Process-Aware Retrieval-Augmented Generation for Auditable Compliance Reasoning

PIC2 - Master in Computer Science and Engineering
Instituto Superior Técnico, Universidade de Lisboa

José Maria Gil de Borja Sacadura e Menezes — 113237*
jose.sacadura.menezes@tecnico.ulisboa.pt

Advisor(s): Chrysoula Zerva, Alessandro Gianola

Abstract Automating compliance checks in public administration remains a significant challenge due to the fragmented nature of administrative data and the inherent complexity of cross-referenced legal regulations. Standard Large Language Model (LLM) approaches often lack the structural awareness required to navigate these dependencies, leading to logical errors and hallucinations. This research proposes a **Multi-Agent Graph-RAG Framework** designed to bridge the gap between unstructured legal narratives and structured process data.

At the core of the framework is a **"Forest of Graphs"** architecture, which organizes information into multi-layered knowledge structures—separating high-level regulatory concepts from granular administrative event logs. By utilizing a team of specialized AI agents, the system moves beyond simple keyword retrieval toward a **three-layer reasoning process** that aligns specific process steps with legal requirements.

The framework's efficacy is evaluated through two distinct lenses: **grounding** and **procedural transparency**. Using the **ECHR dataset** and the **LegalBench-RAG** benchmark, we measure the system's ability to produce binary "conformance" decisions that are semantically aligned with human-annotated legal rationales. Furthermore, utilizing the **BPI Challenge 2018 (BPIC18)** dataset, we demonstrate how the multi-agent approach transforms opaque outputs into verifiable audit trails. This research aims to provide a robust, interpretable, and accurate solution for high-stakes administrative compliance, ensuring that AI-driven decisions are both grounded in law and procedurally sound.

Keywords: Graph-RAG, Multi-Agent Systems, Public Administration Compliance, Forest of Graphs, Process Alignment, Grounding.

*I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa (https://nape.tecnico.ulisboa.pt/en/apoio-ao-estudante/documentos-importantes/regulamentos-da-universidade-de-lisboa/).

Contents

1 Introduction

High-risk, documentation-heavy domains—including public administration, healthcare, legal services, finance and insurance, and safety-critical industrial operations—are governed by complex and frequently evolving bodies of rules, standards, and internal procedures. In these settings, compliance is rarely a matter of checking a single clause in a single document: decisions must be justified through cross-referenced documentation, consistent interpretation of exceptions and definitions, and evidence that the prescribed process was followed. In practice, this verification burden is amplified by heterogeneous information environments that combine structured records (databases, forms, event logs) with unstructured narratives (policies, guidelines, contracts, clinical notes, regulatory texts). As volumes scale, manual review becomes a bottleneck, increasing both cost and the risk of missed obligations or incorrect decisions. Crucially, accountability depends not only on the final decision (e.g., compliant / non-compliant, eligible / ineligible, indicated / contraindicated), but also on producing an auditable rationale: what evidence was used, which norms were applied, and how procedural constraints shaped the outcome. This requirement is especially visible in public administration, where transparency and traceability are central, but it is equally critical in other domains where errors can lead to legal exposure, patient harm, financial loss, or loss of public trust.

Large Language Models (LLMs) offer strong capabilities for parsing and synthesising natural language, but naive deployments are unsuitable for high-stakes compliance. LLMs may generate plausible but incorrect statements ("hallucinations") and can fail to preserve fine-grained constraints when reasoning across multiple documents and long contexts. Simply increasing the context window is not a reliable solution: evidence can be overlooked in long inputs, and long-context inference remains costly and brittle for precision-critical tasks. In regulated decision-making, these failure modes are unacceptable because they can produce confident outputs without sufficient grounding.

Retrieval-Augmented Generation (RAG) partially addresses this problem by grounding generation in retrieved evidence. Conceptually, RAG acts as an "open-book" approach: the system retrieves relevant content from an external corpus and then conditions the LLM response on that evidence. However, standard RAG pipelines are often structurally blind. Compliance sources in domains like public administration, law, and medicine contain dense dependency structures—cross-references, amendments, exceptions, hierarchical definitions, and dependencies between documents and procedural steps. Similarity-based retrieval may return a governing rule while missing its exceptions, or retrieve an obligation while omitting prerequisites, thresholds, or temporal conditions. As a result, a model may generate outputs that are textually supported yet normatively incomplete (legally/clinically/administratively unsound) or procedurally invalid.

