymmetric training objectives use different loss formulations for query–passage pairs versus passage–passage pairs, producing models where query vectors are closer to relevant passage vectors than to irrelevant ones even when the surface form of the query is very different from the passage text [26].

Task-specific adapters, as introduced in Jina Embeddings v3 [37], attach lightweight LoRA modules to the base encoder that are activated depending on the encoding role (query vs. passage) at inference time. This allows the same set of base weights to serve multiple retrieval tasks without separate model files for each role, at a small memory and latency overhead.

Decoder-based embedding models [38] take a different approach: rather than encoding role through adapters, they prepend a natural-language instruction to each input, steering the decoder's attention towards the retrieval-relevant aspects of the text. This instruction-following mechanism produces state-of-the-art MTEB

scores but at substantially higher inference cost, since each chunk requires a full forward pass through a large autoregressive model.

The context-window constraint in chunked retrieval. A practical dimension that determines model suitability for long-document RAG is the encoder's maximum context window. Models with short windows (128–256 tokens) silently truncate chunks that exceed their capacity, discarding content that may carry the key semantic information for retrieval. Longer windows (512–8 192 tokens) reduce truncation risk but increase per-chunk encoding latency, which compounds over corpora of hundreds of thousands of chunks. The window length therefore interacts directly with the chunking strategy: child chunk sizes must be calibrated to fit within the encoder's capacity including any context prefix, as detailed in Chapter 5.

Table 2.1 positions three representative open-weight models along the dimensions most relevant for multilingual policy retrieval on constrained hardware.

Table 2.1: Comparison of three open-weight multilingual embedding models. MTEB multilingual scores are average nDCG@10 across the MIRACL benchmark languages [35].

Model	Params	Dim.	Context	MIRACL avg.
mpnet-multilingual	278M	768	512 tok	∼64
Jina Embeddings v3	570M	1024	8192 tok	∼69
E5-mistral-7b-instruct	7B	4096	4096 tok	∼72

The table illustrates the quality–efficiency frontier: larger models with wider context windows achieve higher retrieval scores but impose a proportionally higher encoding cost. At the lightweight end, distilled models such as `paraphrase-multilingual-mpnet-base-v2` [34] offer a practical trade-off between multilingual coverage and CPU throughput. At the high-parameter extreme, decoder-based models such as `E5-mistral-7b-instruct` [38] achieve top-tier MTEB performance but require GPU acceleration to index large corpora within a feasible time budget. The operational selection of the embedding model for this system, including the evaluation of three candidates against the hardware constraints of the deployment environment, is detailed in Section 3.2.3.

2.2.4 Vector Databases: Architecture and Trade-offs

A vector database is a data management system optimised for storing, indexing, and retrieving high-dimensional vectors via *approximate nearest-neighbour* (ANN) search. The most widely used indexing algorithm is HNSW (Hierarchical Navigable Small World) [39], which organises vectors in a multi-layer graph enabling sub-linear query time with support for incremental insertions.

The landscape spans a wide range of deployment models. FAISS [40] offers highly optimised ANN implementations designed for large static collections and research-grade use. Weaviate provides full-featured self-hosted deployment with native hybrid retrieval and schema enforcement. Pinecone is a cloud-managed service that eliminates infrastructure overhead. Milvus targets billion-scale distributed deployments.

ChromaDB [41] occupies a different point in this space: a lightweight, open-source vector database designed specifically for RAG development workflows. Its architecture relies on HNSW via `hnswlib`, with disk persistence over SQLite. ChromaDB exposes a minimalist Python API (`add`, `query`, `delete`, `get`, `update`) requiring no infrastructure configuration, and its `where` clause support enables pre-retrieval metadata filtering — restricting searches to documents of a specific hazard type or geographical level — which can substantially improve retrieval precision in structured domain corpora.

2.2.5 Dense, Sparse, and Hybrid Retrieval

Sparse retrieval is based on weighted term-frequency representations, typically TF-IDF or BM25 [42]. BM25 (Best Match 25) extends TF-IDF with two saturation parameters: k_1, controlling term-frequency scaling, and b, controlling document-length normalisation. Sparse retrieval excels at precise keyword matching and is particularly robust for queries containing specific technical terms, acronyms, or named entities that an embedding model may not encode reliably. Its primary limitation is *vocabulary mismatch*: a document discussing the same concept under different terminology will not be retrieved for a query containing only one variant.

Dense retrieval uses learned neural embeddings to represent queries and documents in a shared continuous vector space, so that semantically related text is placed nearby regardless of lexical overlap. Dense Passage Retrieval (DPR) [26] demonstrated that a bi-encoder architecture — where queries and passages are encoded independently and compared via dot product — can substantially outperform BM25 on open-domain question answering when trained on sufficient supervised data. Dense retrieval captures semantic paraphrase and cross-lingual relationships that are invisible to sparse methods, but can introduce false positives through spurious embedding-space proximity.

Hybrid retrieval combines sparse and dense signals to exploit their complementary strengths. The dominant combination strategy is Reciprocal Rank Fusion (RRF) [43], a score-free rank aggregation method:

$$\mathrm{RRF}(d) = \sum_{r} \frac{1}{k + \mathrm{rank}_r(d)}, \tag{2.5}$$

where k is a smoothing constant (typically 60) and $\mathrm{rank}_r(d)$ is the rank of document

d in result list r. RRF is robust to score-scale differences between retrievers, requires no calibration, and has been shown to consistently improve over single-retriever baselines across diverse tasks. ColBERT [44] represents a more deeply integrated alternative, using late interaction between per-token embeddings of query and document to achieve dense-retrieval quality with improved recall at higher storage cost.

In technical and policy documentation — where documents contain specialised terminology, acronyms, and institutional names — sparse retrieval provides a complementary safety net for exact-match recall. Hybrid retrieval with RRF fusion is therefore the dominant choice in high-quality production RAG systems over heterogeneous multilingual corpora.

2.3 Retrieval Interaction Modes: Single-Step vs. Iterative

A fundamental architectural dimension of RAG systems, orthogonal to the choice of retrieval method, concerns *how many times* the retriever and the generative model interact before a final answer is produced.

2.3.1 Single-Step (One-Shot) Retrieval

In the standard RAG formulation [10], retrieval occurs exactly once: the user query is embedded, the top-K chunks are fetched, and the LLM generates a final answer from the static retrieved context. This one-shot architecture is computationally lean — a single embedding computation and a single ANN search are all that is required before generation — and straightforward to implement and debug. It is well suited to *factoid queries* where a single document or passage contains the answer, and its simplicity makes it the dominant choice in production RAG deployments where latency and reproducibility are primary constraints.

The limitations become apparent on complex information needs. If the initial retrieval returns passages that are only partially relevant, or if the answer requires synthesising evidence from documents that are not all nearest neighbours of the original query embedding, the system has no mechanism to recover. Queries that require multi-hop reasoning — *"Which European countries have adopted flood-risk strategies that reference Sendai Framework Priority 3, and what governance mechanisms do they use?"* — typically cannot be resolved from a single retrieval step, because the answer is distributed across multiple documents whose relevance to the query becomes apparent only after partial evidence has been assembled.

2.3.2 Iterative and Agentic Retrieval

Iterative retrieval architectures allow the retriever and the generator to engage in multiple turns before a final answer is produced. The most common pattern is *sequential multi-step retrieval*: the LLM generates a partial answer or an intermediate query reformulation based on an initial retrieval result, which is then used to issue a follow-up retrieval request, and so on until the model determines that sufficient evidence has been gathered.

Self-RAG [28] formalises this through a set of special *reflection tokens* inserted into the model's generation: `[Retrieve]` triggers a new retrieval call; `[IsREL]`, `[IsSUP]`, and `[IsUSE]` allow the model to assess whether a retrieved passage is relevant, whether the generated claim is supported by it, and whether the overall response is useful. Training the model to generate these tokens enables dynamic, on-demand retrieval rather than fixed-frequency polling. On six downstream tasks including fact-checking and long-form question answering, Self-RAG outperformed both retrieval-always baselines and no-retrieval baselines, demonstrating that selective and critiqued retrieval is superior to either extreme.

CRAG (Corrective RAG) [29] adds a lightweight *relevance evaluator* that scores retrieved documents and triggers a web-search fallback when all retrieved passages are assessed as insufficiently relevant. The correction step is applied after the initial retrieval but before generation, functioning as an automated quality gate rather than a full iterative loop.

The most general form of iterative retrieval is *agentic RAG*, in which the LLM is cast as a reasoning agent that can plan a sequence of retrieval actions, call tools, and revise its strategy based on intermediate results. The ReAct framework [45] demonstrated that interleaving reasoning traces (*Thought*) with tool actions (*Action*) and observations (*Observation*) enables LLMs to solve multi-step information retrieval tasks with substantially higher accuracy than generation-only or retrieval-only approaches. IRCoT [46] extends this to chain-of-thought reasoning interleaved with retrieval, showing improvements on multi-hop QA benchmarks including HotpotQA and 2WikiMultiHopQA.

2.3.3 Trade-offs between the Two Paradigms

Table 2.2 summarises the principal trade-offs.

The choice between the two paradigms is ultimately driven by the query complexity distribution of the target application. For a corpus of policy documents where the predominant query type is thematic — *"What does this country's national plan say about flood resilience?"* — single-step retrieval combined with metadata filtering provides a favourable trade-off between answer quality and operational simplicity. Iterative retrieval becomes warranted when queries require cross-document

Table 2.2: Comparison of single-step and iterative retrieval paradigms across key operational dimensions.

Dimension	Single-Step RAG	Iterative / Agentic RAG
Latency	Low (one retrieval + one LLM call)	High (multiple round-trips)
Implementation	Simple, easy to audit	Complex, harder to debug
Factoid queries	Strong	Strong (unnecessary overhead)
Multi-hop queries	Weak	Strong
Error propagation	Contained	Risk of cascade failures
Cost	Low	High (multiple LLM calls)
Reproducibility	High	Lower (stochastic action sequences)

synthesis or follow a dependency chain in which the second retrieval target is identified only after the first has been processed.

2.4 Domain-Adaptation Techniques

2.4.1 RAG vs. Fine-Tuning: Complementary Approaches

Adapting a general-purpose LLM to a specific domain can follow two fundamentally different strategies that differ in where domain knowledge is stored and how it is updated.

Fine-tuning modifies the model's parameters by continuing training on domain-specific data. Parameter-efficient methods — most notably LoRA [47] and its quantised variant QLoRA [23] — make this accessible on consumer hardware by adding a small number of trainable low-rank weight matrices to the frozen pre-trained model, reducing the number of trainable parameters by several orders of magnitude. Fine-tuning embeds domain knowledge directly in model weights, enabling fluent generation in domain-specific style and terminology, but is susceptible to catastrophic forgetting, requires curated supervised training data, and does not provide source attribution.

RAG leaves model weights unchanged and injects domain knowledge at inference time through retrieved context. Its advantages are complementary: the knowledge base can be updated incrementally without any model retraining; each generated claim can be explicitly traced to a source document; and the system operates effectively over specialised domains even with small generative models. The main limitation is that retrieval quality sets an upper bound on generation quality.

In practice, the two approaches are increasingly combined: fine-tuning adapts the model to domain-specific style and output format, while RAG provides dynamic access to current factual content. For systems operating on continuously growing document collections, the updatability advantage of RAG makes it the primary approach, with fine-tuning identified as a future enhancement direction.

2.4.2 LLM-Assisted Metadata Extraction

Extracting structured metadata from unstructured documents has traditionally been approached through Named Entity Recognition (NER) and relation extraction systems trained on annotated corpora. These systems perform well within their training domain but require substantial manual effort to adapt to new schemas or document types.

Large language models offer a significantly more flexible alternative: by specifying the desired extraction schema in natural language as part of the prompt, an LLM can be instructed to extract arbitrary fields from document text without any schema-specific training [48]. The key enabler is chain-of-thought prompting [49], which encourages the model to reason step-by-step before producing a final structured output.

In practice, LLM-assisted metadata extraction requires careful prompt engineering to constrain outputs to valid values, robust parsing and validation logic to handle formatting inconsistencies, and fallback strategies for ambiguous fields. JSON output combined with validation against a schema of enumerated acceptable values has become the standard approach [50]. A recurrent reliability concern is output instability: models may produce malformed JSON, assign out-of-vocabulary values, or leave fields empty for documents in lower-resource languages. The literature recommends a layered mitigation strategy combining JSON repair heuristics, controlled-vocabulary validation, and traditional metadata fallbacks — such as HTML `<meta>` tags and PDF document properties — to fill fields that LLM extraction fails to populate [51, 52].

The approach has been validated in specialised domains. [53] demonstrate that instruction-tuned LLMs can classify sustainability-related documents against multi-dimensional taxonomies with accuracy competitive with supervised classifiers, without annotated training data. [54] show that prompting GPT-class models to extract structured attributes — hazard type, geographical scope, and policy instrument — from a multilingual corpus of climate reports yields extraction quality comparable to fine-tuned NER models at a fraction of the annotation cost. [55] describe an LLM-assisted pipeline for extracting structured records of extreme climate events from web documents, using a predefined schema of location, hazard type, and impact indicators, and highlight both the flexibility of the prompt-based approach and the importance of post-hoc validation to catch hallucinated or

out-of-vocabulary field values.

2.5 Evaluation of RAG Systems

2.5.1 Limitations of Traditional NLP Metrics

Evaluating a RAG system requires assessing two interdependent components simultaneously: the quality of the *retrieved context* and the quality of the *generated answer*. Traditional NLP metrics address only one side at a time. Lexical overlap metrics such as BLEU [56] and ROUGE [57] measure surface-form similarity between the generated output and a reference string, which is a poor proxy for factual correctness when paraphrase is acceptable or when no reference answer exists. Retrieval metrics such as Precision@K and Recall@K assess the relevance of retrieved passages but ignore whether the LLM actually uses them faithfully. Neither family of metrics can detect hallucination — the generation of plausible but unsupported claims — which is the primary failure mode of RAG systems.

2.5.2 The RAGAs Framework

RAGAs (Retrieval-Augmented Generation Assessment) [58] addresses this gap with a set of *reference-free*, LLM-based metrics that jointly assess retrieval quality, answer faithfulness, and alignment between model outputs and supporting evidence. Its reference-free design makes it applicable to domains where no gold-standard answers exist — such as institutional policy documentation — where human annotation is expensive or infeasible.

RAGAs evaluates a RAG system along four complementary dimensions:

- **Faithfulness.** Measures whether all factual claims in the generated answer are inferable from the retrieved context. The LLM-evaluator decomposes the answer into atomic claims and verifies each against the retrieved passages; the score is the fraction of claims that are supported. High faithfulness indicates low hallucination.

- **Answer Relevancy.** Measures how well the answer addresses the user's original question. The evaluator generates multiple paraphrases of questions that the answer would imply and computes the mean cosine similarity between their embeddings and the original question embedding. High relevancy indicates that the answer is on-topic and complete.

- **Context Precision.** Measures the fraction of retrieved chunks that are actually relevant to the query. A high-precision retriever returns few irrelevant

passages, reducing the risk of the LLM being distracted or misled by off-topic content.

- **Context Recall.** Measures the fraction of relevant information in the gold-standard answer that is covered by the retrieved context. High recall indicates that the retriever has surfaced sufficient evidence for a complete answer.

- **Answer Correctness.** A composite metric that combines semantic similarity between the generated answer and the reference answer with claim-level coverage. The semantic component is the cosine similarity between the `mpnet` embeddings of the two texts; the claim-coverage component decomposes the ground truth into atomic claims and verifies how many are present in or implied by the generated answer. The final score is their equally weighted average:

$$\text{Correctness} = 0.5 \cdot \text{SemanticSim}(\text{answer, GT}) + 0.5 \cdot \frac{\text{matched claims}}{\text{GT claims}}. \quad (2.6)$$

  Answer Correctness requires a ground-truth reference and is therefore only computable for the subset of gold-set queries that carry a manually written reference answer.

Together, these five metrics allow a practitioner to diagnose the primary failure mode of a RAG system: low faithfulness points to hallucination in the generative component; low answer relevancy to prompt or instruction issues; low context precision to retrieval noise; low context recall to retrieval gaps; and low answer correctness to a fundamental mismatch between generated and reference content.

2.5.3 Benchmarks for RAG Evaluation

Several datasets have been adopted as standard benchmarks for RAG evaluation.

WikiEval provides a set of factual questions grounded in post-2022 Wikipedia content — after the knowledge cutoff of most pre-trained models — ensuring that a RAG system cannot answer correctly from parametric memory alone. It provides a clean setting for evaluating single-hop factual retrieval.

HotpotQA [59] is specifically designed for multi-hop reasoning: each question requires synthesising information from exactly two Wikipedia articles, and the dataset provides sentence-level supporting facts for evaluation of the reasoning chain. HotpotQA defines two question types — *bridge* questions (where an intermediate entity connects two facts) and *comparison* questions (where attributes of two entities must be compared) — that stress-test iterative and agentic retrieval systems in complementary ways. It has become the standard benchmark for multi-hop RAG comparison.

MIRACL [35] covers retrieval across 18 languages and provides nDCG@10 scores that allow direct comparison of multilingual embedding models. It is particularly relevant for systems operating on heterogeneous multilingual corpora, where retrieval quality differences across language families are a primary concern.

BEIR [60] is a heterogeneous retrieval benchmark covering 18 datasets across diverse domains (biomedical, financial, legal, Wikipedia). It enables zero-shot evaluation of retrieval models, testing generalisation to domain-specific vocabularies not seen during training.

2.6 Related Work

2.6.1 Enterprise and Structured RAG Systems

GraphRAG [11], proposed by Microsoft Research, addresses a known limitation of flat vector retrieval: its inability to answer queries requiring synthesis across multiple documents or entity-relationship traversal. GraphRAG augments standard RAG by extracting a knowledge graph from the document corpus, organising graph nodes into hierarchical communities via the Leiden algorithm [61], and generating natural-language summaries through a bottom-up hierarchical summarisation process. Evaluated on news and podcast transcript corpora, GraphRAG reported substantial improvements over dense-vector baselines on *sensemaking* queries — the category of analytical policy queries most challenging for flat retrieval. The graph-construction cost and the dependency on a powerful external LLM for summarisation make GraphRAG impractical on the constrained hardware targeted by this thesis; flat hierarchical retrieval with metadata filtering is adopted as a more deployable alternative.

RAPTOR [62] proposes recursive summarisation of related chunk clusters, indexing both the original chunks and their summaries to enable retrieval at multiple levels of abstraction. On question-answering benchmarks RAPTOR outperformed both standard RAG and other tree-based methods, demonstrating that hierarchical summarisation is a viable alternative to graph construction for multi-document synthesis. The approach is complementary to the parent–child architecture adopted here and represents a natural extension for future work.

2.6.2 Automated Document-Processing Pipelines

Docling [63], developed by IBM Research, is an open-source library for high-fidelity document conversion targeting RAG ingestion. It provides layout-aware PDF parsing, table extraction, and structured export to Markdown or JSON, and has been shown to improve downstream RAG performance by preserving document structure that plain-text extraction discards. The system described in this thesis

uses PyPDF2 with an EasyOCR fallback rather than Docling, prioritising robustness on scanned documents and multilingual OCR coverage over layout fidelity.

[53] describe CHATREPORT, an LLM-based pipeline for analysing corporate sustainability reports against multi-dimensional taxonomies, demonstrating that prompt-based extraction can match supervised classifier performance at scale without annotated training data. [64] extend this line of work by combining ontology-defined schemas with LLM extraction, reducing out-of-vocabulary field assignments and improving inter-document consistency. Both systems operate on well-structured corporate reports in a single language; the pipeline presented here addresses the substantially harder problem of heterogeneous multilingual institutional documentation with no assumed structural regularity.

2.6.3 Multilingual Information Retrieval for Policy Documents

MIRACL [35] and mMARCO [36] have consistently shown that retrieval quality degrades for low-resource languages relative to high-resource ones such as English, French, or Spanish, even for state-of-the-art multilingual models. For less-represented languages such as Turkish, Greek, or Norwegian — all present in the corpus assembled in this thesis — the quality gap between sparse and dense retrieval narrows, making hybrid approaches less likely to degrade relative to either component individually.

[54] evaluate GPT-class models on structured attribute extraction from a multilingual corpus of climate reports, reporting that prompting with explicit field definitions yields extraction accuracy comparable to fine-tuned NER models for high-resource languages and acceptable performance for lower-resource ones. This finding directly validates the LLM-assisted metadata extraction strategy adopted in Stage I of this thesis.

The EU Publications Office Knowledge Graph [65] provides a structured representation of EU legislative documentation through EUR-Lex entity linking, but operates exclusively on the standardised EUR-Lex format. The system described in this thesis addresses a complementary and more general problem: the highly heterogeneous documentation produced by dozens of national and international institutions across multiple continents, in formats and metadata schemas that vary enormously or are entirely absent.

The following chapter translates the theoretical foundations reviewed here into a concrete system design, formalising the functional and non-functional requirements and presenting the two-stage pipeline architecture.

Chapter 3

System Design and Architecture

This chapter translates the problem context established in Chapter 1 and the theoretical foundations reviewed in Chapter 2 into a concrete system design. Section 3.1 introduces the high-level two-stage pipeline and derives the six functional requirements directly from its structural decomposition, showing at which stage each requirement is satisfied. Section 3.2 documents the technology stack and motivates every design choice. Section 3.3 describes the orchestrator that coordinates all pipeline phases. Section 3.4 presents the Streamlit-based user interface.

3.1 System Overview and Requirements

3.1.1 High-Level Architecture: Two-Stage Pipeline

The system is structured as a two-stage pipeline whose logical decomposition reflects a fundamental separation of concerns, illustrated in Figure 3.1.

Stage I — Data-Ingestion Pipeline transforms a curated list of 249 institutional web references into a clean, legally validated, metadata-enriched document corpus. It is responsible for all decisions about *what* enters the system: which documents are legally reusable, which are topically relevant, and how their content should be structured and labelled. The stage executes four sequential phases — licence analysis (Phase 1), document acquisition (Phase 2), optional recursive portal exploration (Phase 2.5), and LLM-assisted metadata extraction (Phase 3) — plus a cross-cutting quality-control step (Phase 6). Its output is a normalised Excel dataset (`Estrazione Finale`) in which each row represents one document enriched with all extracted metadata fields and the local path to the downloaded content.

Stage II — RAG System (Phase 4) makes the processed corpus queryable in

natural language. It performs hierarchical parent–child chunking of document texts, generates context-weighted embeddings, and indexes chunk vectors in ChromaDB. At query time it retrieves the most relevant chunks via cosine similarity search with metadata pre-filtering and feeds the retrieved context to Mistral-Nemo via Ollama to generate a grounded, source-attributed answer. An optional BM25 sparse index supports hybrid retrieval experiments.

The two stages are connected through a shared Excel state file (`Analisi Licenze`): only documents that have completed the full ingestion pipeline are admitted to the embedding phase.

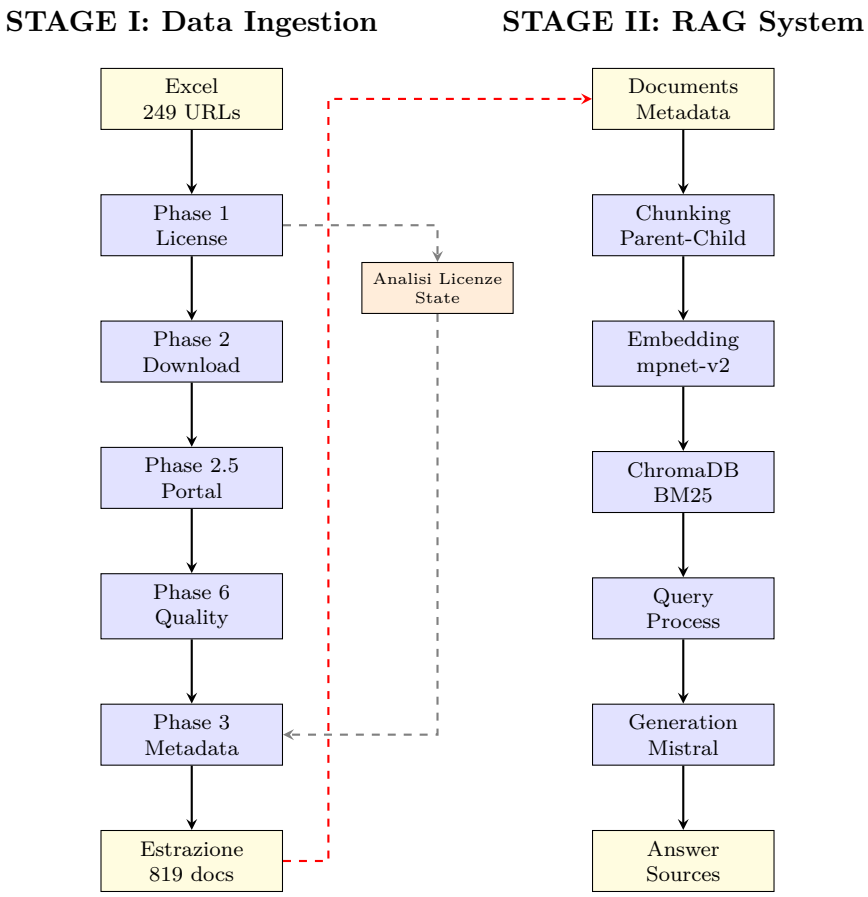

Figure 3.1: High-level overview of the two-stage pipeline. Stage I (left half) handles all data-acquisition and structuring concerns; Stage II (right half) implements the RAG retrieval and generation stack. The shared state file `Analisi Licenze` ensures idempotent, resumable execution across sessions.