A further challenge is that compliance is frequently process-dependent: it is not enough that certain documents exist; the system must verify that actions occurred in the correct order, with required approvals, within deadlines, and with mandatory checks completed. Many real-world failures arise precisely from skipped or misordered steps—for example, administrative approvals not recorded before a decision, contractual steps performed outside authorized workflow, or clinical actions taken without required documentation. Therefore, scalable compliance automation requires both (i) grounded reasoning over heterogeneous documentation and (ii) explicit validation against process constraints.

This thesis proposes a Process-Aware Multi-Agent Graph-RAG Framework designed for high-stakes compliance settings, with public administration as a key motivating domain and a broader target scope that includes legal and clinical contexts. The approach organizes information into a multi-layer "Forest of Graphs" that separates normative sources (e.g., regulations, guidelines, contractual clauses) from operational traces (e.g., events, forms, logs), while preserving explicit links between them. A team of specialized agents decomposes compliance questions into retrieval, interpretation, and verification subtasks, producing outputs with explicit grounding and auditability. Finally, a process alignment layer checks conclusions against extracted or provided procedural constraints, aiming to ensure outputs are not only supported by text, but also procedurally sound.

1.1 Research Questions

This thesis aims to bridge the gap between generative language modeling and structured, rule- and process-governed decision-making in high-risk, documentation-heavy domains—with public administration as a key motivating setting, alongside legal and clinical compliance workflows.Concretely, we develop a system capable of reasoning over hierarchical normative sources (e.g., statutes, regulations, policies, clinical guidelines, contractual frameworks), handling exceptions and cross-references, and validating conclusions against dynamic processes (e.g., approvals, deadlines, required checks, and event sequences). The research is guided by the following four questions:

- **RQ1 (Data Heterogeneity):** How can Large Language Models be engineered to reason simultaneously over heterogeneous data sources—combining unstructured regulatory documents (PDFs) with structured administrative, legal or clinical records (SQL/Forms)—without losing semantic precision?
- **RQ2 (Retrieval Optimization):** How can structured, hierarchy- and graph-aware retrieval improve evidence-grounded reasoning in documentation-heavy domains? To what extent can the integration of *Hierarchical Semantic Trees* and *Graph-based Navigation* optimize RAG performance by mitigating the problem of waving the wrong interpretation ("semantic drift") and the problem of losing connections between the information gathered (multi-hop fragmentation) compared to standard flat-vector approaches (like vector search), and what are the associated trade-offs in computational latency
- **RQ3 (Agentic Performance):** Could we employ a multi-agent architecture that is decomposing tasks and routing to models with specialized roles (e.g., Librarian, Analyst, Auditor) to improve the reliability and depth of compliance reports compared to monolithic, single-agent RAG systems?
- **RQ4 (Process Adherence):** How can agentic systems be aligned with explicit procedural constraints? Can verification-and-correction loops be integrated into agentic workflows to check procedural compliance against explicit process constraints, I.e. to detect, localize, and address non-compliance with required steps, ordering, approvals, and deadlines—improving traceability and error containment.

1.2 Work Objectives

In public administration it's normal to have different types of data with very specific sets of rules. In the case of a legal decision, has multiple related documents like the accusation, the law and the final decision. Any attempt at having a system to make more efficient the processing of the information would need to be able to process and then explain the solution it came up based on the the data it has. The primary objective of this thesis is to develop a **Process-Aware Multi-Agent Graph-RAG Framework** that has the capability of processing and reasoning over multiple different document and get the correct interpretation it needs. To achieve this, the following specific objectives have been defined:

1. **Unify Heterogeneous Data:** Develop a Unified Knowledge Graph that links unstructured legal texts with structured administrative forms, enabling the system to reason across different data formats (Addressing RQ1).
2. **Facilitate Hierarchical Reasoning:** Design a *"Semantic Tree"* indexing strategy that abstracts document chapters into recursive summaries, allowing the system to maintain global context while drilling down into granular legal details (Addressing RQ2).
3. **Enhance Relational Discovery via GNN:** Implement a *Graph Neural Network* (GNN) [1] layer to compute structural embeddings for the Knowledge Graph, enabling the discovery of non-obvious legal dependencies and multi-hop relationships (Addressing RQ2).