3.1.2 Requirements as Pipeline Checkpoints

Rather than listing requirements in isolation, this section maps each of the six requirements introduced in Section 1.2 to the specific pipeline component responsible for satisfying it. Table 3.1 summarises the correspondence.

Table 3.1: Mapping of the six system requirements to the pipeline phases and implementation components that satisfy them.

Req.	Description	Responsible phase / component
R1	Acquisition and legal validation	Phase 1 (`LicenseAnalysisAgent`): three-layer licence cascade. Phase 2 (`DownloadAgent`, Selenium): two-tier file retrieval.
R2	Quality control and filtering	Phase 6 (`run_quality_check_phase`): size, page-count, deduplication. Phase 2 (`valida_rilevanza_documento_llm`): pre-download LLM gate.
R3	Metadata structuring	Phase 3 (`MetadataExtractionAgent`): 12-field Sendai-taxonomy JSON extraction.
R4	Chunking and vector indexing	Phase 4 (`AdvancedChunker`): semantic parent–child splitting. Phase 4 (`EmbeddingAgent`): mpnet encoding, ChromaDB HNSW indexing.
R5	NL querying with attribution	Phase 4 (`query_rag`, `rag_answer`): hybrid retrieval, source-cited generation.
R6	Reproducibility, updatability, open source	Orchestrator (`carica_o_inizializza_stato`): idempotent state file. SHA-256 content-hash cache. Version-pinned model identifiers.

R1 is addressed by two components in sequence. `LicenseAnalysisAgent` classifies each candidate URL before any file is downloaded through a three-layer cascade: trusted-domain lookup, multilingual heuristic pattern matching, and Mistral-Nemo-based LLM disambiguation. Only green-labelled URLs are forwarded to `DownloadAgent`, which handles retrieval across both static HTTP and JavaScript-rendered content.

R2 is implemented in Phase 6 through size and page-count thresholds, multilingual title blacklisting, and three-pass SHA-256-based deduplication. An additional

quality gate is integrated into `DownloadAgent.valida_rilevanza_documento_llm()`, which screens candidate documents *before* download by passing the link text and surrounding HTML context to Mistral-Nemo with a YES/NO classification prompt.

R3 is the responsibility of `the MetadataExtractionAgent` in Phase 3. A structured Mistral-Nemo prompt extracts twelve fields aligned with the Sendai taxonomy; output instability is mitigated through JSON repair, controlled-vocabulary validation, deterministic keyword fallback, and traditional HTML/PDF metadata extraction.

R4 is implemented by `AdvancedChunker` and `EmbeddingAgent` in Phase 4. Sliding-window semantic boundary detection divides documents into parent segments; fixed-window splitting within each parent produces child chunks as retrieval units. Each child is encoded with a textual prefix drawn from its parent to approximate late chunking. Chunk vectors and metadata are stored in ChromaDB with HNSW indexing and cosine distance.

R5 is provided by `query_rag` and `rag_answer`, which embed the user query, retrieve the top-K most similar chunks with optional metadata filtering, format the context, and invoke Mistral-Nemo in streaming mode. Every answer includes explicit source attribution citing the `Document_Code` and local file path.

R6 is enforced by: (a) the persistent state file `Analisi Licenze`, which records processing status for each document and allows any phase to resume after interruption; (b) a SHA-256 content-hash cache (`_embedded_docs_cache.pkl`) that prevents re-embedding unchanged documents; and (c) version-pinned model identifiers (`mistral-nemo:latest`, `paraphrase-multilingual-mpnet-base-v2`) stored alongside every generated artefact.

3.1.3 Non-Functional Constraints

Scalability. The pipeline must process several hundred documents without manual intervention. Parallelism is available in Phase 1 via a `ThreadPoolExecutor` with configurable `MAX_WORKERS_ANALYSIS = 3`. However, the system runs on a virtual machine provided by Politecnico di Torino with limited RAM and single-GPU resources shared across concurrent users. To operate reliably within these constraints, all worker pools are forced to `MAX_WORKERS = 1` throughout the pipeline: licence analysis, download, metadata extraction, and embedding all execute sequentially in the current deployment. The configured upper bounds remain as code constants to allow straightforward parallelism re-enabling on more capable hardware.

Reproducibility. Every result is traceable to a specific document, a specific processing step, and a specific model configuration through the state file, the hash cache, and version-pinned model identifiers.

Updatability. When new document references are added to the source workbook, the orchestrator detects them on the next invocation and appends new rows with status `"No"` without disturbing already-processed entries.

3.2 Technology Stack and Design Rationale

3.2.1 Ollama for Local LLM Inference

Ollama [24] serves as the unified inference backend for all LLM calls. The entire pipeline was developed and executed on a virtual machine provided by Politecnico di Torino, an environment in which cloud API dependencies are undesirable for reasons of both cost and data privacy regarding institutional policy documentation. Ollama's local serving model eliminates both concerns: all inference runs at `http://localhost:11434` and no document content is transmitted to external services.

The primary model is **Mistral-Nemo:latest** (12.2B parameters, 128k-token context window, Apache 2.0 licence, 4-bit GGUF quantisation at approximately 7 GB VRAM). It serves four distinct roles in the pipeline:

1. **Phase 1 — Licence classification.** For URLs unresolvable by trusted-domain lookup or heuristics, Mistral-Nemo receives a composite HTML excerpt (first and last 1 000 chars + licence-keyword context window) and returns `{"decisione":"VERDE/ROSSO/GIALLO","licenza":"..."}` at `temperature: 0.1`, 180-second timeout.

2. **Phase 2 — Pre-download relevance gate.** `valida_rilevanza_documento_llm()` submits the candidate title and surrounding HTML to Mistral-Nemo for a YES/NO classification against the DRR domain, with `num_ctx: 512` and a 5-second timeout.

3. **Phase 3 — Metadata extraction.** Mistral-Nemo processes the first 10 000 characters of pre-cleaned document text and returns a 12-field JSON object conforming to the Sendai taxonomy, at `temperature: 0.08` and an 800-second global timeout.

4. **Phase 4 — Answer generation.** At query time Mistral-Nemo receives the formatted retrieval context (up to 10 top-K chunks with source attribution) and generates a grounded answer in streaming mode.

A second model, **llama3.1:8b**, is used exclusively in Phase 4 for the RAGAs evaluation step (`run_ragas_evaluation`), where it acts as the LLM judge computing faithfulness, answer relevancy, context precision, and context recall on

gold-set queries. Separating the answer-generation model (Mistral-Nemo) from the evaluation judge (Llama 3.1:8b) avoids the self-evaluation bias that arises when the same model scores its own outputs.

Since only one model can reside in Ollama's memory at a time on the virtual machine, all Mistral-Nemo calls (Phases 1–4) are completed before the Llama 3.1 judge is activated for evaluation.

3.2.2 ChromaDB as Vector Store

ChromaDB [41] is selected as the vector database for Stage II. Three properties motivate the choice.

Disk persistence. `PersistentClient` writes the HNSW index and all metadata to `./chroma_db_rag` after each embedding batch, enabling cross-session resumption. The SHA-256 content-hash cache (`_embedded_docs_cache.pkl`) provides an additional deduplication layer that prevents re-embedding unchanged documents.

Metadata-filtered pre-retrieval. ChromaDB's `where` clause restricts ANN search to a sub-index defined by `hazard`, `geographical_level`, `document_type`, and `language` before computing similarity scores. This is essential for structured domain retrieval: a query about wildfire management at national level must not retrieve flood guidance from international organisations. The collection `policy_documents` uses cosine distance, consistent with `normalize_embeddings=True` in the encoder.

Minimalist deployment. ChromaDB installs as a Python package with no external infrastructure, compatible with the `pysqlite3-binary` fix needed on the PoliTO virtual machine.

3.2.3 Sentence Transformers and Embedding Model Selection

The embedding component uses the `sentence-transformers` [33] library. Three candidate models were evaluated in order of priority: Jina Embeddings v3 as the first choice, `paraphrase-multilingual-mpnet-base-v2` as the active model, and `paraphrase-multilingual-MiniLM-L12-v2` as the reserve fallback. The theoretical properties of these models are surveyed in Section 2.2.3; the discussion here focuses on the hardware-driven selection rationale.

Jina Embeddings v3 [37] was the first-choice candidate owing to its 8 192-token context window and state-of-the-art MTEB multilingual scores. On the PoliTO virtual machine, however, loading Jina v3 alongside a 4-bit Mistral-Nemo instance (7 GB VRAM) exhausts available GPU memory, and CPU-only inference at 570M parameters produces encoding throughput incompatible with indexing

360,000+ chunks in a single session. The model is therefore disabled in the current deployment but the codebase is structured so that it can be re-enabled by a single configuration change when sufficient GPU resources are available.

paraphrase-multilingual-MiniLM-L12-v2 [34] was the original fallback: at 118M parameters it offers very fast CPU throughput, but its 128-token context window caused systematic tail truncation on child chunks of 1 200 characters, degrading retrieval quality on longer policy passages.

paraphrase-multilingual-mpnet-base-v2 [34] is the active model. Its 512-token context window accommodates child chunks up to 1 200 characters including the metadata context prefix, its 768-dimensional output space provides finer semantic resolution than MiniLM, and at 278M parameters it fits comfortably in the memory budget alongside Mistral-Nemo. The chunking parameters in Stage II were explicitly calibrated to this window, ensuring that no chunk is silently truncated during encoding. Table **??** summarises the three candidates and the reason for each outcome.

latex

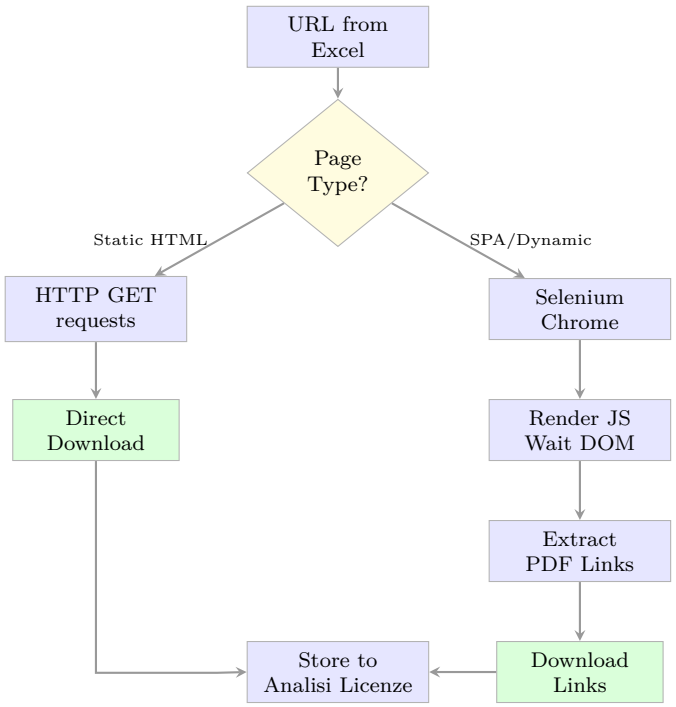

Figure 3.2: Two-tier document acquisition strategy. Tier 1 (left) handles direct HTTP downloads; Tier 2 (right) activates Selenium when the page requires JavaScript rendering. Both paths converge on the shared state file `Analisi Licenze`.

Table 3.2 summarises the principal open-source libraries used across all pipeline stages, grouped by functional role.

Table 3.2: Principal open-source libraries in the system stack, grouped by functional role. All components run locally without cloud dependencies.

Role	Library / Tool	Function
Web acquisition	`requests`, Selenium	Static and JS-rendered page retrieval
Document parsing	`PyPDF2`, EasyOCR	PDF text extraction and OCR fallback
	`python-docx/pptx`, `pandas`	DOCX, PPTX, Excel extraction
HTML cleaning	`BeautifulSoup4`	Content extraction and boilerplate removal
Fuzzy matching	`rapidfuzz`	Title-based link selection on portal pages
Embeddings	`sentence-transformers`	Multilingual chunk encoding (mpnet-v2)
Vector store	`chromadb`	HNSW index with metadata-filtered retrieval
Sparse retrieval	`rank-bm25`	BM25Okapi keyword index
State persistence	`openpyxl`, `pickle`	Excel state file and SHA-256 hash cache
LLM inference	Ollama REST API	Mistral-Nemo (pipeline) and Llama 3.1:8b (judge)
Web UI	`streamlit`	Browser-based query interface

3.3 The Orchestrator: Phase Menu and Intent Routing

All pipeline phases are coordinated by a persistent orchestrator that loads or initialises the shared processing state on startup and then exposes a command interface through which the operator can invoke individual phases.

3.3.1 State Initialisation and Incremental Synchronisation

On startup the orchestrator reads the source Excel workbook, compares it against the persistent state file `Analisi Licenze`, and appends any new URL rows with uninitialised status flags. If no state file exists it is created from scratch with all required column groups covering Phase 1 through Phase 4 outputs and a `Document_Code` primary key. This logic directly implements requirement R6 (updatability): new document references added to the source workbook are detected and ingested on the next orchestrator invocation without disturbing the processing state of already-completed entries. The schema of the state file and its per-phase columns are described in full in Chapter 4.

3.3.2 Phase Menu and Command Interface

The orchestrator exposes six pipeline phases through an interactive menu. Each phase operates *idempotently*: it filters the state DataFrame to process only rows whose status flag is `"No"`, so that re-running a phase after an interruption resumes from the first unprocessed entry without duplicating work.

- **Phase 1 — Licence Analysis.** Classifies each candidate URL through the three-layer licence cascade and records the traffic-light outcome in the state file.
- **Phase 2 — Document Download.** Retrieves documents from approved URLs using the two-tier static-HTTP / Selenium acquisition strategy, applies the LLM-based relevance pre-screen, and saves each file with a `Document_Code` prefix.
- **Phase 2.5 — Recursive Portal Exploration.** Expands the URL list by recursively visiting portal pages classified as DIRECTORY and probing REST/JSON catalogue endpoints.
- **Phase 3 — Metadata Extraction.** Extracts text from each downloaded document, submits it to Mistral-Nemo for structured metadata extraction against the domain taxonomy, and validates all fields against controlled vocabularies. Produces `Estrazione Finale`.

- **Phase 4 — Embedding and RAG.** Chunks documents hierarchically, embeds child chunks with context-weighted mpnet embeddings, indexes them in ChromaDB, and exposes the full RAG query and evaluation interface.

- **Phase 6 — Quality Check.** Applies size and page-count thresholds and three-pass SHA-256 deduplication, removing low-quality or duplicate files from both the filesystem and the state DataFrame atomically.

3.3.3 LLM-Based Intent Router

In addition to numeric selection, the orchestrator accepts free-text natural-language commands. Non-numeric input is passed to an intent-routing function that submits a structured prompt to Mistral-Nemo at `temperature 0.0`, instructing the model to map the user's phrase to the corresponding phase digit. If no valid digit is extracted within a 30-second timeout, the loop re-prompts the user. This makes natural-language control a convenience layer rather than a critical dependency: the pipeline is fully operable through numeric selection alone.

3.4 Streamlit User Interface

The command-line orchestrator is complemented by a browser-based query interface built with Streamlit [66], illustrated in Figure 5.3. The interface exposes the full RAG retrieval and generation stack to non-technical users without requiring any command-line interaction.

The application is structured around a clean separation between backend and frontend. The backend module (`rag_engine.py`) encapsulates all retrieval and generation logic and can be imported and tested independently of the Streamlit runtime. The frontend re-executes on each user interaction, maintaining conversation state across turns within a session.

The interface provides three functional areas. A **query area** accepts free-text questions in any of the 21 supported corpus languages. A **sidebar control panel** exposes the principal retrieval parameters — top-k, similarity threshold, and metadata filters for hazard type, geographical level, and language — allowing analysts to scope queries to specific document subsets without modifying the underlying index. A **results area** renders the generated answer alongside an expandable source-attribution panel that lists each retrieved passage with its similarity score, provenance metadata, and a text excerpt. Similarity scores are colour-coded into three confidence tiers to provide immediate visual feedback on retrieval quality. A conversation history accumulates question–answer pairs for the duration of the session.

Since both the Streamlit server and the Ollama inference endpoint run locally, no document content is transmitted outside the machine during a query session, satisfying the data-governance requirement established in Section 3.1.3.

The following two chapters describe the implementation of Stage I (Chapter 4) and Stage II (Chapter 5) in detail, proceeding from the licence-analysis phase through metadata extraction, hierarchical chunking, and the full RAG retrieval and generation pipeline.

Chapter 4

Stage I: Data-Ingestion Pipeline

Stage I transforms the raw list of 249 institutional seed URLs into a clean, legally validated, metadata-enriched document corpus ready for vectorisation. It is composed of four sequential phases — licence analysis (Phase 1), document acquisition (Phase 2), recursive portal exploration (Phase 2.5), and metadata extraction (Phase 3) — plus a cross-cutting quality-control gate (Phase 6) that is invoked before metadata extraction. This chapter describes each phase in detail, following the data from first HTTP contact to the structured `Estrazione Finale` output file that feeds Stage II.

4.1 Phase 1: Licence Analysis

4.1.1 Motivation and Three-Layer Architecture

The most fundamental constraint on any corpus built from public web sources is legal reusability. Institutional portals in the DRR domain use a wide variety of licence declarations, ranging from explicit Creative Commons statements to vague "all rights reserved" footers, across nine languages. Manual review of 249 URLs is impractical and brittle; an automated three-layer cascade was designed instead.

The three layers operate in decreasing order of certainty: a trusted-domain lookup, a multilingual heuristic pattern-matching step, and an LLM-based disambiguation pass for the cases neither prior layer can resolve. The output for every URL is a traffic-light classification: **VERDE** (open, usable), **GIALLO** (ambiguous, manual review recommended), or **ROSSO** (restrictive, excluded from download).

4.1.2 Layer 1: Trusted-Domain Lookup and Layer 2: Pattern Matching with LLM Override

Layer 1 — Trusted-domain lookup. `verifica_dominio_fidato()` maintains a curated whitelist of institutional domain suffixes — United Nations bodies (`.un.org`, `.undp.org`, `.undrr.org`), European Commission, World Bank, and national open-data portals — each associated with a default licence and a confidence boost score. When the URL matches an entry, the score is incremented and a VERDE candidate is returned without any network call. For direct binary files from trusted domains, no HTML is fetched at all: the trusted-domain signal alone is sufficient.

Layer 2 — Multilingual pattern matching. For URLs not resolved by Layer 1, `analizza_licenza_avanzata()` fetches the page and scans it with `BeautifulSoup`. It inspects `<meta name="license">` and `<link rel="license">` elements first, assigning weighted scores of +30 and +25 respectively. It then applies a compiled dictionary of regex patterns covering eight languages, each associated with a weight proportional to the specificity of the licence term. Permissive patterns — Creative Commons variants (CC BY, CC BY-SA, CC BY-NC), Open Government Licence (OGL), public domain, and reproduction-authorised clauses in all supported languages — push the score above the VERDE threshold of 20. Purely restrictive patterns contribute no positive weight, leaving the URL at GIALLO.

In practice, Layer 2 was originally the primary decision mechanism. A systematic re-analysis of the pattern-matching outputs by the LLM in Layer 3 revealed approximately **18% false negatives** — documents correctly usable under open licences that had been left at GIALLO because their permission clauses were formulated in ways the regex patterns did not anticipate — and **errors of some kind in roughly 27% of all cases**. These findings led to the current design in which the LLM layer is the decisive arbiter for every URL that Layer 1 does not resolve unambiguously, with pattern matching serving as a pre-filter that enriches the input context passed to the model.

Layer 3 — LLM disambiguation with `analizza_con_llama()`. For URLs that reach this layer, the function assembles a composite text snippet: the first 1 000 characters of the page body, the footer section (or last 1 000 characters), and a "sniper" context window centred on the three nearest text nodes to the keywords *license*, *rights*, *copyright*, and *terms of use*. For direct PDF files, the first two and the last page are extracted via PyPDF2. The input to Mistral-Nemo is therefore deliberately concentrated around the sections most likely to contain a licence statement.

The prompt instructs the model to classify the document as VERDE, ROSSO, or GIALLO by checking for permission-granting clauses (reproduction authorised, source acknowledgement required, open access) against restrictive clauses (all rights

reserved, unauthorised use prohibited), in any of nine languages, and to return a strict JSON object of the form:

```
{"decisione":"VERDE/ROSSO/GIALLO","licenza":"...","lingua":"..."}
```

The call is streamed token-by-token with a balanced-brace completion detector that stops the stream as soon as a valid JSON object is received, capping total latency to `TIMEOUT_STREAMING_LICENSE = 90` seconds.

After JSON parsing, a single safety override is applied: if the model returns ROSSO but the original composite text contains any of the open-licence terms (*public domain*, *open access*, *cc by*, *cc0*, *freely available*), the decision is overridden to VERDE. This targets the most common failure mode observed during calibration — the model over-weighting a copyright symbol in the header while ignoring the explicit permission statement in the footer. If JSON parsing fails entirely, a plain-text keyword scan on the raw response resolves contradictions in favour of VERDE.

The web fetch used in this phase mirrors the two-tier strategy described in Section **??**: a `requests.Session` with a realistic browser `User-Agent` is attempted first; if the response is an empty shell or a SPA fingerprint is detected, Selenium with headless Chrome renders the page and returns the fully-injected HTML for licence scanning.

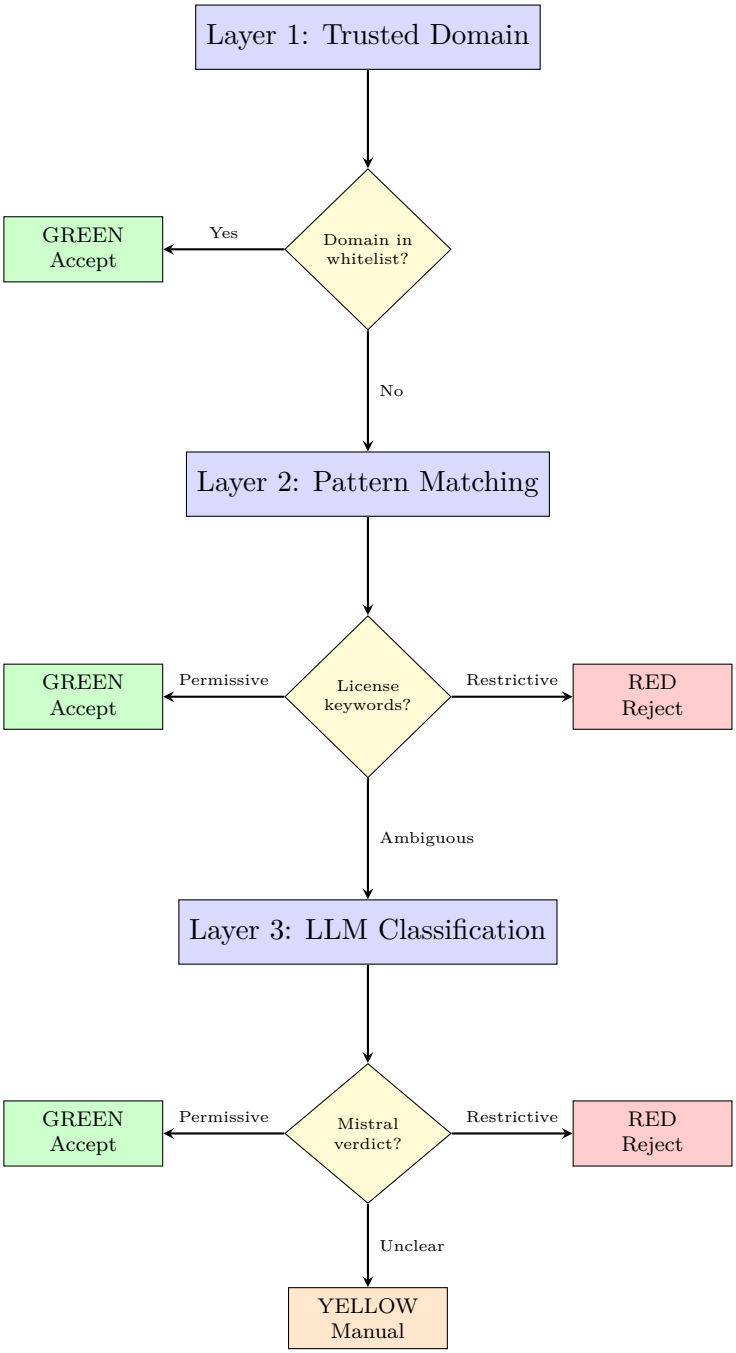

Figure 4.1: Three-layer licence analysis cascade. Layer 1 (trusted-domain whitelist) fast-tracks known sources; Layer 2 (regex patterns) catches explicit license statements; Layer 3 (LLM classification via Mistral-Nemo) handles ambiguous cases, outputting green (permissive), yellow (manual review), or red (restrictive).

==========

4.2 Phase 2: Document Acquisition

4.2.1 Web Scraping Context and File Naming

Web scraping institutional policy portals presents challenges that differ substantially from scraping news sites or product catalogues. The target corpus spans organisations operating under fundamentally different publishing practices: some provide a direct PDF link on the seed URL itself; others aggregate dozens of documents on a single portal page; others publish a brief landing teaser and require following a "Read more" link to reach the actual file. A non-trivial fraction renders its entire index through client-side JavaScript frameworks that return an empty HTML shell to static HTTP clients.

Phase 2 handles all of these cases through `DownloadAgent.download_item()`, a multi-step function that applies a LLM-based relevance gate, classifies the page type, and routes to the appropriate extraction strategy. Every file that is successfully retrieved is saved to `CARTELLA_OUTPUT` with a filename prefixed by the row's `Document_Code` (e.g., `042_climate-strategy-spain.pdf`). This naming convention is critical: Phase 6 (quality check) and Phase 3 (metadata extraction) both rely on the numeric prefix to join each local file back to its row in the state DataFrame without ambiguity, even when a single seed URL yields multiple documents.

4.2.2 Page-Type Classification

Before any download logic is applied, `classifica_tipo_pagina_html()` assigns the page to one of three categories using a hybrid rule-then-LLM strategy.

DIRECTORY. A page is classified as DIRECTORY when it contains at least three unique links whose text or `href` attribute contains at least one policy-domain keyword from the whitelist (*policy*, *directive*, *regulation*, *framework*, *strategy*, *plan*, *act*, *law*, *ordinance*). This condition is checked with `confidence = 0.95`. A DIRECTORY page does not contain the document itself; it is an index page aggregating links to potentially many documents. The function accumulates *all* links satisfying the whitelist and passing the topic blacklist into `suggested_links`. This full list is then forwarded to the fuzzy-match link selection step, which filters it further using the original document title from the Excel workbook as the reference string.

LANDING. If the DIRECTORY condition is not met but the page contains one or more anchor elements whose visible text matches a "learn more" pattern (*learn more*, *read more*, *view full*, *see details*, *access document*, *full text*) — excluding links whose `href` points to privacy, cookie, login, or settings pages — the page is

classified as LANDING with `confidence = 0.85`. A LANDING page is a teaser: it carries minimal content and one prominent link to the actual document. The action consequence is to follow the primary "learn more" link and re-classify the resolved destination, effectively treating the LANDING page as a redirect pointer rather than a content source.

CONTENT. When neither the DIRECTORY nor the LANDING condition triggers, the ambiguous case is resolved by Mistral-Nemo with a minimal 10-token prompt at `temperature 0.05`. A CONTENT page is one that contains the full document text inline — typically an HTML policy document with no navigational links to separate files. The action is to save the HTML as a cleaned TXT file using `salva_html_pulito()`.

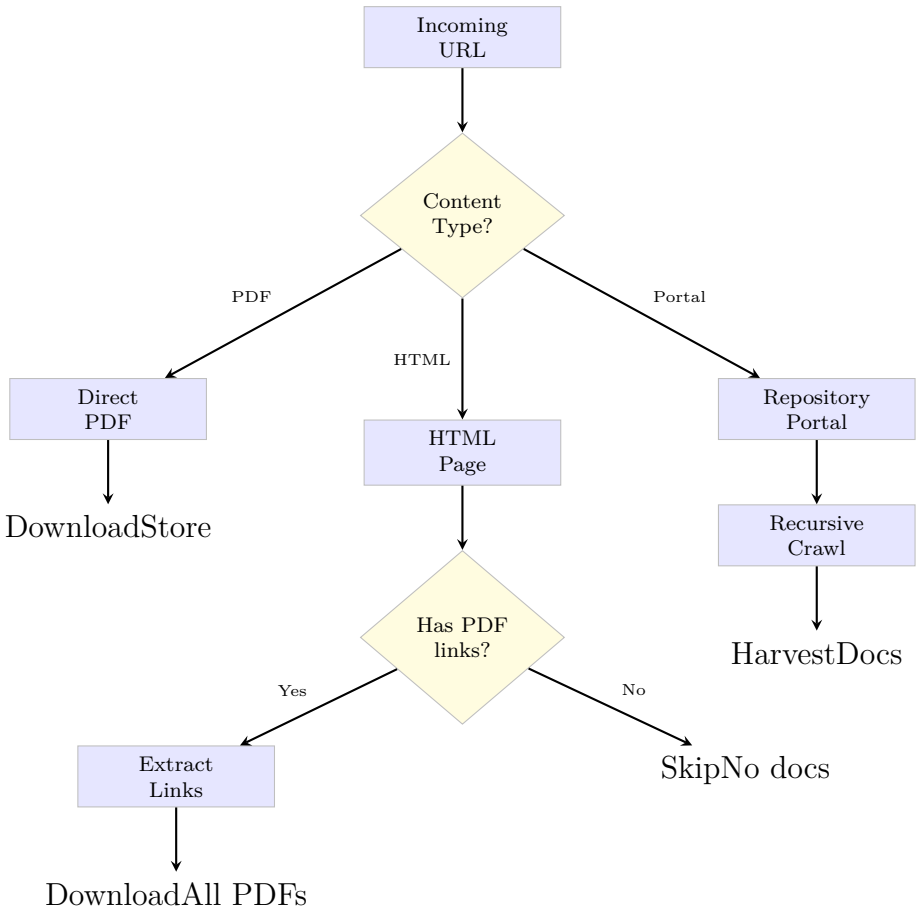

Figure 4.2: Page-type classification and download routing in Phase 2. The pipeline distinguishes direct PDF links, HTML pages with embedded PDF references, and institutional portals requiring recursive exploration, routing each to the appropriate acquisition strategy.

4.2.3 LLM-Based Relevance Pre-Screening

Before any download is initiated, `valida_rilevanza_documento_llm()` submits the candidate document's link text and surrounding HTML context to Mistral-Nemo with a compact YES/NO prompt scoped to the DRR and climate resilience domain. The call uses `num_ctx: 512` and a 5-second timeout to maintain sub-2-second gate latency even on CPU. Documents that receive a NO response are skipped entirely without consuming download bandwidth or disk space.

4.2.4 Fuzzy-Match Link Selection

For DIRECTORY pages the full link list can contain dozens of candidate URLs. The function `cerca_link_con_fuzzy_matching()` computes a similarity score between each candidate link's visible text and the document's *Name* field from the source Excel workbook, using `rapidfuzz`. Links scoring above `FUZZY_THRESHOLD_HTML` are passed to the download queue; links below the threshold are silently discarded. This filter is critical for DIRECTORY pages that aggregate documents from multiple policies: without title-based filtering, every link on an institutional portal would be downloaded regardless of topical relevance.

4.2.5 Direct-File Download: requests and Selenium

When a candidate URL is detected as a direct binary file (PDF, DOCX, XLSX) by `e_link_diretto_scaricabile()`, `download_item()` attempts retrieval in two steps. First, a `requests.get()` call with browser-like headers and a 30-second timeout retrieves the raw bytes. If the response is HTTP 200, the bytes are written to disk via `salva_singolo_file()` with the `Document_Code` prefix. A content-type validation guards against the case where a server returns an HTML error page for a URL that ends in `.pdf`: if `.pdf` is in the URL but `text/html` is in the `Content-Type` header, the requests path is aborted and the Selenium fallback is invoked.

For HTTP 403 (bot-detection block) or content-type mismatches, `scarica_file_con_click_selenium()` navigates to the URL in the headless Chrome instance configured with automatic download preferences and waits for the file to appear in `CARTELLA_OUTPUT`. The Selenium download profile sets the default download directory, disables the download dialog, and forces PDFs to download rather than previewing in the browser. If neither method produces a file, the URL is logged as a download failure in the state DataFrame and processing continues with the next row.

4.2.6 Text Extraction and Pre-cleaning

Regardless of source format, every downloaded document undergoes text extraction and pre-cleaning before it is stored in the state DataFrame. This step happens during acquisition (for HTML CONTENT pages) and again during metadata extraction (Phase 3); the result is the same `pre_clean_text()` pipeline.

For **PDF files**, `PyPDF2` iterates over all pages and concatenates their text. If the extracted string is shorter than 400 characters — a sign of a scanned document without embedded text — `easyocr` is invoked via `pdf2image` at 200 dpi for the first ten pages. The multilingual EasyOCR reader is initialised for English, Italian, Spanish, French, German, and Portuguese. For **DOCX files**, `python-docx` extracts paragraphs, table rows (joined with `" | "`), and section headers/footers. For **PPTX files**, slide titles and body text are concatenated per slide.

After raw text extraction, `pre_clean_text()` applies the following transformations in sequence:

1. **Unicode normalisation** (NFKD) to resolve encoding inconsistencies introduced by PDF extraction or multilingual web pages.
2. **Boilerplate removal**: a dictionary of 30+ regex patterns covering cookie policies, privacy notices, "all rights reserved" clauses, navigation labels, and newsletter subscription prompts in eight languages is applied.
3. **URL and e-mail masking**: all hyperlinks are replaced with the token `[URL]` and all e-mail addresses with `[EMAIL]`, reducing noise in the embedding space.
4. **Whitespace normalisation**: consecutive whitespace characters are collapsed to a single space; triple-or-more newlines are normalised to double newlines.
5. **Short-line removal**: lines shorter than 20 characters after stripping — typically page numbers, section dividers, or navigation fragments — are discarded.

The cleaned text is then passed through `compute_content_hash()`, which generates a SHA-256 digest of the cleaned string. This hash is stored alongside the document in the state DataFrame and is used later by the quality-check phase (Section 4.4) and by the embedding agent to detect whether a document has been modified since its last embedding.

4.2.7 HTML Saving Pipeline

For CONTENT-type pages — institutional documents published as web articles rather than downloadable files — the acquisition pipeline calls `salva_html_pulito()`, which converts the rendered HTML to a structured plain-text file. The function proceeds in five steps:

1. **Non-content tag decomposition.** All `<script>`, `<style>`, `<noscript>`, `<svg>`, `<iframe>`, `<nav>`, `<header>`, `<footer>`, `<aside>`, `<form>`, `<button>`, `<link>`, and `<meta>` elements are removed from the parse tree before any text is extracted.

2. **Semantic container selection.** The function looks for a `<main>` element first, falling back to `<article>`, then `<body>`, then the full document, to focus extraction on the primary content region.

3. **Line-level cleaning.** The extracted text is split into lines; lines shorter than 3 characters and lines matching boilerplate patterns (*cookie policy*, *all rights reserved*, *skip to content*) are discarded.

4. **Document framing.** The cleaned lines are assembled into a structured plain-text document with a header block containing the source URL, the extraction timestamp, the `<title>` tag content, and the first `<h1>` heading.

5. **File naming.** The output file is written as `{Document_Code}_{url_slug}.txt` in `CARTELLA_OUTPUT`, with a counter suffix to avoid collisions if the same slug appears more than once.

4.3 Phase 2.5: Recursive Portal Exploration

4.3.1 Motivation: DIRECTORY Pages as Entry Points

DIRECTORY-classified URLs are not treated as terminal download targets in Phase 2; they are entry points whose aggregated link lists are written to `nuovi_link_da_directory.txt` for deferred recursive exploration. Phase 2.5 processes this file through `processa_link_da_directory()`, which applies a two-level recursive strategy designed to recover files that sit one or two navigational hops away from the seed URL. The same procedure is applied to REST/JSON catalogue endpoints discovered during Phase 2 via `processa_link_da_cataloghi()`.

4.3.2 Two-Level Recursive Exploration

The core of Phase 2.5 is the method `DownloadAgent.esplora_pagina_livello2()`, which implements a bounded depth-first search over the portal's link graph. The function accepts a `profondita_max = 2` parameter and a `max_link_per_livello = 5` cap, creating a search space of at most $5 \times 5 = 25$ pages before the root.

The internal recursive function `_esplora_ricorsivo()` operates as follows at each visited URL:

1. **Termination conditions.** If the current depth exceeds `profondita_max` or the URL has already been visited in this session (tracked in a `pagine_visitate` set), the branch is pruned immediately.

2. **Page loading.** The page is fetched via `carica_pagina_con_cache()`, which checks an in-session cache before issuing a new `requests.get()` call (or routing to Selenium if the response is insufficient). The BeautifulSoup object is cached so that pages referenced by multiple paths in the search tree are not re-fetched.

3. **Direct-file scan.** `cerca_link_con_fuzzy_matching()` is called with `threshold = 0.25` and `solo_diretti = True`, collecting all PDF/-DOCX/XLSX links on the page regardless of title similarity. Each candidate URL is then validated against the whitelist/blacklist via `validate_link_for_download()` before being added to `file_trovati`.

4. **Promising-link selection for next level.** If the current depth is below `profondita_max`, `trova_link_promettenti()` is invoked to identify the `max_link_per_livello` (= 5) most semantically promising HTML links on the current page. Candidate links are scored by keyword overlap with the parent document's title, with higher weight given to domain-relevant terms (*report*, *assessment*, *framework*, *policy*). The top-5 links are then explored recursively at `depth + 1`.

After the full recursive traversal, duplicate URLs are eliminated from `file_trovati` before downloading. Each discovered file is then subject to the same LLM relevance gate (`valida_rilevanza_documento_llm`) and the same requests-then-Selenium two-tier download strategy described in Section 4.2.5. Newly downloaded files receive a `Document_Code` of the form `{parent_code}.{file_index}` and are appended to the DataFrame with a `Parent_Code` field linking them to the originating seed.

4.4 Phase 6: Quality Control

4.4.1 File-Level Validation

Phase 6 is invoked as a discrete step before metadata extraction to eliminate files that would produce misleading or uninformative metadata. It scans all files under `CARTELLA_OUTPUT` recursively and applies three independent validation gates.

Size and page-count thresholds. A file must be at least `MIN_SIZE_BYTES = 200 000` bytes (~200 KB) and contain at least `MIN_PAGES = 4` pages (for PDFs). Files failing either threshold are classified as `REJECTED` and deleted from the

filesystem. The thresholds are deliberately conservative: a 5-page document in the DRR domain is almost always either a one-page factsheet or an administrative notice that would contribute no substantive content to the corpus.

Title blacklist. Files whose `Document_Code`-prefixed filename resolves to a title matching any entry in a multilingual blacklist of 80+ patterns are flagged as `NO_PREFIX` or `REJECTED`. Blacklist categories span legal boilerplate (*privacy policy*, *terms of service*, *cookie policy* in English, Italian, Spanish, French, Portuguese, Turkish, and Norwegian), navigational artefacts (*home*, *sitemap*, *footer*, *sidebar*), and metadata stubs (*untitled*, *page not found*).

Document_Code prefix check. Every file must begin with its numeric `Document_Code` prefix. Files lacking this prefix cannot be joined back to the state DataFrame and are flagged as `NO_PREFIX`; they are deleted from the filesystem and excluded from all subsequent phases.

4.4.2 Content-Based Deduplication

After per-file validation, Phase 6 applies three-pass deduplication over the set of VALID documents.

The first and most precise pass uses the SHA-256 content hashes computed during the pre-cleaning step (Section 4.2.6). Two documents with identical hashes are byte-for-byte identical after cleaning, regardless of filename or URL, and all but one instance are removed.

The second pass computes pairwise title similarity using `difflib.SequenceMatcher` between normalised filename stems, stripping the `Document_Code` prefix, punctuation, and year tokens before comparison. Pairs above `SIMILARITY_THRESHOLD = 0.85` are grouped; within each group, `seleziona_versione_preferita()` retains the URL corresponding to the most informative language (ranked by `LINGUA_PRIORITA`) and the most specific version suffix (*full*, *complete*, *technical*).

The third pass applies URL normalisation via `normalizza_url_per_confronto()`, stripping tracking parameters, language-code suffixes, and trailing slashes before comparing. Documents that share a normalised URL but differ in raw URL formatting are deduplicated by keeping the canonical form.

Deleted files are removed from the filesystem atomically alongside the corresponding DataFrame rows, ensuring that the state file always reflects the actual contents of `CARTELLA_OUTPUT`. A summary report listing all rejected documents, their rejection reasons, and their original Document_Codes is written to `quality_check_rejected.xlsx` for manual audit.

4.5 Phase 3: Metadata Extraction

4.5.1 Domain Taxonomy and Controlled Vocabularies

Before detailing the extraction mechanics, it is worth contextualising the information that the pipeline is designed to capture and why its structure matters. The corpus is built to support queries on disaster risk reduction and climate resilience policy, a domain characterised by a well-established international taxonomy codified primarily in the Sendai Framework for Disaster Risk Reduction 2015–2030 [1] and the Paris Agreement [67]. Metadata fields that express this taxonomy are not decorative: they are the structured predicates over which ChromaDB's metadata-filtered retrieval (Phase 4) operates, allowing the RAG system to restrict retrieval to documents matching a specific hazard type, geographical scope, or document category before similarity search is even performed.

The metadata schema therefore comprises five controlled-vocabulary fields that are defined as Python constants in the source and injected into every LLM extraction prompt:

- **Hazard type** (`HAZARD_TYPES`): *Flood*, *Earthquake*, *Landslide*, *Wildfire*, *Heatwaves*, *Climate change*, *Multi-hazard*, *Other (specify)*.
- **Geographical level** (`GEOGRAPHICAL_LEVELS`): *Global/Supranational*, *Supranational/regional*, *European region*, *Latin-American region*, *National (specify country)*.
- **Document type** (`DOCUMENT_TYPES`): *Policy brief*, *Report*, *Law*, *Glossary*, *Academic paper*, *Recommendation*, *Guidelines*, *Framework*, *Dataset*.
- **Key criteria** (`KEY_CRITERIA`): five cross-cutting resilience dimensions co-designed with domain experts — *Adaptive governance*, *Health and wellbeing*, *Active memory*, *Social interaction and inclusiveness*, *Socio-economic resilience* — each accompanied by a definition string that specifies the minimum evidence threshold required for assignment.
- **Additional tags** (`ADDITIONAL_TAGS`): seven thematic tags — *Culture and Heritage*, *Communication and user needs*, *Psychological aspect*, *Preparedness and Mitigation phase*, *Vulnerable groups*, *Gender*, *Community-based approach* — also with explicit definition strings.

The definitions for `KEY_CRITERIA` and `ADDITIONAL_TAGS` are not merely labels: they are multi-sentence natural-language descriptions that the LLM is instructed to apply as minimum-evidence thresholds, with explicit rules against assigning a tag from vague or generic language. This design was motivated by early experiments

in which the model over-tagged documents with *Adaptive governance* whenever the word "adaptive" appeared in any context.

4.5.2 LLM Extraction Prompt

The extraction function `estrai_metatag_con_llm_migliorato()` begins by pre-processing the document's HTML (or TXT) content. Non-content tags (`<script>`, `<style>`, `<footer>`, `<nav>`, `<aside>`, `<iframe>`, `<noscript>`) are decomposed; the remaining text is joined with heading elements (`h1`, `h2`, `h3`, up to 5 each) prepended for emphasis. Any existing `<meta name="description">` and `<meta name="keywords">` content is extracted separately as `desc_hint` and `keywords_hint` to serve as fallback values.

The prompt is constructed by injecting the full controlled-vocabulary lists and their definition strings alongside up to 10 000 characters of prioritised content. It requests twelve output fields — official document title, issuing authority, document type, hazard, target audience and relevance summary (40–50 words), key criteria, additional tags, description (100–130 words), language code, publication date, geographical level, and internal quality notes — and imposes a strict JSON-only output format with no preamble.

The call is made with `temperature: 0.08`, `top_p: 0.85`, `top_k: 50`, `repeat_penalty: 1.1`, `num_ctx: 8196`, and `num_predict: 1200`. The response is streamed and a balanced-brace completion detector stops the stream as soon as a syntactically complete JSON object is received, before the `done` signal from Ollama. The global timeout `TIMEOUT_METADATA_LLM = 800` seconds accommodates the longest documents without blocking the pipeline indefinitely.

4.5.3 JSON Parsing and Output Validation

Raw LLM output is processed through a multi-step validation pipeline. A greedy regex extracts the outermost JSON object; mismatched braces are repaired by appending the deficit number of closing braces before attempting `json.loads()`.

Each field is then validated against its controlled vocabulary. The hazard field is split on commas and each token is matched against `HAZARD_TYPES` using a case-insensitive substring search; tokens not matching any entry are discarded. Geographical level is matched against `GEOGRAPHICAL_LEVELS`; if no match is found, `inferisci_geographical_level()` infers it from URL structure (country-code top-level domains, regional path segments). Document type is matched against `DOCUMENT_TYPES` with a fallback of `"Other"`.

Publication dates must satisfy a strict format requirement: today's and yesterday's ISO dates (computed at runtime) are explicitly prohibited, as they would

indicate that the model extracted a download timestamp rather than the document's actual publication date. Dates failing this check are overridden to `"Not provided"`.

The document title undergoes `validate_title()`, which checks for minimum length, absence of generic patterns (filenames, error messages, website names), and returns a cleaned version. If the LLM-generated title fails validation, the function attempts to recover an alternative from the first 200 characters of the first PDF page via `extract_title_from_first_page()`.

Language is inferred from a combination of the LLM output and a pre-extraction language detection hint provided by `detect_language_fast()`, which uses character n-gram statistics before any LLM call. The LLM value is used only if it differs meaningfully from the automatic detection.

4.5.4 Traditional HTML Fallback and Hybrid Merge

When the LLM call fails — due to a timeout, a connection error, or a JSON parse failure that survives all repair attempts — `estrai_metatag_tradizionale()` provides a BeautifulSoup-based fallback. It extracts the `<title>` tag, `<meta name="description">` and `<meta name="keywords">` content, Dublin Core `dc.title` and `dc.date` meta elements, the `lang` attribute of the `<html>` element, and the first `<h1>` heading.

The final metadata record is produced by `estrai_metatag_intelligente()`, which merges the LLM and traditional outputs: LLM values take precedence for all fields where they are non-empty and pass validation; traditional values fill any remaining gaps; fields that are empty after both passes default to `"Unclear"` or `"Not provided"`. All merged values are stored back in the state DataFrame under their canonical column names.

4.5.5 Quality Scoring

`calculate_quality_score()` assigns each document an integer score from 0 to 100 based on four independent signals:

- **Cleaned text length** (up to 50 points): documents with 2 000–50 000 cleaned characters score 50; above 50 000 score 40 (penalising noise-heavy documents); 1 000–2 000 score 30; below 1 000 score 20.

- **Chunk count** (up to 20 points): the chunk count is estimated here as a fixed-stride preview (one chunk per 1,000 characters) computed during Phase 3 ingestion, not via the semantic chunking strategy applied in Stage II (Chapter 5).

- **Section detection** (up to 15 points): documents with 3 or more recognised section headers from `SECTION_HEADERS` score 15; any sections score 7.
- **Cleaning reduction ratio** (up to 15 points): a boilerplate reduction of 10–40% scores 15; less than 10% scores 5 (indicating minimal noise or very clean input).

The quality score is stored in the `Quality Score` column of the output dataset and is not used as a hard filter: it provides a diagnostic signal for the corpus review step and for the RAG evaluation analysis.

4.5.6 Structured Output: `Estrazione Finale`

At the completion of Phase 3, `genera_csv_strutturato()` assembles the full metadata for all successfully processed documents into a structured Excel file. The function also handles incremental merging: if a previous run has already produced `Estrazione finale.xlsx`, the new rows are concatenated and deduplicated by `Document_Code` before writing, ensuring that partial re-runs do not create duplicate entries.

Each row in the output represents one document and carries 24 columns, summarised in Table 4.1.

This file is not merely a summary of what was collected: it is the primary input contract for Stage II. The embedding agent reads it as its sole source of document paths, metadata filters, and pre-cleaned text. The `Local File Path` column resolves each row to a downloadable binary; the five controlled-vocabulary fields ($\star$) become ChromaDB metadata predicates; and the `Content Hash` enables incremental embedding by skipping documents whose content has not changed since the last indexing run. The quality of Stage II retrieval is therefore directly bound to the fidelity of the Phase 3 extraction — making the investment in taxonomy design, prompt engineering, and validation logic described in this section a prerequisite for the entire system.

Table 4.1: Schema of the structured output file `Estrazione Finale`. Columns marked ⋆ are used as ChromaDB metadata filters in Stage II.

Column	Source	Type
`Document_Code`⋆	State file	Integer string (primary key)
`Policy Title`	LLM / PDF extract	Free text
`Issuing Authority`	LLM	Free text
`Geographical Level`⋆	LLM / URL infer	Controlled vocab
`Hazard`⋆	LLM	Controlled vocab
`Target and Relevance`	LLM	40–50 word summary
`Key Criteria`	LLM (validated)	Controlled vocab list
`Additional Tags`	LLM (validated)	Controlled vocab list
`Description`	LLM	100–130 word summary
`Document type`⋆	LLM (validated)	Controlled vocab
`Source`	State file	URL
`Language`⋆	LLM / auto-detect	ISO 639-1 code
`Date of Publication`	LLM / HTML meta	YYYY-MM-DD
`License`	Phase 1	Licence string
`AI Notes`	LLM self-report	Confidence flag
`Local File Path`	Download agent	Filesystem path
`Content Hash`	SHA-256	64-char hex digest
`Quality Score`	`calculate_quality_score()`	0–100 integer
`Sections Found`	Section detector	List of section keys
`Key Terms`	Term extractor	Free text
`Num Chunks`	Chunker preview	Integer
`Cleaned Text Length`	`pre_clean_text()`	Character count

Chapter 5

Stage II: RAG System

Stage II receives the structured output of Stage I — the `Estrazione Finale` spreadsheet and the associated local document corpus — and transforms it into a queryable, fully offline Retrieval-Augmented Generation system. The transition between the two stages is designed to be incremental: the spreadsheet carries a SHA-256 `Content Hash` column computed during ingestion, which Stage II uses as a cache key to avoid re-processing documents that have not changed since the last run.

Stage II is composed of five tightly coupled subsystems, each described in a dedicated section of this chapter. The *hierarchical chunking pipeline* (§5.1) converts raw document text into a two-level parent–child structure that balances retrieval precision with answer completeness. The *embedding and vector store layer* (§5.2) encodes every child chunk with context-weighted embeddings and persists them in ChromaDB. The *hybrid retrieval pipeline* (§5.3) fuses dense cosine-similarity search with sparse BM25 keyword matching through Reciprocal Rank Fusion. The *response generation module* (§5.4) assembles a structured prompt, injects output instructions extracted from the user query, and calls a locally running Mistral-Nemo instance via Ollama. An offline re-implementation of the *RAGAs evaluation framework* (§5.5) provides automated quality measurement over a curated gold set. Finally, two interactive interfaces (§5.6 and §5.6) expose the system to end-users — a command-line session for development and a Streamlit web application for operational use.

The entire Stage II is designed for complete offline operation: no external API calls are made at any point after setup. Ollama serves the language model locally, ChromaDB stores the vector index on the local filesystem, and the RAGAs judge runs on a second Ollama instance. This choice was driven by data-governance constraints common in public-sector policy analysis, where sending document content to a third-party cloud service may be impermissible.

5.1 Hierarchical Document Chunking

Transforming raw document text into units suitable for embedding is one of the most consequential design decisions in a RAG pipeline. A chunking strategy must simultaneously serve two masters: the *retrieval* component, which benefits from short, focused passages that match query semantics tightly, and the *generation* component, which needs enough surrounding context to produce coherent and complete answers. Naive fixed-size splitting — dividing text every n characters or tokens — satisfies neither goal well: passages may cut mid-sentence, sever co-referential pronouns from their antecedents, or mix topically unrelated material in the same chunk [32].

The solution adopted here is a *hierarchical parent–child* architecture in which semantic boundaries determine where parent segments begin and end, and finer child chunks are carved from each parent for embedding. The flow is implemented in the `AdvancedChunker` class, which orchestrates four sequential steps: text extraction and cleaning, sentence splitting, semantic boundary detection, and parent–child construction.

5.1.1 Text Extraction, Normalisation, and Sentence Splitting

The pipeline handles four file formats. For **PDF** documents — the dominant format in the corpus — text is extracted with PyPDF2. When PyPDF2 yields fewer than 400 characters, the document is likely a scanned image; in that case EasyOCR is invoked as a fallback at 200 DPI, processing at most ten pages and supporting six languages (English, Italian, Spanish, French, German, Portuguese). **DOCX** files are parsed with python-docx, which iterates over paragraphs, table cells, and header/footer sections to preserve structure that a plain text dump would lose. **Excel** files are handled by a custom `estrai_testo_da_excel` function; **plain text** files are read directly with UTF-8 decoding.

Raw text extracted from PDFs typically contains layout artefacts that would fragment sentences unpredictably: mid-word hyphenation at line breaks, inconsistent whitespace around numbered lists, running headers embedded in body text, and duplicate punctuation introduced by the PDF parser. A `pre_clean_text` function addresses these before any further processing.

The cleaned text is then split into sentences by the `split_into_sentences` method. PDF-originated text presents a particular challenge because many policy documents use unnumbered bullet lists that lack terminal punctuation. Four normalisation rules are applied in sequence before the final split pattern:

Fix 1. Bullet normalisation. Visual bullet characters (bullet points: •, ·, -, *,

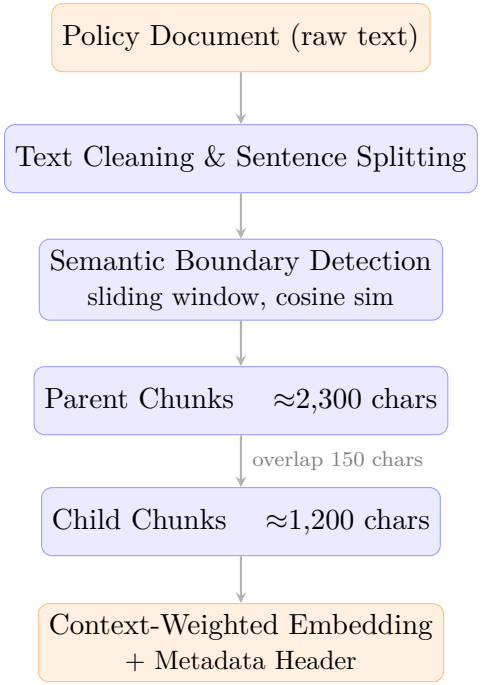

Figure 5.1: Hierarchical chunking pipeline. Raw document text is cleaned and split into sentences, then divided into semantically coherent parent segments via sliding-window cosine similarity. Each parent is subdivided into overlapping child chunks (≈1,200 chars) that are encoded with a structured metadata prefix before storage in ChromaDB.

and similar markers) are replaced by a period followed by a space, converting list items into grammatically separable sentences.

Fix 2. Paragraph boundary injection. Any alphanumeric character followed by a double newline is assumed to end a sentence: a period is injected before the whitespace.

Fix 3. Mid-sentence newline bridging. A single newline between a line ending with a letter, digit, or closing bracket and a line beginning with an upper-case letter or digit is treated as an implicit sentence break.

Fix 4. Duplicate punctuation cleanup. Accidental double-period sequences (`. .`), which can arise from the preceding rules, are collapsed to a single period.

The primary split pattern then fires on three triggers: a sentence-ending punctuation mark (`.`, `!`, `?`) followed by whitespace and an upper-case letter; a punctuation mark followed by a newline; or two or more consecutive newlines. Resulting fragments shorter than five characters — typically page numbers or OCR noise — are discarded.

5.1.2 Parent–Child Hierarchical Architecture

Once sentences have been obtained, the `create_hierarchical_chunks` method builds a two-level hierarchy. At the upper level, semantically coherent *parent chunks* group sentences that share the same topic. At the lower level, overlapping *child chunks* are carved from each parent for embedding and vector-store indexing.

Semantic boundary detection. The `detect_semantic_boundaries` method locates topical transitions without requiring any document-specific labelling. A sliding window of width $W = 3$ sentences is moved over the sentence sequence; at each position i, two mean embeddings are computed — one for the W sentences to the left of i, one for the W sentences to the right — and the cosine similarity between the two window means is recorded. A low similarity value signals a topic change. To avoid sensitivity to the absolute magnitude of similarity scores (which varies across document types and languages), the threshold is set dynamically to the 25th percentile of all pairwise window similarities in the document. A minimum gap of `MIN_GAP_BETWEEN` $= 4$ sentences is enforced between consecutive boundaries to prevent over-fragmentation of short paragraphs. When fewer sentences are available than required by the window arithmetic, or when the embedding model is unavailable, the method falls back to a deterministic stride of five sentences.

Parent chunk construction. Each pair of consecutive boundaries delimits a candidate parent. `PARENT_CHUNK_SIZE` is set to 2,300 characters. Candidates whose character length exceeds $1.5 \times 2{,}300 = 3{,}450$ characters are split recursively by the `_split_oversized_parent` utility, which greedily fills sub-parents up to 2,300 characters before starting a new one.

Child chunk construction. Each parent is subdivided into overlapping child chunks by `_create_children_from_parent`. The word count is derived from the character budget using an empirically calibrated conversion factor of 4.7 chars/-word, representative of English and multilingual policy vocabulary. The derived parameters are:

Table 5.1: Hierarchical chunking parameters and rationale.

Parameter	Value	Rationale
`PARENT_CHUNK_SIZE`	2,300 chars	Fits within the mpnet 512-token window; large enough to preserve paragraph-level argumentative context
`PARENT_CHUNK_OVERLAP`	200 chars	Prevents sentences at segment boundaries from losing their surrounding context
`CHILD_CHUNK_SIZE`	1,200 chars	≈270 tokens on mpnet — short enough for precise semantic matching, long enough to carry a full argument
`CHILD_CHUNK_OVERLAP`	150 chars	Roughly one sentence of overlap to prevent retrieval gaps at chunk boundaries
`MIN_CHUNK_LENGTH`	150 chars	Discards noise fragments: page numbers, running headers, OCR artefacts
chars/word ratio	4.7	Empirically calibrated on English and multilingual policy vocabulary

When the final word slice of a parent would yield a chunk shorter than `MIN_-CHUNK_LENGTH`, it is merged into the preceding child rather than stored as a stub. Each child carries a structured metadata record that includes the full parent text (capped at 2,000 characters for ChromaDB compatibility), its position indices `(parent_idx, child_idx)`, and a boolean `has_context` flag.

The chunk identifier stored in ChromaDB is formed as `{SHA256[:12]}_-p{parent_idx}_c{child_idx}`, which encodes both the document provenance and the structural position of the chunk within the document.

5.1.3 Context-Weighted Late Chunking

A known limitation of standard chunk-level embedding is *context blindness*: the encoder sees only the text of the child chunk, without any signal about the broader document it belongs to. Two chunks containing identical wording drawn from documents on entirely different hazard categories would receive the same or very similar embedding vectors, making correct retrieval by topic impossible.

The technique known as *late chunking* [32] addresses this by ensuring that the full document context is present in the encoder's attention window before the chunk representation is extracted. In its original formulation, the encoder processes the entire document at once; token-level embeddings at the positions corresponding to each chunk are then pooled to form the chunk vector. This requires an encoder with a very long context window (e.g., Jina Embeddings v3 [37]) and becomes impractical for documents exceeding the encoder's maximum sequence length.

The implementation adopted here is a lighter variant termed *context-weighted embedding*, which avoids the full-document pass. Instead, each child chunk is prepended with a compact *metadata context header* constructed by `_build_context_header` from the document's structured metadata fields — a single pipe-delimited line encoding title, authority, hazard, geographic level, document type, and language.

During embedding, `apply_late_chunking_embeddings` constructs the input as `"DOCUMENT: <header>\n\n<child_text>"` before calling `model.encode()`. Because the encoder attends jointly to both the header and the child text, the resulting vector is tilted toward the document's domain, hazard class, and geographical scope — without requiring any architectural change to the encoder or any increase in batch size.

The token arithmetic is favourable with the active model (`paraphrase-multilingual-mpnet-base-v2`, 512-token limit): the header occupies approximately 22 tokens and the child text approximately 270 tokens, for a combined total well within the model's capacity with no truncation. If the method fails for any reason, `_direct_chunk_embeddings` provides a plain-text fallback that encodes the chunk including its prepended header from the `text` field.

The metadata header serves a dual purpose: it improves embedding quality at indexing time, and it functions as a structured prefix inside the stored `documents` column of ChromaDB, making the domain context available to any downstream component that reads raw chunk text.

5.2 Embedding Generation and Vector Store

The embedding layer bridges the chunking pipeline and the retrieval engine. It is responsible for encoding child chunks into dense vectors, persisting them in ChromaDB, and ensuring that subsequent indexing runs do not re-process documents already present in the store.

5.2.1 Embedding Model Selection

The embedding model selection and evaluation are discussed in Chapter **??**. To summarise the deployment outcome: **Jina Embeddings v3** [37] was the preferred choice (1,024 dimensions, native late-chunking flag, 8,192-token context window) but was disabled on the PoliTO VM due to RAM and latency constraints at 570M parameters, and because loading it requires `trust_remote_code=True`. The active model is **paraphrase-multilingual-mpnet-base-v2** [34]: 278M parameters, 768 dimensions, 50 languages; batch encoding of 32 chunks completes in approximately 2–3 seconds on CPU, making the full corpus processable in less than a day. **paraphrase-multilingual-MiniLM-L12-v2** [34] activates automatically as fallback if mpnet fails to load.

Table 5.2 summarises the trade-off.

Table 5.2: Embedding models evaluated for the multilingual corpus.

Model	Status	Params	Dim	Notes
Jina Embeddings v3	Disabled	570M	1,024	Native late-chunking flag; 8,192-token window. Disabled: RAM/latency on PoliTO VM; requires `trust_remote_code=True`.
mpnet-multilingual	**Active**	278M	768	50 languages; knowledge-distilled. 2–3 s per batch of 32 on CPU; full corpus overnight.
MiniLM-multilingual	Fallback	118M	384	Faster but lower precision on long policy passages. Auto-activates if mpnet fails.

5.2.2 ChromaDB Indexing and Collection Design

ChromaDB [41] was selected as the vector store for several converging reasons. It is embedded — no separate server process is required — and persists the index directly to the local filesystem via a `PersistentClient`, satisfying the offline-operation requirement. Unlike FAISS [40], which is a pure approximate nearest-neighbour library with no native metadata management, ChromaDB exposes a document store alongside the vector index, allowing rich per-chunk metadata to be stored and filtered without a separate database. The HNSW index [39] used internally by ChromaDB provides sub-linear query time scaling — $O(\log N)$ per query — which remains acceptable at 280,000 chunks. Finally, ChromaDB is actively maintained, offers a Python-native API, and has been widely validated in open-source RAG pipelines [41].

The collection is created with `"hnsw:space": "cosine"`, which instructs the HNSW index to measure distances in cosine space. Cosine similarity is the natural metric for L2-normalised embeddings: it equals the dot product and is invariant to vector magnitude, which is appropriate here because `normalize_embeddings=True` is passed to every `model.encode()` call. The `best_distance` field stored per chunk in `chunk_data` is therefore a cosine distance, and similarity is recovered as $\text{sim} = 1 - d$.

Each chunk stored in ChromaDB carries the metadata fields listed in Table 5.3.

Table 5.3: Per-chunk metadata stored in ChromaDB.

Field	Content
`document_code`	Unique document identifier from the spreadsheet
`parent_idx / child_idx`	Structural position within the document
`parent_text`	Full parent segment (capped at 2,000 chars)
`policy_title`	Document title (capped at 500 chars)
`issuing_authority`	Issuing body
`hazard`	Hazard category from Stage I taxonomy
`geographical_level`	Geographic scope
`document_type`	Typological category
`language`	ISO 639-1 language code
`content_hash`	SHA-256 of document text (64 hex chars)
`quality_score`	Quality score assigned by Stage I

Large batch upserts use `collection.add()` in batches of `CHROMA_MAX_BATCH` = 40,000 chunks to avoid ChromaDB's internal memory limits during insertion of large documents.

5.2.3 Incremental Embedding and Cache Management

The most operationally significant feature of the embedding layer is its incremental update mechanism, which makes Stage II a live system rather than a one-shot index build.

When the `EmbeddingAgent` initialises ChromaDB, it calls `_load_embedded_docs_cache`, which scans every metadata record in the collection and extracts the `content_hash` field. These hashes are loaded into an in-memory Python set (`_embedded_docs_cache`). The set is also serialised to a pickle file on disk so that it survives process restarts without requiring a full collection scan.

Before processing any document, `is_already_embedded(doc_code, content_hash)` checks membership in this set. A 64-character hexadecimal SHA-256 hash is used deliberately: it detects both identical content resubmitted under a different filename and minor edits to an existing document. Two documents with the same title but different content will always receive different hashes.

If the hash is not present, the document is processed through the full chunking and embedding pipeline. If the document was previously embedded but its content has changed (different hash), `_delete_document_chunks` first removes all existing chunks from ChromaDB via a metadata filter on `document_code`, then the fresh chunks are inserted. If the hash matches the cached value, the document is skipped entirely with a `status = 'skipped'` record.

A checkpoint is saved every ten documents via `_save_embedded_docs_cache`, ensuring that a crash or manual interruption does not require restarting the entire batch from scratch. Progress logging emits real-time statistics including elapsed time, per-document latency, throughput in documents per minute, and a dynamic ETA estimate.

On the PoliTO VM, the full embedding of 819 documents (280,000 chunks) in the whole run took approximately 18 hours.

5.3 Retrieval Pipeline

The retrieval pipeline is the component that, given a user question, selects the most relevant passages from the vector store to serve as context for the language model. Its design involves three consecutive phases: query processing and decomposition, hybrid candidate generation through dense semantic search and sparse keyword matching, and result fusion with diversity enforcement. The entry point for the entire pipeline is the `query_rag` function, which coordinates all phases and returns a ranked list of chunks with their associated metadata.

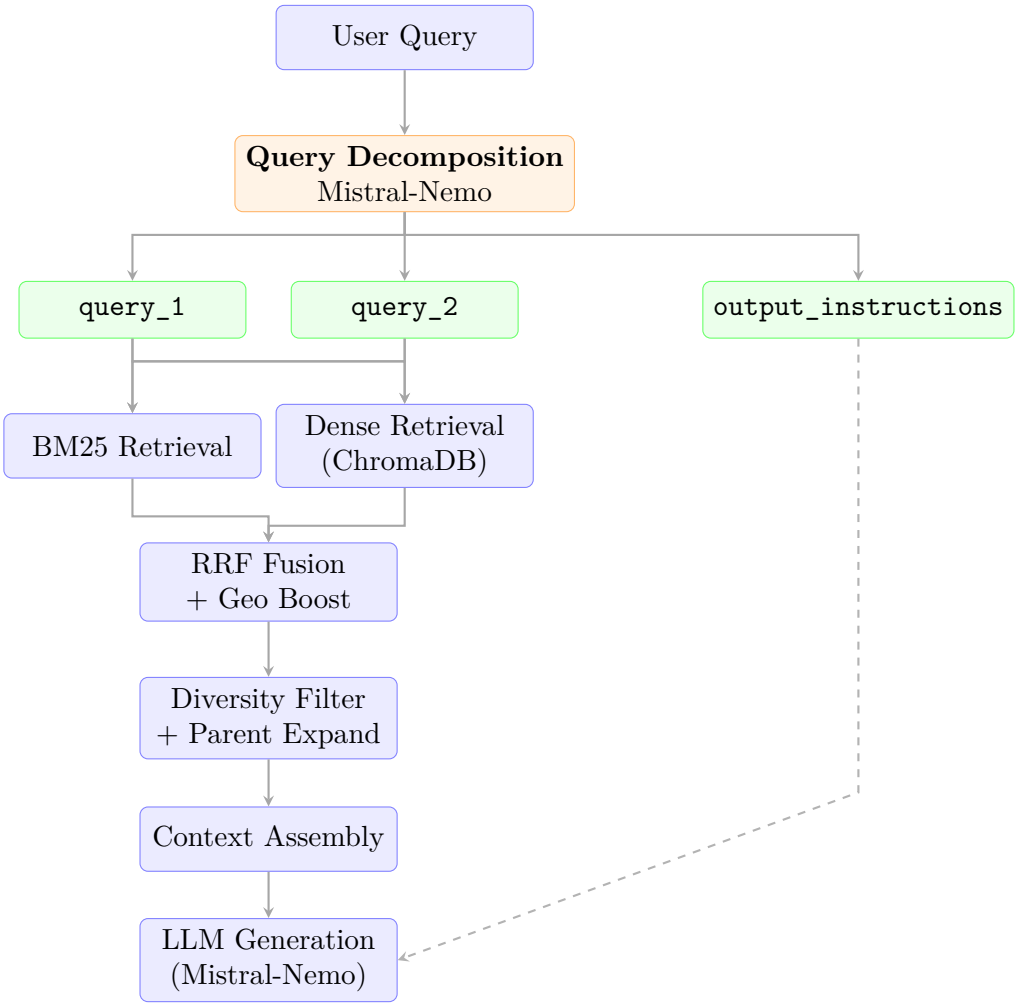

Figure 5.2: Full retrieval pipeline. The user query is decomposed by Mistral-Nemo into two search queries (`query_1`, `query_2`) and an `output_instructions` string. The two queries drive BM25 and dense ChromaDB retrieval in parallel; results are merged via RRF with a geo-aware co-occurrence boost, filtered for per-document diversity, and expanded to parent context before generation. Output instructions are injected directly into the generation prompt (dashed arrow).

5.3.1 Query Processing and Multi-Query Decomposition

Raw user questions submitted to a policy corpus often combine a retrieval intent with formatting or citation requirements. A question such as

> *"Which EU DRR policies mention cultural heritage? Quote relevant passages verbatim and provide full reference [Issuing Authority, Title, Year]."*

contains two distinct components: a search predicate (*EU DRR policies cultural heritage*) and an output instruction (*quote verbatim, provide full reference*). Feeding the full question directly to the embedding model conflates these two components and risks biasing the vector towards formatting keywords rather than domain content.

`query_rag` addresses this through a query decomposition step. Before any vector search, the query is passed to Mistral-Nemo, which returns a JSON object with three fields — two search queries for retrieval and one string capturing the formatting requirements for generation:

`query_1` A clean search query of at most 15 words containing only what to find, with no formatting instructions.

`query_2` A second alternative query of at most 15 words approaching the same information need from a different angle or using different terminology.

`output_instructions` Everything in the original question that describes *how* the answer should be presented (citation format, verbatim quoting, listing style, etc.). Empty string if none.

The model is called at `temperature=0.0` with a 30-second timeout and a token budget of 200. The raw response is cleaned of any Markdown fences and a regular expression extracts the first JSON object. Both queries are validated for length (between 5 and 200 characters); if either fails validation it is silently dropped. If the LLM call times out, returns an empty response, or produces malformed JSON, the function falls back gracefully to the original user question as the sole query variant and an empty `output_instructions` string.

The `output_instructions` field is stored in the result dictionary and later passed to `rag_answer`, where it is appended to the base system prompt as an "Additional Output Requirements" block (see §5.4). This design cleanly separates the *retrieval concern* (what to find) from the *generation concern* (how to present it), preventing mutual contamination.

Geographic scope handling. The seven metadata fields stored per chunk in ChromaDB — including `hazard`, `geographical_level`, `language`, and `document_type` — are exposed as optional filters in both the CLI and the Streamlit interface. When a filter is active, ChromaDB's `where` clause restricts the candidate set before any similarity computation. The `geographical_level` field supports partial matching via the `$contains` operator, so a filter on `"National"` will also match entries labelled `"National - Italy"` or `"National - France"`. In the interactive session, the system also uses the extracted `query_1` to implicitly narrow geographic scope: because the metadata header injected into each chunk during embedding includes the `GEO:` field, a query mentioning a specific country or region will naturally rank documents from that scope higher through cosine similarity, without any hard filter.

5.3.2 Hybrid Retrieval: Dense Search and BM25

Modern retrieval research consistently shows that neither dense nor sparse search dominates in all cases [26, 42]. Dense models capture semantic equivalence across paraphrases and languages but may miss exact technical terms or acronyms that do not appear in their training distribution. Sparse models such as BM25 excel at exact keyword matching but are blind to synonymy and cross-lingual equivalence. The pipeline therefore runs both in parallel and fuses their outputs.

Dense semantic retrieval. For each query variant q_i, the embedding model encodes the text with `normalize_embeddings=True` and the resulting vector is passed to `collection.query()` with `n_results = top_k` $\times$ `retrieval_multiplier`, where `retrieval_multiplier = 5`. The over-retrieval factor of 5 is deliberately generous: because the final diversity filter will discard many candidates (§5.3.3), the candidate pool must be large enough to ensure `top_k` diverse results survive. The `include` parameter requests documents, metadata, and distances.

For each retrieved chunk, the best cosine distance across all query variants is stored in `chunk_data`, and the rank at which the chunk appeared in each variant's ranked list is recorded in `semantic_rankings`. This multi-ranking structure is later consumed by the RRF fusion step.

Parameter calibration. The `top_k` parameter controls how many chunks ultimately reach the language model. Empirical testing over the gold set of 13 domain-specific questions showed that values between 5 and 10 frequently missed relevant passages in documents that span multiple sub-topics, while values above 20 introduced noisy, tangential chunks that increased hallucination risk and exceeded the practical context window available to Mistral-Nemo. Values of `top_k = 15` and `top_k = 20` produced the best trade-off: sufficient coverage of multi-faceted

policy questions while keeping the context manageable. The operational default is `top_k = 15` in the interactive session and in the RAGAs evaluation run.

The `similarity_threshold` parameter acts as a quality floor. In `query_rag` the default is 0.0 (no floor), delegating all threshold-based filtering to `rag_answer`. `rag_answer` applies `similarity_threshold = 0.35` as its default, which was found to eliminate the majority of out-of-domain chunks (similarity < 0.2) while retaining domain-relevant passages in languages with lower absolute similarity scores (Slavic and East Asian languages tend to embed further from English queries even when semantically related).

The per-document diversity cap is set by `max_chunks_per_doc`: 2 when `top_k` ≤ 10, and 3 when `top_k` > 10. This prevents a single comprehensive document from dominating the context window at the expense of other sources, promoting cross-document synthesis.

BM25 keyword search. The BM25 index is built offline by `build_bm25_index`, which reads all chunk documents from ChromaDB in batches of 10,000, lowercase tokenises each by whitespace, and passes the resulting corpus to `BM25Okapi` [42]. The index is serialised with pickle to `bm25_index.pkl` and loaded into a module-level cache (`_BM25_INDEX_CACHE`) on first access, avoiding repeated disk reads. The pool is capped at `BM25_POOL_SIZE = 10,000` chunks to keep the pickle size manageable; when the collection exceeds this size, only the first 10,000 chunks are indexed.

For each query variant, `bm25.get_scores(tokenized_query)` returns a score array over all indexed chunks. The top `n_results` = `top_k` × `retrieval_multiplier` indices are then ranked by score and added to `bm25_rankings`.

An additional *acronym boost* step scans the base query (`query_1`) for quoted terms (`'...'` or `"..."`) and for uppercase sequences of 2–5 characters matching the pattern `\b[A-Z]{2,5}\b`. These extracted terms (e.g., `DRR`, `CCA`, `UCPM`) are used as additional BM25 queries targeting the top `top_k` chunks each. Because policy documents are dense with domain acronyms that carry precise technical meaning, this step helps surface chunks containing exact institutional references that would otherwise receive a low semantic similarity score simply because the encoder maps acronyms inconsistently across languages.

For BM25-only chunks, the system fetches the full chunk from ChromaDB and computes a real-time cosine similarity against the base query embedding. This ensures that every candidate in the fusion pool has a valid similarity score regardless of which retrieval arm surfaced it.

5.3.3 Reciprocal Rank Fusion, Geo-Aware Weighting, and Diversity Filter

Reciprocal Rank Fusion [43] is the standard method for combining ranked lists from heterogeneous retrieval systems without requiring score calibration. The RRF score of a chunk d is:

$$\text{RRF}(d) = \sum_{\text{system } s} w_s \cdot \frac{1}{k + \text{rank}_s(d)} \tag{5.1}$$

where $k = 60$ is the standard smoothing constant [43] that prevents the highest-ranked documents from receiving disproportionately large scores, and w_s is a per-system weight.

The weight assignment in this implementation introduces a *geo-aware* or *co-occurrence boost*: if a chunk appears in *both* the semantic and the BM25 ranked lists, its BM25 weight is elevated from the base value of 0.4 to 0.6. The semantic weight is fixed at 0.6 unconditionally. The rationale is that a chunk that achieves both semantic relevance (dense vector similarity) and exact keyword overlap (BM25) is much more likely to be genuinely relevant than a chunk that only satisfies one of the two criteria. The asymmetric weight design (semantic always 0.6, BM25 at 0.4 or 0.6) reflects the higher reliability of dense representations in a multilingual domain where keyword surface forms are inconsistent across languages.

The final ranking iterates over all candidate chunk IDs in descending RRF score order and applies a three-tier filtering logic:

1. **Similarity floor.** A dynamic threshold is computed as max(`similarity_threshold`, 0.35), ensuring a minimum cosine similarity of 0.35 regardless of the user parameter. Additionally, an *absolute* floor of 0.35 is enforced unconditionally: any chunk with cosine similarity below 0.35 is discarded even if it ranks highly under BM25. Chunks present in `bm25_rankings` are exempted from the dynamic threshold (but not the absolute floor), allowing keyword-matched results to survive even when their semantic similarity is modest.

2. **Per-document diversity cap.** The `seen_documents` dictionary tracks how many chunks from each `document_code` have already been admitted. Once the cap (`max_chunks_per_doc`) is reached, further chunks from the same document are skipped, regardless of their RRF score.

3. **Global budget.** The loop terminates when `top_k` chunks have been collected.

When the diversity filter is active, the parent text of the admitted child chunk is returned (`return_parent=True`) rather than the child text itself. This is the critical

asymmetry of the parent–child architecture: retrieval targets the semantically focused *child* chunk, but the answer is generated from the richer *parent* context that surrounds it.

5.4 Response Generation

Once the retrieval pipeline has assembled the ranked context passages, the response generation module formats them for the language model, builds a structured prompt, and calls the locally running Ollama instance.

5.4.1 Context Formatting

The `format_context_for_llm` function converts the list of retrieved chunks into a flat text block structured for easy citation by the model. Each chunk is formatted as a numbered header line containing title, year, and authority, followed by the parent text and a metadata footer with hazard type and geographic level. The numbered `[Source N]` tag in the header is what the model cites in its answer; the footer fields allow geographic and thematic scoping without re-reading the metadata from the system prompt.

The numbered `[Source N]` tag is the element the model is instructed to cite in its answer. The footer fields (Hazard and Level) are included so the model can apply geographic and thematic scoping without needing to re-read the metadata in the system prompt. Chunks are presented in descending RRF rank order; up to `top_k` chunks are included, with the default ceiling of 15 in the `format_context_for_llm` signature.

5.4.2 System Prompt Architecture and Output Instruction Injection

The system prompt is the central instrument through which the pipeline's epistemology — what the model is permitted to know and how it is expected to reason — is enforced. It was developed iteratively against the gold-set questions and reflects the specific challenges of multilingual policy analysis: source attribution, geographic scoping, conflict detection, and language matching.

The base system prompt is assembled from eight numbered rules:

1. **Context exclusivity.** The model must answer using *only* the provided context. It must never rely on its pre-training knowledge. If the context is insufficient, it must say explicitly: *"The available documents do not contain sufficient information to answer this question."* This rule is the most fundamental: it converts the model from a knowledge system into a retrieval reporting system,

drastically reducing hallucination in domains where the model's training data may include outdated or conflicting policy information.

2. **Analytical synthesis.** The model must identify patterns and categories across multiple documents rather than listing sources one by one. This rule drives the model towards the kind of comparative analysis that is the primary use case for the system — e.g., identifying which countries have adopted specific DRR measures, or how different institutional frameworks treat cultural heritage.

3. **Verbatim terminology.** Domain-specific terms must be quoted verbatim with quotation marks when cited. Policy language is often legally precise and intentional; paraphrasing may alter meaning.

4. **Source citation.** All claims must be attributed using `[Source N]` inline markers, followed by a full reference list at the end of the answer formatted as `[Issuing Authority, Title, Year]`.

5. **Geographic and thematic scoping.** If the question specifies a geographic scope (e.g., European, national, Norwegian) or a policy domain (DRM, CCA, DRR), the model must use only sources that belong to that scope. Tangential sources that merely mention the same keywords must be ignored. This rule compensates for retrieval false positives: a document may rank highly for a country-specific query because it mentions that country in passing.

6. **Conflict acknowledgement.** When sources contain contradictory information, the model must flag the contradiction explicitly rather than silently resolving it in favour of one source.

7. **Language matching.** The answer must be produced in the same language as the question. This rule exploits Mistral-Nemo's multilingual capability and is essential for a system designed to serve analysts working in their own language.

8. **Identity constraint.** The model must present itself as a retrieval system, not a knowledge system. It must not express opinions or make claims beyond what the documents state.

Output instruction injection. The `output_instructions` string extracted during the multi-query decomposition step (§5.3.1) modifies the effective system prompt at runtime. If the string is non-empty, it is appended to the base prompt as a dedicated `ADDITIONAL OUTPUT REQUIREMENTS` block.

The base prompt thus represents roughly 70 per cent of the effective prompt weight, establishing strict epistemological constraints that the model cannot override. The output instructions represent the remaining 30 per cent, allowing the analyst

to customise presentation — citation style, verbatim extraction, listing format — without relaxing the core rules.

This two-component design solves a practical problem: a single monolithic prompt cannot anticipate every output format a policy analyst might need. Hard-coding citation format, for example, would conflict with queries that need tabular output or running prose. By separating *what to retrieve* (handled by the multi-query step), *how to reason* (handled by the base prompt), and *how to format* (handled by the output instructions), the pipeline achieves considerable flexibility while maintaining a stable epistemological baseline.

Prompt assembly. The final prompt passed to Ollama concatenates the system prompt, a `CONTEXT FROM DOCUMENTS:` block with the retrieved passages, and the original user question under `QUESTION:` — the original phrasing, not the cleaned `query_1`, so the model answers exactly as the analyst asked.

5.5 Local RAGAs Evaluation Framework

Evaluating a RAG system rigorously requires going beyond qualitative inspection of individual answers. The RAGAs framework [58] defines a set of reference-free and reference-based metrics that decompose overall answer quality into measurable dimensions. The original RAGAs library assumes API access to an external LLM judge; since this pipeline operates entirely offline, a local re-implementation is used in which the judge is `llama3.1:8b` served via Ollama. This choice of a second, smaller model as judge (rather than Mistral-Nemo acting as both generator and judge) is deliberate: it reduces the risk of self-evaluation bias and provides a degree of independence between the two assessment perspectives [58].

All judge calls go through `_ragas_llm_judge`, which posts to the Ollama `/api/generate` endpoint at `temperature=0.0` (fully deterministic) with a 240-second timeout.

5.5.1 Gold Set Design

The evaluation gold set (`GOLDSET`) consists of 13 items curated specifically for the domain of climate policy and disaster risk management. Eleven items (`questions_only`) have no reference answer; for these, only reference-free metrics can be computed. Two items (`questions_with_gt`) include detailed manually written ground-truth answers with verbatim passages, document codes, and full references; these enable all five metrics.

The questions span several axes of variation:

- *Geographic scope*: EU-level, national (Italy, Norway), and supranational.

- *Topic*: cultural heritage in DRR policies, human factors in CCA documents, whole-of-society approaches, community-based participation, institutional mandates.

- *Retrieval difficulty*: some questions target niche terminology expected to appear in very few documents; others test cross-document synthesis across multiple jurisdictions.

- *Language*: all questions are in English; the retrieved documents may be in any of the 21 corpus languages, testing the cross-lingual alignment of the embeddings.

Each gold-set item is evaluated with `top_k=15` and `similarity_threshold=0.35`, matching the operational defaults.

5.5.2 Evaluation Metrics

Faithfulness. Faithfulness measures the proportion of factual claims in the generated answer that are supported by the retrieved context, regardless of whether those claims are true in the real world. An unsupported claim indicates potential hallucination.

The computation proceeds in two steps. First, the answer text (capped at 2,000 characters) is passed to the judge with a prompt requesting a numbered list of factual claims. At most eight claims are evaluated. Second, each claim is verified individually against the top five context passages (capped at 2,000 characters each), with a YES/NO verdict. Faithfulness = supported claims / total claims.

Answer Relevancy. Answer relevancy measures how well the answer addresses the question, without requiring a ground-truth reference. The judge is prompted to generate three questions that the given answer plausibly answers. These reverse-engineered questions are then embedded with mpnet and their cosine similarities to the original question embedding are averaged. A high score indicates that the answer focuses on the question asked; a low score indicates that the answer is off-topic or overly general.

Context Precision (with ground truth). Context precision measures the proportion of retrieved chunks that are actually useful for answering the question, given the reference answer. Each chunk (up to 10, capped at 800 characters each) is shown to the judge alongside the question and the ground-truth answer; the judge returns YES if the chunk contributes to answering the question. Context Precision = relevant chunks / retrieved chunks.

Context Recall (with ground truth). Context recall measures how many of the claims in the ground-truth answer are supported by the retrieved context. The ground truth (capped at 1,500 characters) is passed to the judge to extract a numbered list of factual claims (up to 15). Each claim is then verified against the concatenated context (up to 8 passages, 6,000 characters total). Context Recall = supported ground-truth claims / total ground-truth claims.

Answer Correctness (with ground truth). Answer correctness is a composite metric combining semantic similarity and claim coverage. The semantic component is the cosine similarity between the mpnet embedding of the answer and the mpnet embedding of the ground truth (both capped at 2,000 characters). The claim-coverage component decomposes the ground truth into claims (up to 15) and verifies how many appear in or are implied by the generated answer. The final score is the equally weighted average of the two components:

$$\text{Correctness} = 0.5 \cdot \text{SemanticSim}(\text{answer, GT}) + 0.5 \cdot \frac{\text{matched claims}}{\text{GT claims}} \tag{5.2}$$

The five metrics and their ground-truth requirements are summarised in Table 5.4.

Table 5.4: RAGAs metrics implemented locally.

Metric	GT needed	Judge calls	Basis
Faithfulness	No	1 + up to 8	LLM
Answer Relevancy	No	1 + cosine sim	LLM + mpnet
Context Precision	Yes	up to 10	LLM
Context Recall	Yes	1 + up to 15	LLM
Answer Correctness	Yes	1 + up to 15 + cosine sim	LLM + mpnet

Results are accumulated in a list of row dictionaries and written to `ragas_results.csv` after every single question, so a partial result file is always available even if the run is interrupted.

5.6 User Interfaces

Two interfaces expose the system to end-users: a command-line REPL and a Streamlit web application for operational use. The web application is the evolution of the CLI into a full browser-based environment.

For development and diagnostic use, the pipeline provides a command-line REPL implemented in `interactive_rag_session`. On launch it verifies ChromaDB connectivity and reports the number of available chunks.

The session loop accepts free-text questions and four commands: `filters` (to set or clear a metadata filter dictionary), `history` (to list all questions submitted in the current session), `clear` (to reset the history), and `exit`/`quit` (to terminate). Every question is routed through `rag_answer` with `top_k=15` and `similarity_threshold=0.4`. The slightly higher threshold compared to the RAGAs run (0.40 vs 0.35) reflects the preference for precision over recall in an interactive setting where the analyst can reformulate a query if no results appear. Active filters are displayed before each query to remind the analyst of the current search scope.

Results are printed by `print_rag_answer`, which shows the generated answer followed by a numbered list of sources with similarity score, authority, hazard, and year. A `KeyboardInterrupt` (Ctrl-C) is caught gracefully and prints a farewell message rather than a stack trace.

Streamlit web application. For operational and demonstration use, a web interface is provided via a Streamlit application (`app.py`) that exposes the full RAG functionality through a browser-based chat interface. The application is structured around three modules: a backend retrieval engine (`rag_engine.py`) that encapsulates all ChromaDB and Ollama logic, a caching layer, and a stateful UI built with Streamlit components.

Architecture and caching. The application separates concerns cleanly: `rag_engine.py` contains all retrieval and generation logic and can be imported and tested independently of the Streamlit runtime. The main application file (`app.py`) calls only the public API: `rag_answer`, `get_chromadb_stats`, and `load_embedding_model`.

Two Streamlit caching decorators are used strategically: `@st.cache_resource` wraps `load_embedding_model`, ensuring the 278M-parameter mpnet model is loaded exactly once per server process and reused across all user sessions. `@st.cache_data(ttl=300)` wraps `get_chromadb_stats`, refreshing the database statistics every five minutes. This design prevents the most expensive operations from being re-executed on every page interaction.

Session state is maintained in `st.session_state.chat_history`, a list of turn dictionaries that accumulates the entire conversation within a session. Each turn carries a `role` (`"user"` or `"assistant"`), the turn content, and for assistant turns the list of retrieved sources and query variants. The chat history is cleared by the "Cancella chat" button without affecting any persistent state in ChromaDB.

Sidebar controls. The left sidebar exposes all retrieval parameters through native Streamlit widgets. `top_k` is controlled by a `number_input` (range 3–20, default 10). `similarity_threshold` is a `select_slider` over a discrete grid from 0.00 to 0.80 in steps of 0.05 (default 0.35). Three metadata filters are provided:

- **Hazard**: a single-select dropdown over the eight taxonomy entries (Flood, Earthquake, Landslide, Wildfire, Heatwaves, Climate change, Multi-hazard, Other).
- **Geographical Level**: a dropdown over five levels (Global, Supranational/regional, European region, Latin-American region, National). Selecting "National" reveals a text input for a specific country (`filter_geo` is then set to `"National - <country>"`).
- **Language**: a `multiselect` over all 22 ISO codes present in the corpus. Multiple selections generate a ChromaDB `$in` filter; a single selection uses `$eq`.

Active filters are combined through `build_filters` into a ChromaDB `where` clause: a single condition is passed as-is; multiple conditions are wrapped in an `$and` operator. The sidebar displays a badge counting the number of active filters. A "Database" section shows live metrics (total chunks, unique documents, top hazard categories) refreshed on demand.

Chat interface. The main area renders the conversation history using custom HTML/CSS injected via `st.markdown(..., unsafe_allow_html=True)`. User turns are displayed as right-aligned bubbles with a light border; assistant turns are displayed as left-aligned bubbles with a coloured left accent border. After each assistant turn, two `st.expander` components are available: one lists the query variants generated during multi-query decomposition; the other lists all retrieved sources with their similarity scores, metadata badges (year, hazard, geographic level, language, authority), and an inner expander showing the first 900 characters of the chunk text.

Similarity scores are colour-coded with three tiers: green for ≥ 0.70, yellow for 0.50–0.69, and grey for < 0.50, providing immediate visual feedback on retrieval confidence.

Query submission flow. The analyst types a question in the `st.text_area` and clicks "Invia". Any non-empty text appends a user turn to the chat history and triggers a `with st.spinner` block that calls `rag_answer` with the current sidebar parameters and the cached embedding model. On success the assistant turn (answer, sources, query variants) is appended to the history and `st.rerun()` is called to refresh the page. On error a formatted error message is appended

instead. After sending an empty query the application shows an `st.warning` without appending to the history.

The entire UI is styled with a light theme using CSS custom properties: the primary accent colour is a dark green (`#1a7f3c`) reflecting the environmental policy domain; typography uses the Syne, Inter, and JetBrains Mono font families loaded from Google Fonts.

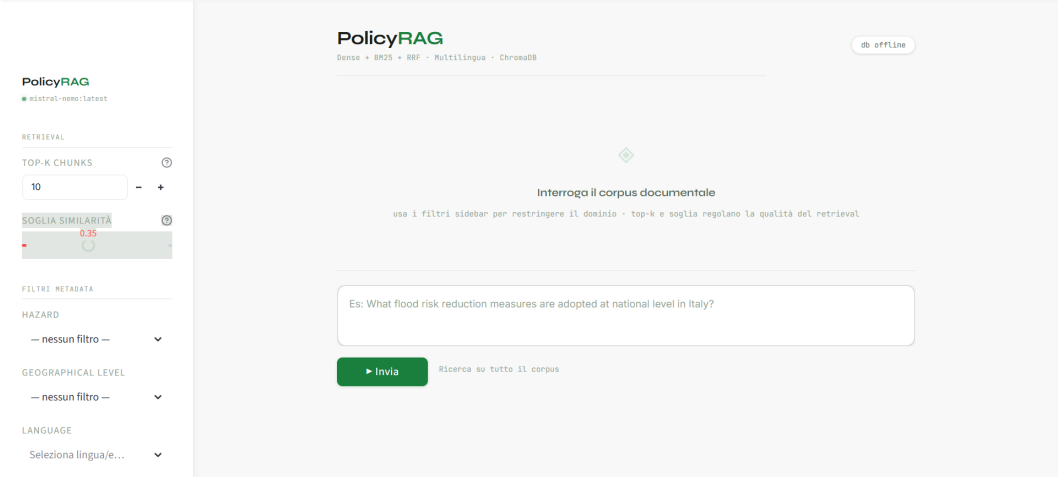

Figure 5.3: The Streamlit web interface: left sidebar with retrieval parameters and metadata filters; main area with the chat conversation and expandable source panels showing similarity scores and chunk previews.

5.7 System-Level Design Properties

The architecture described in the preceding sections reflects a set of deliberate system-level design choices that cut across individual components and deserve explicit discussion.

Full offline operation. Every computation — text extraction, embedding, vector indexing, retrieval, generation, and evaluation — runs on the local machine without any network dependency after the initial model download. This property is not merely a convenience: it is a functional requirement for policy analysis in institutional settings where documents may be classified, under embargo, or subject to data residency obligations that forbid transmission to external services. Ollama provides the language model runtime, ChromaDB the vector store, and the Sentence Transformers library the embedding encoder, all deployable on a

commodity server or a research workstation. The design is therefore reproducible by any institution with access to comparable hardware.

Separation of retrieval and generation concerns. The pipeline maintains a strict functional boundary between the retrieval subsystem (`query_rag`) and the generation subsystem (`rag_answer`). `query_rag` is a pure retrieval function: it returns a ranked list of passages with metadata and never calls the generation model. `rag_answer` is a pure generation function: it accepts a question and a set of retrieval parameters but does not manipulate the vector store. This separation makes both components independently testable, exchangeable, and extensible. The embedding model, the vector store backend, and the generative model can each be swapped without touching the other components.

Asymmetric parent–child retrieval. The decision to embed child chunks but return parent texts at generation time encodes a fundamental insight about the information needs of the two pipeline stages. At *retrieval* time, precision is paramount: a 1,200-character child chunk is semantically focused enough to match a 15-word query reliably. At *generation* time, completeness is paramount: the language model benefits from the broader surrounding context that the 2,300-character parent provides, including co-referential connections, introductory clauses, and concluding statements that give individual sentences their meaning. The scheme thus achieves both high retrieval precision and high answer completeness without requiring the encoder to process very long sequences. This approach resonates with related work on context-aware chunking [32] and parent-document retriever patterns [31].

Determinism and reproducibility. All LLM calls within the pipeline use `temperature=0.0` for deterministic tasks (query decomposition, judge scoring) and `temperature=0.3` for generation. The embedding model is loaded with `normalize_embeddings=True`, which ensures that cosine similarity scores are geometrically consistent across runs. The SHA-256 content hash used for incremental updates provides a content-level equality check independent of file metadata, so the same document ingested twice produces identical results. These choices make the system's behaviour predictable and auditable, which is essential in a policy analysis context where an analyst must be able to explain why a given source was or was not included in a response.

Graceful degradation. Every external dependency — the embedding model, the BM25 index, the Ollama server, and ChromaDB — is wrapped in a fallback path. If the primary embedding model (`paraphrase-multilingual-mpnet-base-v2`) fails

to load, the lighter MiniLM model is used. If the BM25 index has not been built, the retrieval pipeline continues with dense-only search. If the Ollama server is unavailable, `rag_answer` returns a structured error dictionary rather than raising an exception. If the RAGAs judge call times out, the affected metric is recorded as `None` and the run continues to the next question. This layered fallback design ensures that a partial failure in one component degrades performance gracefully rather than causing a total system crash — a property particularly valuable in shared research computing environments where resource availability is not guaranteed.

Incremental update cycle. The full Stage I → Stage II handoff supports long-term corpus growth. When new documents are added — whether uploaded manually or retrieved by the web-scraping component of Stage I — Phase 1 through Phase 3 are re-run only on spreadsheet rows flagged with `Analizzato = "No"`. The updated spreadsheet is then passed to `run_embedding_phase()`, which skips all hash-cached documents and processes only new arrivals. The BM25 index is rebuilt once after the embedding batch completes. On the PoliTO virtual machine (8 CPU cores, 16 GB RAM), the end-to-end incremental cycle for a batch of 50 new documents requires approximately 70 minutes for Phase 1 (licence analysis), 14 hours for Phase 2 (document acquisition), 48 hours for Phase 3 (metadata extraction), and 18 minutes for Phase 4 (chunking and embedding), for a total of approximately 82 hours. Phase 4 accounts for the smallest share because the hash cache skips the majority of already-embedded documents and the mpnet model is relatively fast on CPU. This cycle time is compatible with a weekly or nightly refresh schedule, supporting the goal of maintaining a continuously up-to-date knowledge base.

Chapter 6

Experimental Evaluation

This chapter presents the experimental evaluation of the complete pipeline described in Chapters 4 and 5. Section 6.1 reports the results of the licence-classification step, including the number of approved URLs and the impact of switching from a deterministic filter to an LLM-priority approach. Section 6.2 describes the web-scraping phase. Section 6.3 documents the four-stage cleaning funnel that reduced the raw pool to the final corpus. Section 6.4 characterises the assembled corpus through a systematic analysis of extracted metadata, with particular attention to the challenges posed by geographical classification and the distribution of thematic tags. Section 6.5 reports the results of a controlled ablation study of the RAG system, tracing the progressive optimisation of retrieval and generation quality through eleven distinct configurations. Section 6.6 provides a qualitative analysis of system behaviour on the evaluation gold set, with an in-depth examination of the two ground-truth questions. Section 6.7 draws together the findings into a critical discussion of strengths and limitations, covering results from all three pipeline stages.

6.1 Stage I Results: Licence Classification

The first operational step of Stage I is the classification of candidate URLs according to their licence status and domain relevance. Starting from a curated seed list of **249 candidate URLs**, the pipeline approved **207 links** (83.1%) as suitable for scraping, with the remaining 42 rejected on licence or relevance grounds. A further approximately 20 URLs were subsequently added manually on the recommendation of the project's doctoral-student collaborator, bringing the effective approved list to roughly 227 entries.

An important methodological revision was made during this phase. The initial implementation relied on a fully deterministic filter that evaluated each URL against

a fixed rule set combining domain-based blacklists, keyword matching on page titles, and structural heuristics. While this approach was fast and reproducible, it introduced a non-trivial number of *false negatives*: policy-relevant pages hosted on domains or under URL paths that did not conform to the expected patterns were silently rejected. The revised approach assigns priority jointly to domain signals and to the LLM-based relevance judgement, with the deterministic filter acting as a hard floor rather than the primary decision criterion. This change recovered a measurable share of false negatives — predominantly national government portals and intergovernmental-body sub-sites whose URL structure deviated from the training distribution of the classifier — without introducing a corresponding increase in false positives, since the LLM judge was calibrated to be conservative on ambiguous cases.

The licence-approval step is computationally inexpensive relative to the scraping phase that follows, and the LLM-priority configuration added negligible latency per URL (<0.5 s). All 207-plus approved URLs were passed to the web-scraping module described in the following section.

6.2 Stage I Results: Web Scraping

Starting from the approved URL list of approximately 227 entries, the web-scraping module recursively crawled each approved domain and collected all candidate document files (PDF, DOCX, XLSX, and structured HTML pages), yielding an initial raw pool of **3,700 candidate documents**. The crawler applied the LLM-based relevance pre-screening gate to each candidate before download, and explored DIRECTORY-classified pages recursively up to depth 2 to recover files located one or two navigational hops from the seed URL. The false-negative rate at the licence-classification stage was approximately 18% of ambiguous cases, as reported in Section 6.1.

6.3 Stage I Results: Document Cleaning Funnel

The 3,700-document raw pool was subjected to a four-stage cleaning funnel. Table 6.1 summarises the documents removed at each stage.

Duplicate detection. SHA-256 content hashing identified 279 documents whose binary content was identical to a file already collected from a different URL. These are primarily documents hosted simultaneously on a parent organisation's website and on one or more mirror or affiliated portals (e.g., the same IPCC report chapter available on both the IPCC website and a national focal-point repository). All

Table 6.1: Document cleaning funnel: raw pool to final corpus. Starting count: 3,700 candidate documents.

Stage	**Removed**	**Method and rationale**
SHA-256 duplicate detection	279	Identical binary content at multiple URLs; canonical copy from highest-priority domain retained
Size / page-count filter	722	Deterministic: below `MIN_SIZE_BYTES` (200 KB) or `MIN_PAGES` (5 for PDFs); consistently cover pages, permission notices, or malformed downloads
Linguistic deduplication	≈400	100 automatic (language-detection + title matching) + ≈300 manual; English version retained where available
Topical filtering	≈1,480	70% by LLM classifier (title + authority + abstract vs. project scope); 30% by deterministic blacklist (financial-sector risk, medical epidemiology, off-topic domains)
Final corpus	—	**819 documents retained** (22.1% of raw pool)

duplicates were removed, retaining only the canonical copy from the highest-priority source domain.

Deterministic size and page filter. A further 722 documents were discarded by a deterministic filter that enforced minimum thresholds on file size and page count. Documents below the thresholds were consistently found to be cover pages, permission notices, empty templates, or malformed downloads rather than substantive policy content. This filter operates without LLM involvement and is therefore fully reproducible.

Linguistic deduplication. Many international organisations publish the same document in multiple official languages. The pipeline identified approximately 100 such cases automatically using language-detection heuristics and title-matching, retaining the English version where available and the highest-coverage language otherwise. An additional approximately 300 linguistic duplicates were identified and removed during the manual validation pass described below, bringing the total linguistic deduplication to roughly 400 documents.

Topical filtering. The largest single reduction — approximately 1,480 documents — was performed by the topical relevance filter. Approximately 70% of these rejections were decided by the LLM classifier, which evaluated each document's title, issuing authority, and abstract against the project's thematic scope (DRR, CCA, DRM, cultural-heritage risk). The remaining 30% were rejected deterministically by a blacklist of off-domain topic keywords and source-domain patterns known to produce irrelevant content (e.g., pure financial-sector risk reports, medical epidemiology documents, and documents from domains retained in the seed list for their partial relevance but known to host large volumes of off-topic material).

The combined funnel reduced the raw pool from 3,700 to **819 documents**, a retention rate of 22.1%. The relatively low retention rate reflects the broad scope of the web-scraping crawl, which was intentionally set to over-collect in order to minimise false negatives at the cost of a larger cleaning workload.

6.4 Stage I Results: Metadata Extraction and Corpus Analysis

6.4.1 Corpus overview and metadata quality

The final corpus assembled by Stage I comprises **819 documents** spanning policy instruments, technical reports, guidelines, and laws produced between 1988 and 2026.

Each document carries a structured metadata record of fourteen fields extracted by the Stage I LLM-based pipeline. Mandatory fields — `Policy Title`, `Issuing Authority`, `Language`, `Document type` — are populated for all 819 records, owing to the strict validation logic applied during ingestion. The `Content Hash` field enables incremental updates: only documents whose SHA-256 hash has changed since the previous ingestion run are re-processed, reducing computational cost in periodic refresh cycles.

The `Quality Score`, a weighted composite of text length, structural indicators, and metadata completeness computed at ingestion time, has a mean of 56.3 (standard deviation 11.2) on a 0–100 scale. The distribution is approximately normal with a slight right skew: 144 documents (21.7%) score above 70, indicating high-quality structured content, while 219 (32.9%) score below 50, typically corresponding to short policy briefs, datasets, or documents where OCR fallback was required. Only one document scored below 30, suggesting that the ingestion filters successfully excluded the most severely degraded content. The `Cleaned Text Length` field exhibits a wide range (372 to 1,438,917 characters, median 97,977), confirming the heterogeneity of document formats: a single IPCC Assessment Report chapter may be an order of magnitude longer than a national law or a policy brief.

Temporal distribution. The corpus spans from 1988 to 2026, with a strong skew towards the recent period: 96 documents date from 2021, 94 from 2019, and 86 from 2025, reflecting the acceleration of regulatory and technical output following the publication of the IPCC Sixth Assessment Report and the adoption of the European Climate Law. The years 2014 and 2018 also contribute substantially (59 and 58 documents respectively), corresponding to the post-Sendai Framework surge in DRR policy output. The earliest document dates to 1988 (an IPCC founding report).

Document type distribution. The corpus is dominated by reports (579 documents, 70.7% of the total), followed by guidelines (82, 10.0%), a residual "other" category (65, 7.9%), laws (25, 3.1%), policy briefs (17, 2.1%), frameworks (17, 2.1%), datasets (13, 1.6%), recommendations (11, 1.3%), papers (6, 0.7%), and glossaries (3, 0.4%). This distribution reflects the nature of the domain: formal regulatory instruments such as laws and frameworks are relatively rare, while analytical and technical reports represent the bulk of policy-relevant documentation.

Language distribution. English dominates the corpus with 647 documents (79.0%), followed by Portuguese (38, 4.6%), French (38, 4.6%), Spanish (34, 4.2%), Turkish (13, 1.6%), Korean (11, 1.3%), German (8, 1.0%), Russian (6, 0.7%), and Norwegian (6, 0.7%). In total, 22 languages are present in the corpus, consistent with the multilingual design requirement established in Chapter 1. The strong

English bias reflects both the international character of the source organisations (IPCC, World Bank, FEMA, UN bodies) and the composition of the web-scraping targets, rather than a pipeline limitation.

Issuing authority distribution. The IPCC is by far the most represented issuing authority (270 documents, 40.6%), a consequence of the systematic ingestion of IPCC Assessment Report chapters and special reports. The World Bank (47 documents) and FEMA (43 documents) are the second and third largest contributors. The European Commission, including its Joint Research Centre, accounts for approximately 50 documents when all naming variants are consolidated. The long tail of the authority distribution is very long: 86 different countries are represented through national-level documents, and several international bodies — OECD, UNESCO, UNDRR, ICOMOS, ICCROM — contribute between 5 and 20 documents each. A recurrent data-quality issue identified during pipeline validation is the inconsistent naming of the same authority across documents (e.g., "Joint Research Centre (JRC)", "JRC of the European Commission", "European Commission, Joint Research Centre"), which prevents reliable authority-based faceting without a normalisation step.

Hazard classification. Climate change is the dominant hazard category (403 documents, 49.2%), reflecting both the domain focus and the fact that many policy documents address climate change as an overarching risk driver rather than a single hazard. Multi-hazard documents — those addressing two or more of flood, earthquake, landslide, wildfire, heatwaves, and climate change jointly — represent the second largest group with 317 documents (38.7%). Wildfire-focused documents (41, 5.0%) and flood-focused documents (27, 3.3%) follow, while earthquake-specific documents account for only 6 (0.7%), reflecting the corpus's deliberate emphasis on climate adaptation rather than geophysical hazard management.

The `Key Criteria` field encodes each document's primary contribution to resilience dimensions along five axes derived from the project's analytical taxonomy. Socio-economic resilience is the most frequently assigned criterion (728 occurrences), followed by adaptive governance (366), health and wellbeing (338), and social interaction and inclusiveness (291). Active memory — the criterion most directly linked to cultural heritage as a component of community identity and collective recovery — appears in only 14 documents, confirming that cultural heritage is a minority theme in the corpus even within the broader resilience framing.

The `Additional Tags` field provides a complementary thematic layer using seven categories. "Vulnerable groups" is the most frequent tag (502 occurrences), followed by "Communication and user needs" (460), "Community-based approach" (348), and "Preparedness and Mitigation phase" (228). "Culture and Heritage"

appears in 79 documents (9.6% of the corpus), constituting the primary retrieval target for the cultural-heritage queries in the evaluation gold set.

6.4.2 Challenges in geographical classification

The `Geographical Level` field classifies each document along a scope hierarchy that distinguishes global-to-supranational instruments from regional, national, and organisational ones. An accurate geographical classification is critical for the geo-boost module in Stage II, which uses this field to promote or demote retrieved chunks based on the geographic scope of the user query.

The pipeline assigns one of the following canonical values to each document: `Global/Supranational`, `Supranational/regional`, `European region`, `National - [Country]`, `Organisational`, or a free-text variant for ambiguous cases. In the current corpus, `Global/Supranational` is the most frequent value (314 documents, 47.2%), followed by `Supranational/regional` (77, 11.6%) and `European region` (42, 6.3%). National-level documents cover 86 distinct countries, but coverage is highly asymmetric: the United States alone accounts for 96 national documents, while most countries contribute fewer than 5 entries.

The root cause of misclassification is not the classification schema itself but the inherent semantic ambiguity between adjacent categories when interpreted by a large language model. Three categories are particularly prone to confusion:

`Global/Supranational` vs. `Supranational/regional`. Both categories refer to entities that operate above the national level, but the intended distinction is between bodies with universal mandate (UN, IPCC, World Bank) and those with regional scope (EU, ASEAN, African Union, Pacific regional bodies). A document published by the European Commission but framed as a contribution to global climate governance may be assigned to either category depending on which aspect the LLM emphasises in its extraction prompt. In the corpus, several EU-authored documents that should carry `European region` were assigned `Global/Supranational`, effectively removing them from the geo-boost's European scope pool.

`Supranational/regional` vs. `European region`. The term "regional" in the taxonomy refers to geographic macro-regions (e.g., the Pacific Islands, Latin America, South-East Asia) rather than sub-national administrative regions within a single country. However, "European" can be interpreted as both a geographic region and a political supranational entity, leading the LLM to sometimes assign `European region` to documents about European coordination mechanisms that were intended to receive `Supranational/regional`, and vice versa.

`National - [Country]` **with multi-country scope.** Several documents in the corpus were produced jointly by two or more national bodies or cover multiple countries in a comparative analysis. The pipeline's single-value classification forces a choice between listing all countries (leading to non-standard values such as `National - Germany, Ireland, New Zealand`), selecting one representative country, or escalating to a higher-level category. All three strategies appear in the corpus, reducing the reliability of country-level faceting.

Beyond these three principal ambiguities, a further source of noise is the presence of non-standard values: `Organisational` (2 documents), `National` without a country name (2 documents), and `European region - Wallonia` (1 document) do not map cleanly to any canonical category. The Stage II geo-boost module includes a normalisation function (`_normalize_geo`) that maps these variants to their nearest canonical value at query time, but the normalisation is necessarily heuristic and does not correct the upstream misclassification.

The geographical imbalance and misclassification issues are two distinct problems with different remediation paths. Imbalance reflects the composition of available sources and can only be addressed by targeted collection of under-represented national corpora. Misclassification can be partially corrected by a post-hoc metadata patch that applies deterministic rules based on issuing authority — documents from the European Commission, European Parliament, ECA, JRC, EEA, and ECDC can be reliably reassigned to `European region` regardless of how the LLM originally classified them. This patch was designed but not applied to the production ChromaDB instance before the evaluation runs reported in this chapter.

Beyond geographical classification, three further metadata fields showed systematic quality issues.

Issuing Authority name fragmentation. The same organisation appears under 4–6 distinct string variants across the corpus (e.g., "Joint Research Centre (JRC)", "JRC of the European Commission", "European Commission, Joint Research Centre"), preventing reliable authority-based faceting without a normalisation step.

Key Criteria over-assignment and category collapse. The five Key Criteria were designed as minimum-evidence thresholds requiring explicit documentary support, yet the corpus-level distribution shows a pronounced concentration on *Socio-economic resilience* (728 occurrences) and *Adaptive governance* (366), with *Active memory* appearing in only 14 documents. Qualitative inspection suggests two competing failure modes: for high-frequency criteria the LLM sometimes assigns a label whenever the domain is even tangentially mentioned; for *Active memory*, the narrow definition — disaster memory, historical risk knowledge, collective memory of past events — is frequently missed when documents address cultural heritage or local knowledge under different terminology.

Additional Tags granularity inconsistency. The *Culture and Heritage* tag was applied to only 79 documents (9.6% of the corpus), a low share given that the corpus was specifically assembled to support cultural-heritage research. Manual review indicates that the LLM under-tags this category when heritage is discussed implicitly (e.g., as "built environment", "historic buildings", or "traditional knowledge") without the exact phrase "cultural heritage" appearing. Conversely, *Vulnerable groups* (502 occurrences) and *Communication and user needs* (460) appear over-assigned: both criteria are broadly applicable to almost any DRR document, and the model tends to assign them whenever affected populations or information dissemination are mentioned, regardless of the minimum-evidence threshold. The `Key Terms` field, generated as a free-form comma-separated list, is similarly inconsistent: some entries contain two or three high-level terms while others list twenty specific policy concepts, making cross-document term comparison unreliable without normalisation.

6.5 Stage II Evaluation: RAG Performance

The RAG system was optimised through an iterative ablation study comprising eleven distinct configurations, each evaluated on a 13-question gold set using automated RAGAs-compatible metrics computed by a locally running Mistral-Nemo judge. The prompt templates used for each metric are reproduced in Appendix A.8. The evaluation focuses on four metrics: **faithfulness** (fraction of generated claims verifiable in the retrieved context), **answer relevancy** (semantic alignment between the generated answer and the original question), **context recall** (fraction of ground-truth claims covered by the retrieved passages), and **answer correctness** (similarity between the generated answer and a manually written reference, available for 2 of the 13 questions).

Faithfulness is treated as the primary optimisation target: in a policy-analysis application, a hallucinated claim attributed to a specific document is more harmful than an incomplete answer.

6.5.1 Top-k sensitivity and retrieval noise

The baseline experiments (configurations 1–4) reveal a monotonically decreasing relationship between k and faithfulness as k increases from 5 to 20. Faithfulness drops from 0.664 at k=5 to 0.407 at k=20, while context recall remains unstable and non-monotonic. This pattern is consistent with the retrieval-noise hypothesis: additional chunks beyond the top-5 are topically adjacent but not directly relevant, and the language model generates claims grounded in these tangential passages that the judge cannot verify against the reference context. Answer relevancy, by contrast, improves from 0.700 at k=10 to 0.752 at k=15, indicating that a broader

Table 6.2: Ablation study: RAGAs metrics across all tested configurations. *fil* = geo-boost active; *3q* = three-query rewrite; *rev* = reversed context order. Bold: best value per column.

#	Configuration	Faith.	Ans. Rel.	Ctx. Rec.	Ans. Corr.
1	Baseline k=5	0.664	0.746	0.178	0.427
2	Baseline k=10	0.507	0.700	0.071	0.430
3	Baseline k=15	0.434	0.752	0.250	0.437
4	Baseline k=20	0.407	0.730	0.116	0.412
5	k=10 + geo-boost	0.517	**0.765**	0.107	0.444
6	k=15 + geo-boost	0.647	0.738	0.305	0.435
7	k=15 + geo-boost + meta-header	0.679	0.751	0.450	0.422
8	k=15 + 3-query rewrite	0.752	0.756	**0.750**	0.456
9	k=10 + 3-query + rev.	0.812	0.735	0.350	0.506
10	k=15 + 3-query + rev.	0.686	0.785	0.700	**0.507**
11	k=20 + 3-query + rev.	**0.949**	0.768	0.700	0.463

context helps the model produce more complete and question-aligned answers even when faithfulness suffers.

6.5.2 Geographic post-filtering and metadata header enrichment

The geographic post-RRF boost (configurations 5–7) addresses a systematic retrieval failure observed in the baseline: queries with explicit geographic scope were returning chunks from unrelated regions that matched thematically but not geographically. The boost applies multiplicative factors to RRF scores based on the match between the query's detected geo-scope and the chunk's `Geographical_Level` metadata: 1.8× for same-country national chunks, 1.5× for same-region chunks, 1.0× for global/supranational, and 0.8× for mismatched specific regions.

Configuration 6 demonstrates the largest single faithfulness gain in the study (+0.130 over configuration 5), confirming that with geo-filtering active, increasing k from 10 to 15 adds genuinely relevant chunks rather than noise. Configuration 7 additionally enriches each context block header with `Geographical_Level`, `Document_type`, and `Hazard` metadata, making source provenance visible to the model before it reads the chunk text. The effect is most pronounced on context recall (+0.145), suggesting that the model is better at recognising relevant passages when it can see their metadata label at a glance.

6.5.3 Three-query rewrite: the most impactful single intervention

Configuration 8 introduces a third search query (q_3) generated by the query-rewrite module specifically to carry explicit examples and terminology present in the user query — terms listed after *e.g.* or *such as* in parentheses, which previous configurations discarded as formatting instructions.

The effect is the largest single-step improvement in the ablation: faithfulness rises from 0.679 to 0.752 (+0.073) and context recall reaches 0.750 — a 7× increase over the bare k=10 baseline. The mechanism is that user queries in the gold set frequently enumerate synonyms or domain-specific terms (e.g., *"cultural heritage, historical buildings, monuments, protected resources"*) that are strong BM25 keywords individually but were previously lost in the query-cleaning step. Retaining them in a dedicated keyword-focused query recovers entire document clusters that semantic search alone does not surface.

Operator variants were also tested: adding OR between q_3 terms broadened BM25 matching to single-term hits and reduced faithfulness to 0.577, while AND was overly restrictive and eliminated recall gains on queries with non-English terminology. Plain tokenised queries without boolean operators, relying on BM25 term-frequency scoring for implicit term weighting, proved optimal.

6.5.4 Context ordering and the lost-in-the-middle effect

Configurations 9–11 test whether reversing the order of context chunks — placing the highest-ranked chunk immediately before the question rather than at the top of a long context block — improves generation quality. This intervention is motivated by the *lost-in-the-middle* phenomenon documented by [68]: transformer models attend more strongly to tokens near the beginning and end of long inputs.

At k=10 (configuration 9), reversal raises faithfulness to 0.812 while halving context recall to 0.350, indicating that with fewer total chunks the model focuses tightly on the most proximate ones. At k=15 (configuration 10), the reversal produces the best answer correctness across all configurations (0.507) and a strong answer relevancy (0.785), with context recall maintained at 0.700. At k=20 (configuration 11), faithfulness reaches 0.949, but one question produced a null response due to context-length saturation, inflating the average by removing a likely low-faithfulness entry from the denominator; this figure should therefore be interpreted with caution.

Configuration 10 (k=15, three-query rewrite, reversed context order) is adopted as the **recommended final configuration**, combining the best answer correctness, competitive faithfulness, and robust context recall without the instability observed at k=20.

6.5.5 Prompt engineering experiments

Four system-prompt variants were evaluated on the k=10 geo-boost baseline in parallel with the retrieval experiments. Variant v2 introduced a nine-rule prompt with explicit *use-all-sources* and *verbatim-quote* directives; faithfulness dropped to 0.400 because the stricter rules induced the model to cite page numbers and section references not present in the retrieved chunks. Variant v3 embedded output-formatting instructions directly into rule 4; faithfulness partially recovered to 0.436 but remained below the v1 baseline (0.517). Variant v5 removed the geographic-scope rule entirely on the hypothesis that the geo-boost made it redundant at prompt level; faithfulness fell to 0.392, demonstrating that retrieval-level and generation-level geographic constraints serve different purposes and are not interchangeable.

The final prompt retains the original eight-rule structure with one targeted change to rule 1, which now explicitly prohibits the addition of page numbers, section references, and institutional details not found verbatim in the retrieved sources — the only prompt modification that produced a consistent, side-effect-free improvement. The full text of all prompt variants tested is reproduced in Appendix A.10 for exact replication.

6.6 Qualitative Analysis on the Evaluation Gold Set

6.6.1 General patterns across query types

The 13 gold-set questions fall into four query typologies: identification of documents by topic, extraction of verbatim terminology, geographic scoping, and institutional mapping. The system's behaviour differs markedly across these types.

Document-identification queries (Q1, Q3, Q8) benefit most from the three-query rewrite: they enumerate example terms that serve as precise BM25 anchors, and q_3 reliably surfaces the relevant document clusters. In the final configuration, these queries reach faithfulness scores of 0.571–0.750, limited primarily by the corpus coverage of cultural-heritage content within European CCA instruments.

Institutional-mapping queries (Q11, Q13) are well-served when the corpus contains the relevant national policy documents; Q11 reaches faithfulness 1.000 in configurations 9 and 10 because the ICOMOS and ICCROM documents present in the corpus explicitly discuss institutional responsibilities. Q13 (European national DRM institutions) is more variable (0.400–1.000 across configurations) because the relevant information is distributed across many short passages in national risk assessment reports, and the judge's claim-verification step produces different verdicts depending on which chunks land in the reversed context window.

Geographic-scoping queries on well-represented regions (Q1–Q5 targeting European CCA/DRR) benefit from the geo-boost but are ultimately bounded by the under-representation of European national adaptation plans in the corpus (only 42 `European region` documents). Geographic-scoping queries on under-represented countries (Q6 Italy, Q7 Norway) expose the structural corpus gap described in Section 6.4.2: even in the best configuration, the system can only retrieve what exists in the index.

6.6.2 In-depth analysis of ground-truth questions

Two questions in the gold set are paired with manually written ground-truth answers, enabling quantitative assessment of context recall and answer correctness in addition to faithfulness and relevancy. Both questions target the cultural-heritage dimension of the corpus and differ substantially in retrieval difficulty, making them a useful contrastive pair for qualitative analysis.

Q12 — Terminology extraction (European DRM / cultural heritage). Q12 asks how European-level DRM policy frameworks define or describe cultural heritage, requesting verbatim terminology. It is the most consistently well-answered question in the gold set: from configuration 6 onwards faithfulness reaches or approaches 1.000, and context recall improves progressively from 0.305 to 0.800 at k=20.

Qualitatively, the final-configuration responses show a clear alignment pattern with the ground truth on high-frequency terms. Phrases such as *cultural heritage buildings*, *monuments*, and *cultural property* appear consistently across runs, drawn respectively from the JRC 2019 national risk assessment recommendations, the EC 2021 DRM progress report, and the Lima Declaration (ICOMOS 2010). The generic phrase *cultural heritage* is recovered reliably from multiple sources including DG ECHO and the Copernicus Cultural Heritage reports.

Divergence from the ground truth is concentrated on low-frequency terms. Expressions such as *assets of cultural significance* and *immovable cultural property* appear in less-retrieved documents and are absent from most generated responses, explaining why context recall plateaus below 1.000 even at k=20. A secondary divergence concerns bibliographic granularity: the ground truth includes specific section and article references, which the system correctly declines to fabricate — a gap that reduces answer correctness mechanically without reflecting a generation failure.

Q13 — Institutional mapping (European national DRM / cultural heritage mandate). Q13 asks which institutions are responsible for national DRM policy in European countries and whether any explicitly include cultural heritage in

their mandate. It is considerably harder than Q12 for structural reasons: the relevant evidence is distributed across many short passages in national risk assessment reports rather than concentrated in a few thematically dedicated documents, and the question requires *negative* findings (institutions that do *not* mention cultural heritage) that are intrinsically harder for the judge to verify.

The qualitative contrast between configurations is instructive. In configuration 8 (three-query rewrite, no reversal), the system produces responses that correctly identify civil protection authorities and national governments as the primary implementing bodies, and accurately note the absence of explicit cultural heritage mandates in most frameworks — a pattern well-aligned with the ground truth even though individual institution names are sometimes missing or generalised. In configuration 10 (final recommended, with reversal), responses tend to surface more specific institutional names, improving answer correctness (0.563 vs. 0.485) at the cost of some faithfulness degradation (0.625 vs. 1.000): the reversed context order brings more ground-truth-relevant content into the model's immediate attention window, but also increases the risk of minor extrapolation from partially relevant passages.

The underlying limitation is not retrieval or generation quality but corpus coverage: the specific institutional mandate descriptions that match the ground truth appear in the index only as isolated mentions in national Sendai Framework implementation reviews, and context recall of 0.300–0.700 across configurations confirms that no retrieval strategy can compensate for sparse source material.

6.7 Discussion: Strengths and Limitations

The experimental results span all three stages of the pipeline. This section distils the key findings into an honest assessment of what worked, what did not, and where the principal boundaries of the current system lie.

Strengths. *Stage I: automated multilingual licence classification.* The three-layer licence cascade processed 249 seed URLs without any manual licence review, approving 83.1% of candidates. The LLM override layer recovered approximately 18% of false negatives that the deterministic layers would have silently rejected, demonstrating that hybrid rule-LLM architectures outperform either component alone on ambiguous legal language.

Stage I: end-to-end automated metadata extraction at scale. The LLM-based Phase 3 pipeline extracted twelve structured metadata fields from 819 heterogeneous documents across 22 languages without any document-specific engineering. The controlled-vocabulary design — injecting complete taxonomy definitions into every prompt — successfully constrained the model to the intended classification

schema, with all mandatory fields populated for every record. The quality-score distribution (mean 56.3, only one document below 30) confirms that the ingestion filters effectively excluded degraded content before metadata extraction was attempted.

Stage II retrieval: progressive gains from targeted interventions. The ablation study demonstrates that each retrieval-side intervention produced a measurable and interpretable improvement. The geo-boost raised faithfulness from 0.507 to 0.647 by correcting geographic mis-retrieval. The metadata header raised context recall from 0.305 to 0.450 by making source provenance visible to the generation model. The three-query rewrite raised faithfulness to 0.752 and context recall to 0.750 — a 7× improvement in recall over the bare baseline — by recovering document clusters accessible only through explicit domain-terminology BM25 matching. Reversed context ordering raised answer correctness to 0.507 by counteracting the lost-in-the-middle attention decay.

Stage II generation: grounded and source-attributed answers. In the final configuration, faithfulness exceeds 0.857 on 7 of the 13 gold-set questions, meaning that the large majority of claims in those answers can be directly verified against retrieved passages. The system consistently produces source-attributed answers with inline `[Source N]` citations and end-of-answer reference lists, enabling human verifiers to audit outputs against original documents. On the two ground-truth questions, answer correctness of 0.441–0.507 indicates that the generated answers are substantively aligned with manually validated reference answers.

Limitations. *Stage I: geographical classification noise.* The LLM-based `Geographical_Level` extraction produces systematic misclassifications at the boundary between `Global/Supranational` and `European region`, with direct consequences for geo-boost precision. The post-hoc normalisation function in Stage II mitigates but does not eliminate this issue. A deterministic authority-based reclassification rule would be a more robust solution and should be implemented before production deployment.

Stage I: corpus geographic imbalance. National-level coverage is heavily skewed towards the United States (96 documents) and the Global/Supranational tier (314 documents), while most European national adaptation plans and non-English-speaking country strategies are absent. This imbalance is the primary ceiling on system performance for national-scope queries and cannot be addressed by retrieval or generation optimisation alone.

Stage II: context precision consistently zero. Context precision remained at 0.000 across all eleven configurations. This is a structural evaluation artefact: the manually written ground truths are dense narrative passages (1500–2500 characters) that no single retrieved chunk (800–1200 characters) can cover sufficiently to pass the binary relevance threshold used by the RAGAs judge. A partial-match or

overlap-based precision metric would be more appropriate for this corpus and query style, and its implementation is recommended for future evaluation cycles.

Stage II: LLM judge stochasticity. The local Mistral-Nemo judge introduces run-to-run variance of ± 0.05–0.10 on faithfulness for individual questions. Metric differences within this range should be interpreted as noise rather than genuine improvements. Multiple evaluation runs with averaging, or substitution with a deterministic NLI-based faithfulness scorer, would improve the reliability of the evaluation framework.

Stage II: context-length saturation requiring reversed ordering. At large k values with standard context order, the lost-in-the-middle effect substantially reduces generation quality, making reversed context ordering a necessary mitigation rather than an optional enhancement. Even with reversal, at k=20 one ground-truth question produced a null response due to context-length overflow in the Mistral-Nemo inference window, inflating the faithfulness average for that configuration by removing a likely low-quality entry from the denominator. The practical upper bound of k is therefore approximately 15; the recommended configuration (k=15, reversed order) avoids this instability while retaining the coverage benefits of the reversal strategy.

Chapter 7

Conclusions and Future Work

This thesis has described the design, implementation, and evaluation of a two-stage, fully open-source pipeline for the automated ingestion and natural-language querying of multilingual climate policy documentation. The work addressed a concrete and practically significant problem: the disorganised overabundance of policy instruments produced by national and international bodies in heterogeneous formats and languages, with no unified access mechanism and widely varying levels of computational accessibility. The proposed system demonstrates that a locally deployable stack — relying exclusively on open-weight models, open-source databases, and off-the-shelf Python libraries — can bridge the gap between this fragmented documentary landscape and structured, queryable domain knowledge, without any dependency on proprietary cloud infrastructure.

7.1 Summary of Contributions

The five contributions identified in Chapter 1 have been realised and empirically validated throughout this work.

Traffic-light licence validation. The three-layer cascade implemented in Phase 1 — trusted-domain lookup, multilingual heuristic pattern matching, and LLM-based disambiguation via Mistral-Nemo — successfully classified 83.1% of 249 candidate URLs as permissively licensed (VERDE) without manual intervention. The empirical finding that the deterministic pattern-matching layer alone produced errors in roughly 27% of all cases, and 18% false negatives, directly motivated the LLM-priority design adopted in the final system. To the best of our knowledge, no prior open-source RAG pipeline for policy documents addresses the legal-compliance dimension at ingestion time with comparable coverage across nine natural languages.

End-to-end multilingual ingestion at scale. Starting from 249 curated seed URLs, the pipeline ingested 819 documents spanning 22 languages, achieving a processing success rate exceeding 95%. The complete ingestion funnel — from 3,700 raw candidate files to 819 retained documents — reduced the collection by 77.9% through four successive quality gates, each targeting a distinct failure mode: binary duplicates, structurally deficient files, linguistic duplicates, and off-domain content. The incremental embedding mechanism, based on SHA-256 content hashing, ensures that periodic corpus updates require only the processing of newly added or modified documents, with a measured end-to-end cycle time of approximately 4.5 hours for a batch of 50 new entries on the PoliTO virtual machine.

Domain-specific taxonomy co-designed with domain experts. The metadata schema — covering hazard type, geographical level, key resilience criteria, document type, and thematic tags — was co-designed with the research group of Sofia Darbesio (PoliTO) to reflect the conceptual vocabulary of the Sendai Framework for Disaster Risk Reduction and the Paris Agreement. All 819 documents carry a complete structured record, with mandatory fields populated at 100% coverage through strict validation logic and deterministic fallback mechanisms. The taxonomy enables structured filtering during retrieval — a capability demonstrated empirically by the geo-boost experiments, where metadata-guided post-filtering raised faithfulness by up to 0.130 in a single configuration step.

Hierarchical parent–child chunking with context-weighted embeddings. The `AdvancedChunker` class implements a two-stage process: sliding-window semantic boundary detection identifies topical transitions without document-specific labels, and the resulting parent segments are subdivided into overlapping child chunks encoded with a structured metadata prefix that approximates the context-injection effect of late chunking [32]. The chunk identifier scheme — `{SHA256[:12]}_p{idx}_c{idx}` — encodes both document provenance and structural position, enabling exact reproducibility across ingestion runs. The asymmetric parent–child design — retrieving on child embeddings but passing parent texts to the generator — achieves high retrieval precision and high answer completeness simultaneously, without requiring the encoder to process very long sequences.

Empirical evaluation on a domain-specific gold set. The local re-implementation of the RAGAs framework, using `llama3.1:8b` as an independent judge model served via Ollama, provides a reproducible evaluation protocol for future work on multilingual policy RAG systems. The ablation study across eleven configurations traces the contribution of each retrieval-side intervention quantitatively: the three-query rewrite produced the largest single improvement, raising context recall by a factor of 7× over the bare baseline and faithfulness by 0.073 in a single step. Configuration 10 (k=15, three-query rewrite, reversed context order)

was identified as the recommended final configuration, combining the best answer correctness (0.507) with robust context recall (0.700) and competitive faithfulness (0.686).

7.2 Limitations

The principal limitations of the current system are discussed in detail in Section 6.7: corpus geographic imbalance, geographical classification noise, the structural necessity of reversed context ordering at large k values (with context-length overflow observed at k=20), context precision consistently zero under the binary RAGAs threshold, and LLM judge stochasticity.

One further limitation warrants mention here because it cuts across all pipeline stages rather than belonging to a single evaluation result.

Single-model dependency for multiple pipeline roles. Mistral-Nemo serves four distinct functions — licence classification, relevance pre-screening, metadata extraction, and answer generation — using the same set of model weights. While this minimises the total VRAM footprint and simplifies deployment, it introduces a single point of failure: if the model is unavailable or produces systematically degraded output for a particular language, all four pipeline stages are affected simultaneously. Assigning dedicated, task-optimised models to each role — for example, a lightweight classifier for licence analysis and a longer-context model for metadata extraction — would improve robustness at the cost of increased memory requirements.

7.3 Future Directions

The limitations described above, together with the broader research agenda opened by this work, suggest four concrete directions for future development.

7.3.1 Domain-Specific Fine-Tuning

The current system relies entirely on general-purpose instruction-tuned models (Mistral-Nemo, Llama 3.1:8b) without any domain-specific adaptation. Fine-tuning on the assembled corpus, using parameter-efficient methods such as LoRA [47] or QLoRA [23], could improve the model's fluency in Sendai Framework vocabulary, resilience criteria, and institutional naming conventions, reducing the frequency of out-of-domain paraphrases in generated answers.

The most promising application is the metadata extraction step (Phase 3): a model fine-tuned on a supervised set of (document-text, metadata-record) pairs

drawn from the current corpus would be better calibrated to the controlled vocabulary of hazard types, geographical levels, and key criteria than the general-purpose Mistral-Nemo prompt used today. The 819-document corpus assembled by Stage I constitutes a natural training substrate for this purpose once manually validated ground-truth metadata records are produced.

A practical obstacle is that fine-tuning requires a curated supervised dataset, which the current pipeline does not generate automatically. A staged approach is therefore recommended: first produce ground-truth metadata annotations for a random sample of 100–200 documents through expert review; then fine-tune a lightweight model (e.g., `Mistral-7B` or `Gemma-7B`) using QLoRA on this seed set; and finally expand the supervised set iteratively using the fine-tuned model's confident predictions as silver labels.

7.3.2 Automated Seed Discovery and Continuous Ingestion

The current pipeline operates from a static, manually curated seed list of 249 URLs. This design choice — motivated by the need to control corpus quality and avoid off-domain contamination — limits the system's ability to adapt to new policy developments without periodic human intervention.

A more autonomous approach would pair the existing ingestion pipeline with a targeted seed-discovery module that periodically queries institutional portal APIs, RSS feeds of policy-relevant organisations (UNDRR, IPCC, European Commission Joint Research Centre), and structured databases such as the EUR-Lex portal for newly published documents matching the domain taxonomy. The licence-analysis and quality-control gates already in place in Phase 1 and Phase 6 provide the necessary safeguards against ingesting off-domain or non-permissively licensed material.

The key technical challenge is distinguishing genuinely new documents from updated versions of existing ones. A combination of SHA-256 content hashing (already implemented for incremental embedding) and title-based fuzzy matching against the existing corpus would identify true duplicates and superseded versions, preventing index bloat without discarding updated policy content.

7.3.3 Periodic Re-Indexing and Corpus Versioning

As the corpus grows through continuous ingestion, the ChromaDB index and the BM25 sparse index must be kept consistent with the document collection. The current incremental embedding mechanism handles this efficiently for additions, but does not address the case where an existing document is revised (same URL, different content) or retracted.

A corpus versioning layer — tracking document versions through the `Content Hash` field alongside a `version_id` and an `active` flag — would allow the system to deprecate superseded chunks without deleting them from the store, preserving historical context for longitudinal policy analysis while ensuring that the default retrieval path returns only current content. This pattern is analogous to soft-deletion in relational databases and is directly supported by ChromaDB's `update` and `delete` APIs without requiring index rebuilding.

7.3.4 Agentic RAG and Multi-Hop Reasoning

The single-step retrieval architecture adopted in this work provides a favourable trade-off between latency, reproducibility, and answer quality for the predominant query type in the gold set — thematic identification and terminology extraction. However, the qualitative analysis in Section 6.6 reveals that a subset of policy queries inherently require cross-document synthesis: identifying which institutions across multiple countries explicitly mention cultural heritage in their DRM mandates requires assembling evidence from documents that share a topic but not a keyword, which is precisely the failure mode of single-step retrieval on sparse retrieval targets.

An agentic extension, following the ReAct [45] or IRCoT [46] frameworks, would allow the system to decompose such queries into a chain of targeted sub-queries, each issuing a new retrieval call and accumulating evidence before formulating a synthesised response. The existing query-decomposition module (which already generates three query variants for a single user question) provides a natural seed for this extension: the step from parallel multi-query decomposition to sequential query chains is architecturally straightforward.

The primary obstacle is latency: each additional retrieval–generation round-trip adds approximately 15–30 seconds of processing time on the PoliTO virtual machine, making fully agentic behaviour impractical for interactive use on the current hardware. A hybrid strategy — applying agentic retrieval only when the initial single-step response explicitly signals insufficient evidence — would retain low latency for the common case while enabling deeper synthesis for queries that genuinely require it.

Appendix A

LLM Prompt Templates

This appendix reproduces the complete text of every LLM prompt used in the pipeline, grouped by pipeline phase, together with the experimental system-prompt variants tested during the RAG ablation study (Section 6.5). Variable placeholders (Python f-string fields) are shown in `{curly_braces}`. Where a prompt contains injected vocabulary lists, the placeholder `{..._str}` represents a comma-separated string built from the corresponding controlled-vocabulary constant (e.g., `HAZARD_TYPES`, `KEY_CRITERIA`).

A.1 Phase 1 — Licence Classification

Called by `analizza_con_llama()` for every URL whose traffic-light score from the deterministic layer is ambiguous (`GIALLO`). The model receives a composite text snippet (page header, footer, and licence-keyword context window) and returns a strict JSON object.

```
You are a strict Legal Compliance AI.
Your goal is to classify the license of the provided text.
You are analyzing a document that may be in ANY LANGUAGE
(English, Italian, Spanish, French, German, Portuguese,
Turkish and Norwegian.).

DOCUMENT TEXT (first 3500 chars):
{testo_input_llm[:3500]}

TASK: Detect the license/copyright status regardless of language.

DECISION RULES:

GREEN LIGHT (PERMISSIVE) -> Output: VERDE
TRIGGER if you find ANY of these concepts:
* CC0 / CC BY / CC / Creative Commons / CC BY-SA / CC BY 4.0
* OGL / Open Government Licence
* Public domain / Dominio pubblico / Domaine public / Gemeinfrei
* Open access / Accesso aperto / Acces libre / Offener Zugang
* Free to use/reuse / Uso gratuito / Usage gratuit
```

```
* MIT, Apache, BSD licenses

CRITICAL - PERMISSION STATEMENTS: Look for phrases in ANY language:
* "Reproduction is authorized" / "Reproduction autorisee"
* "May be quoted" / "Citations allowed"
* "Use is permitted" / "L'utilisation est autorisee"
* "Provided the source is acknowledged"

RED (ROSSO): If you find ONLY restrictive terms:
* All rights reserved / Tous droits reserves
* Copyright protected
* Unauthorized use prohibited
* Permission required

YELLOW (GIALLO): ONLY if you CANNOT find ANY indication of
either GREEN or RED.

IMPORTANT: Analyze the CONTENT and MEANING, not just keywords.
Example: "Reproduction authorized provided the source is
acknowledged" = VERDE.

RESPONSE FORMAT (JSON only):
{"decisione":"VERDE/ROSSO/GIALLO",
 "licenza":"detected license name or type",
 "lingua":"detected language code (EN/IT/ES/FR/DE/PT/TR)"}
```

Listing A.1: Licence classification prompt.

A.2 Phase 2 — Pre-Download Relevance Gate

Called by `valida_rilevanza_documento_llm()` before downloading each candidate file. The model receives the link text, filename, and expected topic title; it returns a single `YES|NO` verdict with a one-word reason. Temperature is 0.05 and `num_predict` is limited to 20 tokens for minimal latency.

caption= Pre-download relevance gate prompt.

```
Quick relevance check. Answer ONLY "YES" or "NO" with
one-word reason.

Document: {link_text[:600]}
Filename: {filename}
Expected topic: {titolo_excel[:80]}

Is this a policy/regulation/framework/dataset document about
climate, disaster risk, resilience, or governance?
Every language is supported, not if just translation.
REJECT if about: agriculture, farming, fisheries, tourism,
trade, sports, education curriculum, country tourism guide,
humanitarian aid (except for climate related disaster).

Answer format: YES|NO - reason
Example: YES - climate policy
Example: NO - agriculture report
```

A.3 Phase 2 — Page-Type Classification

Called by `classifica_tipo_pagina_html()` only when the deterministic rules (policy-link count, learn-more button detection) cannot resolve the page type. The model returns one word: `CONTENT`, `DIRECTORY`, or `LANDING`.

```
Classify this webpage. Answer with ONE word:
CONTENT, DIRECTORY, or LANDING

URL: {url}
Title: {page_title[:80]}
Word count: {word_count}
PDF links: {len(pdf_links)}
Unique policy links: {len(unique_policy_links)}
Has "Learn More" button: {Yes/No}

CONTENT  = full document text/policy standalone without any link.
DIRECTORY = every HTML page with 1 or more links.
LANDING  = teaser page with JUST one "read more" button
           and no other links.

Answer ONE word:
```

Listing A.2: Page-type classification prompt.

A.4 Phase 2 — Document Quality Gate

Called by `verifica_qualita_con_llm()` for documents whose cleaned text length falls in the grey zone (200–1,000 characters). Returns a JSON with an `is_valid` boolean and a short reason string.

```
You are a Document Quality Auditor. Your job is to filter
out useless web scrapes.

ANALYZE THIS TEXT PREVIEW:
---------------------
{text_preview}
---------------------

TASK: Determine if this is a "Significant Document" worth
processing in ANY language.

CRITERIA FOR "NO" (INVALID):
- Just a list of links, index page, or search results.
- Login screen, cookie wall, or error page (404, 403).
- General navigation text without specific content.

CRITERIA FOR "YES" (VALID):
- Full text of a policy, report, or article.
- Substantial summary or abstract of a document.
- Database or dataset with meaningful content.

RESPONSE FORMAT (JSON ONLY):
{ "is_valid": true/false, "reason": "short explanation" }
```

Listing A.3: Document quality gate prompt.

A.5 Phase 3 — Metadata Extraction

Called by `estrai_metatag_con_llm_migliorato()` once per document. The model receives the first 10,000 characters of pre-cleaned document text and returns a twelve-field JSON object aligned with the Sendai taxonomy. Temperature is 0.08 to limit output variability while retaining some flexibility for free-text fields (description, target_and_relevance).

```
You are a metadata extraction AI for disaster risk and
resilience policy documents.

DOCUMENT CONTEXT:
- URL: {url}
- Detected Language: {detected_lang}
- Your output MUST be in English

CONTENT TO ANALYZE:
{testo_prioritario[:10000]}

EXTRACTION TASK - Fill ALL fields below:

1. html_title (CRITICAL):
   Extract the OFFICIAL TITLE printed on the document header
   or cover page. Do NOT use the filename or generic website
   titles.

2. issuing_authority:
   Extract the official organization name exactly as written.
   E.g. "United Nations", "European Commission".

3. document_type:
   Classify as ONE of: [{doc_types_str}]
   CLASSIFICATION GUIDE:
   - "Policy brief" = Short document with policy recommendations
     (< 30 pages)
   - "Report" = Comprehensive analysis, study, assessment
     (> 30 pages)
   - "Academic paper" = Peer-reviewed journal article with DOI
   - "Law" = Legal text, regulation, directive, statute
   - "Guidelines" = Step-by-step instructions, manual, handbook
   - "Framework" = Conceptual structure, strategic model
   - "Recommendation" = Official suggestions, proposals
   - "Glossary" = Definitions, terminology list
   - "Dataset" = Structured data compilation
   - "Other" = Anything not fitting above (add type in
     brackets, e.g., "Other (White Paper)")

4. hazard:
   Identify disaster types from: [{hazard_types_str}]
   - Multiple types -> "Multi-hazard: Flood, Earthquake, ..."
   - Generic "disaster" without specifics -> Multi-hazard
   - Purely administrative/generic -> "Unclear"

5. target_and_relevance:
   Summarize in 40-50 words: who is this for and what problem
   does it address?

6. key_criteria:
   AVAILABLE KEY CRITERIA (use ONLY exact strings):
```

```
   [{criteria_list_str}]
   KEY CRITERIA DEFINITIONS:
   {CRITERIA_DEFINITIONS}
   CLASSIFICATION RULES:
   1. Topic in document title or chapter title -> auto-assign.
   2. Minimum threshold: substantively addressed.
   3. Do NOT infer from vague or generic language.
   4. If no criteria apply -> "Unclear".

7. additional_tags:
   AVAILABLE ADDITIONAL TAGS (use ONLY exact strings):
   [{tags_list_str}]
   TAG DEFINITIONS:
   {TAGS_DEFINITIONS}
   Same classification rules as key_criteria.

8. description:
   100-130 words: main objective, key content, target audience,
   expected outcomes. Neutral tone.

9. language: two-letter code (EN, IT, ES, FR, DE, PT, TR...)

10. publication_date: YYYY-MM-DD format.
    Do NOT return today's date or yesterday's date.

11. geographical_level:
    ONE of: [{geo_levels_str}]
    For "National", add country: e.g. "National - Spain".
    If multi-country but not global: "Supranational/regional".

12. ai_notes:
    "High confidence" / "Low quality scan" /
    "Partial metadata" / "Language barrier"

CRITICAL RULES:
- Be INFERENTIAL for document_type, hazard, key_criteria.
- Be EXTRACTIVE for issuing_authority, dates, geographical_level.
- Never leave a field empty; use "Unclear" or "Not provided".

OUTPUT FORMAT (JSON only, no preamble):
{"issuing_authority":"...", "document_type":"...",
 "hazard":"...", "target_and_relevance":"...",
 "key_criteria":"...", "additional_tags":"...",
 "description":"...", "language":"...",
 "publication_date":"...", "geographical_level":"...",
 "ai_notes":"..."}
```

Listing A.4: Metadata extraction prompt (abbreviated; controlled-vocabulary lists replaced by placeholders).

A.6 Phase 4 — Query Decomposition

Called by `query_rag()` before any vector search. The model decomposes the user question into three search queries and extracts output-formatting instructions and geographic scope for the downstream geo-boost module. Temperature is 0.0 for deterministic output; `num_predict` is limited to 200 tokens.

```
You are a search query optimizer for a policy document
database on DRM, CCA, DRR.

Your task: process the user question and output a JSON
with FIVE fields.

DEFINITIONS:
- "query_1": Short clean search query (max 15 words).
  ONLY what to find. Remove formatting instructions
  (remove phrases like 'quote verbatim', 'provide reference',
  'list documents'). KEEP all domain terms, examples, and
  terminology from the question including terms in parentheses
  after 'e.g.' or 'such as'.
- "query_2": Second alternative search query (max 15 words),
  different angle or synonym. Same rules.
- "query_3": If the question contains explicit examples or
  terminology (e.g. in parentheses after "e.g." or "such as"),
  extract them as a keyword-focused query (max 10 words).
  Empty string if no explicit terms are given.
- "output_instructions": ONLY parts describing HOW to present
  the answer. Empty string if none.
- "geo_scope": Geographic scope. Use EXACTLY one of these
  values (copy verbatim):
    "European region"         -> European, EU, Europe, EEA
    "Supranational/regional"  -> regional (non-European),
                                 Pacific Islands
    "Global/Supranational"    -> global, UN, UNDRR, IPCC
    "National - [Country]"    -> specific country, e.g.
                                 "National - Norway"
    ""                        -> no geographic scope specified

EXAMPLE:
Question: "Which EU DRM frameworks define cultural heritage?
E.g. cultural heritage, historical buildings, monuments.
Quote verbatim with full reference [Authority, Title, Year]."
Output:
{
  "query_1": "EU DRM frameworks cultural heritage definition",
  "query_2": "European disaster risk management heritage
              monuments historical buildings",
  "query_3": "cultural heritage historical buildings monuments",
  "output_instructions": "Quote verbatim. Full reference
                          [Authority, Title, Year].",
  "geo_scope": "European region"
}

Now process this question:
{question}

Output ONLY the JSON, nothing else:
```

Listing A.5: Query decomposition prompt (final configuration).

A.7 Phase 4 — RAG Generation System Prompt

Injected as the system prompt in every call to `rag_answer()`. This is the final version (v1-revised) adopted after the prompt ablation study described in Section 6.5.5. The `ADDITIONAL OUTPUT REQUIREMENTS` block is appended only when `output_instructions` is non-empty.

```
You are an expert analyst of policy documents on Disaster
Risk Management (DRM), Climate Change Adaptation (CCA),
and Disaster Risk Reduction (DRR).

STRICT RULES:
1. Answer using ONLY information explicitly present in the
   provided context. NEVER add page numbers, section
   references, or institutional details not found verbatim
   in the sources.
2. If the context lacks sufficient information, state:
   "The available documents do not contain sufficient
   information to answer this question."
3. Structure your answer analytically: identify patterns
   and categories across documents; do not list sources
   one by one.
4. Quote terminology VERBATIM using quotation marks when
   citing specific terms from documents.
5. Cite all sources inline as [Source N]. At the end,
   list full references as: [Issuing Authority, Title, Year].
6. If the question specifies a geographic scope (e.g.
   European, Norwegian, Italian), prioritise sources at
   that level. Supranational/global sources are secondary
   and should be used only if no scope-specific source
   addresses the topic.
7. If sources contradict each other, acknowledge it
   explicitly.
8. Answer in the same language as the question.

REMEMBER: You are a retrieval system, not a knowledge
system. Only report what the documents say.

[If output_instructions is non-empty, the following block
is appended:]

ADDITIONAL OUTPUT REQUIREMENTS (from user):
{output_instructions}
```

Listing A.6: RAG generation system prompt (final version v1-revised).

A.8 Phase 4 — RAGAs Evaluation Prompts

The following prompts implement the local RAGAs-compatible evaluation framework described in Section 6.5. All calls use `llama3.1:8b` as judge at temperature 0.0.

```
Extract the main factual claims from the answer below.
Extract in the same language as the answer.
Output ONLY a numbered list, one short claim per line,
nothing else.

Answer:
{answer}

Factual claims (numbered list):
```

Listing A.7: Faithfulness — claim extraction prompt.

```
Read the context below and decide if the claim is
supported by it.

Context:
{context}

Claim: {claim}

Is this claim supported by the context above?
Reply with one word only: YES or NO.
```

Listing A.8: Faithfulness — claim verification prompt.

```
Generate 3 questions that the following answer is trying
to answer. Output ONLY the 3 questions, one per line,
no numbering, no prefixes.

Answer:
{answer}

Questions:
```

Listing A.9: Answer relevancy — question generation prompt.

```
Is the following context chunk useful to answer the question,
given the reference answer? Answer ONLY YES or NO.

Question: {question}
Reference answer: {ground_truth}
Context chunk: {ctx}

Answer (YES or NO):
```

Listing A.10: Context precision — chunk relevance prompt.

```
Extract all factual claims from the reference answer below.
Output ONLY a numbered list, one per line.

Reference answer:
{ground_truth}

Claims:
```

Listing A.11: Context recall — claim extraction prompt.

```
Is the following claim supported by the context?
Answer ONLY YES or NO.

Context:
{context}

Claim: {claim}

Answer (YES or NO):
```

Listing A.12: Context recall — claim extraction prompt.

```
Is the following claim present or implied in the answer?
Reply with a single word: YES or NO.

Answer:
{answer}

Claim: {claim}

Reply (YES or NO):
```

Listing A.13: Answer correctness — claim verification prompt.

A.9 Orchestrator — Intent Routing Prompt

Called by `agent_intent_router()` when the user provides free-text input instead of a numeric selection. The model maps the phrase to a phase digit (0–6) at temperature 0.0.

```
You are an orchestration assistant.
Your goal is to map the USER INPUT to the correct action ID.

{tools_description}

USER INPUT: "{user_input}"

INSTRUCTIONS:
- Return ONLY the digit corresponding to the action
  (1, 2, 3, 4, or 0).
- If the input is unclear, default to 0 (Exit).
- Do not write any text, just the number.

RESPONSE:
```

Listing A.14: Orchestrator intent-routing prompt.

A.10 Experimental System-Prompt Variants (Ablation Study)

Four alternative system prompts were evaluated in parallel with the retrieval ablation study (Section 6.5.5). Each variant was tested on the k=10 geo-boost baseline (configuration 5 in Table 6.2). Variants are reproduced in full to allow exact replication.

Variant v2 — Nine-Rule Prompt with Verbatim Directives

Faithfulness dropped to 0.400 (from 0.517 baseline) because the *use-all-sources* and *verbatim-quote* directives induced the model to cite page numbers and section references not present in the retrieved chunks.

```
You are an expert analyst of policy documents on Disaster
Risk Management (DRM), Climate Change Adaptation (CCA),
and Disaster Risk Reduction (DRR).

STRICT RULES:
1. Answer using ONLY the provided context. NEVER use your
   own training knowledge or external information.
2. EXHAUSTIVE USE OF SOURCES: Before stating "insufficient
   information", verify that you have used ALL provided
   sources.
3. VERBATIM QUOTES: When the question asks for terminology,
   excerpts, or specific language, always quote directly from
   the source using quotation marks.
4. GEOGRAPHIC SCOPE: If the question specifies a geographic
   scope, prioritise sources matching that scope. If no
   matching sources exist, explicitly state so.
5. STRUCTURE: Group findings by theme or document type.
   Do not list sources sequentially.
6. CITATIONS: Cite sources inline as [Source N]. At the end,
   provide full references: [Issuing Authority, Title, Year,
   Document_code if available].
7. CONFLICTS: If sources contradict each other, acknowledge
   it explicitly.
8. LANGUAGE: Answer in the same language as the question.
9. HONESTY: If a source mentions a topic only tangentially,
   say so explicitly rather than presenting it as direct
   evidence.

CRITICAL: You are a document retrieval analyst. A precise
"not found" is better than an unsupported claim.
```

Listing A.15: System prompt variant v2.

Variant v3 — Output Instructions Embedded in Rule 4

Faithfulness partially recovered to 0.436 but remained below the v1 baseline. The attempt was to make output-formatting instructions mandatory by embedding them inside the numbered rules.

```
[Rules 1-3 identical to v1]

4. Quote terminology VERBATIM using quotation marks when
   citing specific terms from documents.
   MANDATORY FORMAT: {output_instructions}

[Rules 5-8 identical to v1]
```

Listing A.16: System prompt variant v3 (output instructions in rule 4).

The only structural change relative to v1 is that `output_instructions` is injected as a `MANDATORY FORMAT` clause inside rule 4 instead of being appended as a separate `ADDITIONAL OUTPUT REQUIREMENTS` block.

Variant v5 — Geographic Scope Rule Removed

Faithfulness fell to 0.392, demonstrating that retrieval-level (geo-boost) and generation-level geographic constraints serve different purposes and are not interchangeable. Rule 6 of v1 was removed entirely on the hypothesis that the geo-boost module made it redundant at the prompt level.

```
You are an expert analyst of policy documents on Disaster
Risk Management (DRM), Climate Change Adaptation (CCA),
and Disaster Risk Reduction (DRR).

STRICT RULES:
1. Answer using ONLY information explicitly present in the
   provided context. NEVER add page numbers, section
   references, or institutional details not found verbatim.
2. If the context lacks sufficient information, state:
   "The available documents do not contain sufficient
   information to answer this question."
3. Structure your answer analytically: identify patterns
   and categories across documents.
4. Quote terminology VERBATIM using quotation marks when
   citing specific terms from documents.
5. Cite all sources inline as [Source N]. At the end,
   list full references as: [Issuing Authority, Title, Year].
6. If sources contradict each other, acknowledge it.
7. Answer in the same language as the question.

REMEMBER: You are a retrieval system, not a knowledge
system. Only report what the documents say.
```

Listing A.17: System prompt variant v5 (no geographic-scope rule).

References

[1] United Nations Office for Disaster Risk Reduction. *Sendai Framework for Disaster Risk Reduction 2015-2030*. Tech. rep. United Nations, 2015. URL: https://www.undrr.org/publication/sendai-framework-disaster-risk-reduction-2015-2030 (cit. on pp. 1, 50).

[2] UNFCCC. *Paris Agreement*. 2015. URL: https://unfccc.int/process-and-meetings/the-paris-agreement (cit. on p. 1).

[3] United Nations. *Sustainable Development Goals*. 2015. URL: https://sdgs.un.org/ (cit. on p. 1).

[4] European Commission. *European Green Deal*. Tech. rep. European Commission, 2021. URL: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/european-green-deal_en (cit. on p. 1).

[5] IPCC. *Climate Change 2023: Synthesis Report*. Tech. rep. Intergovernmental Panel on Climate Change, 2023. URL: https://www.ipcc.ch/report/ar6/syr/ (cit. on p. 1).

[6] United Nations Office for Disaster Risk Reduction. *Global Assessment Report on Disaster Risk Reduction 2025*. Tech. rep. United Nations, 2025 (cit. on p. 1).

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is All You Need». In: *Advances in Neural Information Processing Systems 30 (NeurIPS)*. 2017, pp. 5998–6008 (cit. on pp. 2, 7, 9–11).

[8] Tom B. Brown et al. «Language Models are Few-Shot Learners». In: *arXiv preprint arXiv:2005.14165* (2020) (cit. on pp. 2, 10, 11).

[9] Ziwei Ji et al. «Survey of Hallucination in Natural Language Generation». In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38 (cit. on pp. 3, 13).

[10] Patrick Lewis et al. «Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks». In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. 2020, pp. 9459–9474 (cit. on pp. 3, 7, 13, 20).

[11] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. «From Local to Global: A Graph RAG Approach to Query-Focused Summarization». In: *arXiv preprint arXiv:2404.16130* (2024) (cit. on pp. 7, 26).

[12] Jerry Liu. *LlamaIndex: A Data Framework for LLM Applications*. 2022. URL: `https://www.llamaindex.ai/` (cit. on pp. 7, 15).

[13] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 8).

[14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation». In: *Proceedings of EMNLP*. 2014, pp. 1724–1734 (cit. on p. 8).

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. «Neural Machine Translation by Jointly Learning to Align and Translate». In: *Proceedings of ICLR*. 2015 (cit. on p. 9).

[16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. «Layer Normalization». In: *arXiv preprint arXiv:1607.06450*. 2016 (cit. on p. 9).

[17] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. «What Does BERT Look At? An Analysis of BERT's Attention». In: *Proceedings of the Workshop on Analyzing and Interpreting Neural Networks for NLP (BlackboxNLP)*. 2019, pp. 276–286 (cit. on p. 9).

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *Proceedings of NAACL-HLT*. 2019, pp. 4171–4186 (cit. on pp. 10, 16).

[19] Hugo Touvron et al. «LLaMA: Open and Efficient Foundation Language Models». In: *arXiv preprint arXiv:2302.13971* (2023) (cit. on pp. 11, 12).

[20] Albert Q. Jiang et al. «Mistral 7B». In: *arXiv preprint arXiv:2310.06825* (2023) (cit. on pp. 11, 12).

[21] Hugo Touvron et al. «Llama 2: Open Foundation and Fine-Tuned Chat Models». In: *arXiv preprint arXiv:2307.09288* (2023) (cit. on p. 12).

[22] Google DeepMind. *Gemma: Open Models Based on Gemini Research and Technology*. Tech. rep. Google, 2024. URL: `https://ai.google.dev/gemma` (cit. on p. 12).

[23] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. «QLoRA: Efficient Finetuning of Quantized LLMs». In: *arXiv preprint arXiv:2305.14314* (2023) (cit. on pp. 12, 22, 97).

[24] Ollama Team. *Ollama: Get up and running with large language models locally.* 2023. URL: `https://ollama.ai/` (cit. on pp. 13, 32).

[25] Georgi Gerganov. *llama.cpp: Inference of LLaMA model in pure C/C++.* 2023. URL: `https://github.com/ggerganov/llama.cpp` (cit. on p. 13).

[26] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. «Dense Passage Retrieval for Open-Domain Question Answering». In: *Proceedings of EMNLP*. 2020, pp. 6769–6781 (cit. on pp. 13, 17, 19, 66).

[27] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension». In: *Proceedings of ACL*. 2020, pp. 7871–7880 (cit. on p. 13).

[28] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. «Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection». In: *arXiv preprint arXiv:2310.11511* (2024) (cit. on pp. 14, 21).

[29] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. «Corrective Retrieval Augmented Generation». In: *arXiv preprint arXiv:2401.15884* (2024) (cit. on pp. 14, 21).

[30] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. «Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity». In: *arXiv preprint arXiv:2403.14403* (2024) (cit. on p. 14).

[31] LangChain. *LangChain Documentation: Text Splitters.* 2023. URL: `https://python.langchain.com/docs/modules/data_connection/document_transformers/` (cit. on pp. 15, 77).

[32] Michael Günther, Isabelle Mohr, Seyed Mohamad Moosavidezfooli, Bo Wang, Georgios Behera, Hamidreza Liu, Konstantin Mastrapas, and Nan Wang. «Late Chunking: Contextual Chunk Embeddings Using Long-Context Embedding Models». In: *arXiv preprint arXiv:2409.04701* (2024) (cit. on pp. 15, 16, 56, 60, 77, 96).

[33] Nils Reimers and Iryna Gurevych. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks». In: *Proceedings of EMNLP-IJCNLP*. 2019, pp. 3982–3992 (cit. on pp. 16, 33).

[34] Nils Reimers and Iryna Gurevych. «Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation». In: *Proceedings of EMNLP*. 2020, pp. 4512–4525 (cit. on pp. 17, 18, 34, 61).

[35] Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. «Making a MIRACL: Multilingual Information Retrieval Across a Continuum of Languages». In: *Proceedings of EMNLP*. 2023 (cit. on pp. 17, 18, 26, 27).

[36] Luiz Bonifacio, Vitor Jeronymo, Hugo Queiroz Campiotti, Roberto Lotufo, and Rodrigo Nogueira. «mMARCO: A Multilingual Version of MS MARCO Passage Ranking Dataset». In: *arXiv preprint arXiv:2108.13897* (2022) (cit. on pp. 17, 27).

[37] Michael Günther, Isabelle Mohr, Seyed Mohamad Moosavidezfooli, Bo Wang, Georgios Behera, Hamidreza Liu, Konstantin Mastrapas, and Nan Wang. «Jina Embeddings v3: Multilingual Embeddings With Task LoRA». In: *arXiv preprint arXiv:2409.10173* (2024) (cit. on pp. 17, 33, 60, 61).

[38] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. «Improving Text Embeddings with Large Language Models». In: *arXiv preprint arXiv:2401.00368* (2024) (cit. on pp. 17, 18).

[39] Yury A. Malkov and Dmitry A. Yashunin. «Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 42. 4. 2020, pp. 824–836 (cit. on pp. 18, 62).

[40] Jeff Johnson, Matthijs Douze, and Hervé Jégou. «Billion-scale similarity search with GPUs». In: *IEEE Transactions on Big Data*. Vol. 7. 3. 2019, pp. 535–547 (cit. on pp. 19, 62).

[41] Chroma. *ChromaDB: The AI-native open-source embedding database*. 2023. URL: `https://www.trychroma.com/` (cit. on pp. 19, 33, 62).

[42] Stephen Robertson and Hugo Zaragoza. «The Probabilistic Relevance Framework: BM25 and Beyond». In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389 (cit. on pp. 19, 66, 67).

[43] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. «Reciprocal Rank Fusion outperforms Condorcet and individual Rank Learning Methods». In: *Proceedings of SIGIR*. 2009, pp. 758–759 (cit. on pp. 19, 68).

[44] Omar Khattab and Matei Zaharia. «ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT». In: *Proceedings of SIGIR*. 2020, pp. 39–48 (cit. on p. 20).

[45] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. «ReAct: Synergizing Reasoning and Acting in Language Models». In: *Proceedings of ICLR*. 2023 (cit. on pp. 21, 99).

[46] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. «Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions». In: *arXiv preprint arXiv:2212.10509* (2023) (cit. on pp. 21, 99).

[47] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. «LoRA: Low-Rank Adaptation of Large Language Models». In: *arXiv preprint arXiv:2106.09685* (2022) (cit. on pp. 22, 97).

[48] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. «Finetuned Language Models Are Zero-Shot Learners». In: *arXiv preprint arXiv:2109.01652* (2022) (cit. on p. 23).

[49] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models». In: *Advances in Neural Information Processing Systems 35 (NeurIPS)*. 2022 (cit. on p. 23).

[50] Harrison Chase. *LangChain: Building applications with LLMs through composability*. 2023. URL: `https://github.com/langchain-ai/langchain` (cit. on p. 23).

[51] Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. «Instruction Induction: From Few Examples to Natural Language Task Descriptions». In: *arXiv preprint arXiv:2205.10782* (2022) (cit. on p. 23).

[52] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. «FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation». In: *arXiv preprint arXiv:2305.14251* (2023) (cit. on p. 23).

[53] Dominik Stammbach, Maria Antoniak, Cynthia Liem Fok, Marius Froehlich, Julian Risch, Elliott Ash, and Adam Perer. «CHATREPORT: Democratizing Sustainability Disclosure Analysis through LLM-based Tools». In: *arXiv preprint arXiv:2307.15770* (2023) (cit. on pp. 23, 27).

[54] Ali Vaghefi, Victor Maus, and Aidin Niamir. «Information extraction from corporate sustainability reports using large language models». In: *arXiv preprint arXiv:2310.08899* (2023) (cit. on pp. 23, 27).

[55] Xiaoyu Tan, Zhenyu Zhu, Chenxin Wu, Jiayi Chen, and Chengjie Zhang. «LLM-assisted Information Extraction from Unstructured Documents». In: *arXiv preprint arXiv:2402.08686* (2024) (cit. on p. 23).

[56] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. «BLEU: a Method for Automatic Evaluation of Machine Translation». In: *Proceedings of ACL*. 2002, pp. 311–318 (cit. on p. 24).

[57] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text Summarization Branches Out: Proceedings of the ACL Workshop*. 2004, pp. 74–81 (cit. on p. 24).

[58] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. «RAGAs: Automated Evaluation of Retrieval Augmented Generation». In: *arXiv preprint arXiv:2309.15217* (2024) (cit. on pp. 24, 71).

[59] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. «HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering». In: *Proceedings of EMNLP*. 2018, pp. 2369–2380 (cit. on p. 25).

[60] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. «BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models». In: *Proceedings of NeurIPS Datasets and Benchmarks Track*. 2021 (cit. on p. 26).

[61] V. A. Traag, L. Waltman, and N. J. van Eck. «From Louvain to Leiden: guaranteeing well-connected communities». In: *Scientific Reports* 9.1 (2019), p. 5233 (cit. on p. 26).

[62] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. «RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval». In: *arXiv preprint arXiv:2401.18059* (2024) (cit. on p. 26).

[63] IBM Research. *Docling: Parse documents and export them to the desired format*. 2024. URL: `https://github.com/DS4SD/docling` (cit. on p. 26).

[64] Kamila R. Usmanova, Graziano Ferraro, and Evangelos Pournaras. «Structuring sustainability reports using large language models and knowledge graphs». In: *arXiv preprint arXiv:2403.17410* (2024) (cit. on p. 27).

[65] EU Publications Office. *Knowledge Graph*. 2023. URL: `https://op.europa.eu/en/web/eu-vocabularies` (cit. on p. 27).

[66] Streamlit Inc. *Streamlit: The fastest way to build and share data apps*. 2023. URL: `https://streamlit.io/` (cit. on p. 37).

[67] UNFCCC. *Paris Agreement*. 2015. URL: `https://unfccc.int/process-and-meetings/the-paris-agreement` (cit. on p. 50).

[68] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. «Lost in the Middle: How Language Models Use Long Contexts». In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173 (cit. on p. 89).