4. **Optimize Performance via Multi-Agent Coordination:** Implement a multi-agent orchestration layer consisting of specialized agents (Librarian, Graph Analyst, and Report Auditor) to manage retrieval and reasoning process of each agent. (Addressing RQ3).

5. **Integrate Process Feedback:** Design a *"Process Alignment"* module that learns procedural constraints in order to provide feedback and correct the agents, ensuring the AI's reasoning adheres to official administrative workflows and regulatory constraints (Addressing RQ4).

2 Background

This section establishes the theoretical and technical foundations of the research. It begins by examining the Transformer architecture that underpins modern Large Language Models (LLMs). It then explores the evolution of LLMs from simple text generators to reasoning agents capable of tool use. Finally, it provides a detailed analysis of Retrieval-Augmented Generation (RAG), focusing specifically on the challenges and methods for integrating heterogeneous data sources (structured and unstructured) in compliance contexts.

2.1 The Transformer Architecture

The foundation of modern Natural Language Processing (NLP) is the *Transformer* architecture, introduced by Vaswani et al. [2]. Prior to this, sequence processing relied heavily on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks [3]. These earlier architectures processed data sequentially, which precluded parallelization and led to the "vanishing gradient" problem, making it difficult to retain information over long sequences.

2.1.1 Self-Attention Mechanism

The core innovation of the Transformer is the *Self-Attention* mechanism. This allows the model to weigh the importance of different words in a sequence relative to one another, regardless of their positional distance. Formally, the attention function maps a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

The attention calculation is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

Where:

- Q represents the **Queries** (what the token is looking for).
- K represents the **Keys** (what the token identifies as).
- V represents the **Values** (the actual informational content).
- $\sqrt{d_k}$ is a scaling factor to prevent vanishing gradients in the softmax function.

In the context of compliance checking, this mechanism enables the model to resolve long-range dependencies. For instance, it can understand that a "deadline" mentioned in a policy document is semantically bound to a specific date mentioned several paragraphs earlier—a capability that was severely limited in RNN-based architectures.

2.1.2 Context Window and Quadratic Complexity

While Transformers enable deep semantic understanding, they introduce a computational bottleneck known as *Quadratic Complexity* ($O(n^2)$). As the length of the input sequence (n) increases, the memory and computational resources required to compute the attention matrix (QK^T) grow quadratically.

This limitation is the primary technical justification for the adoption of **Retrieval-Augmented Generation (RAG)** [4]. Since it is computationally prohibitive to feed an entire corpus of public administration regulations into a single Transformer context window, RAG serves as an external memory mechanism, selecting only the most relevant "chunks" for the attention mechanism to process.

2.2 Foundations of Large Language Models

Modern Large Language Models (LLMs) are based on the Transformer architecture, specifically utilizing the decoder-only variant to perform autoregressive text generation. This section details the transition from raw text to numerical representations and the training regimes that enable high-level reasoning.

2.2.1 Tokenization and Vocabulary Mapping

Before an LLM can process natural language, text must be converted into discrete numerical units called tokens. Modern models primarily utilize *Byte-Pair Encoding* (BPE) [5], a sub-word tokenization method that balances vocabulary size and the ability to represent rare words.

Once tokenized, each token is mapped to a high-dimensional vector known as an *embedding* [6]. Formally, a sequence of n tokens is represented as a matrix $X \in \mathbb{R}^{n \times d}$, where d is the model's hidden dimension. To preserve the order of the sequence, positional encodings are added to these embeddings, as the attention mechanism is inherently permutation-invariant.

2.2.2 Decoder-Only Transformer Architecture

While the original Transformer proposed by Vaswani et al. [2] featured an encoder-decoder structure, most state-of-the-art LLMs (such as the GPT family) utilize a *decoder-only* architecture [7] .

The core of the decoder is the Masked Multi-Head Self-Attention. Unlike the standard attention, the masked version ensures that the prediction for position i can only depend on known outputs at positions less than i, maintaining the autoregressive property. The probability of a sequence is decomposed as:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_1, \ldots, x_{i-1}) \tag{2}$$

At each step, the model produces a distribution over the vocabulary via a softmax layer. During inference, tokens are selected using decoding strategies such as Top-p sampling or Greedy search to generate the final response.

2.2.3 Training Paradigms: From Pre-training to Alignment

The development of a functional Large Language Model (LLM) involves a progression through three distinct stages, shifting from raw knowledge acquisition to behavioral refinement.

1. **Pre-training (Knowledge Acquisition):** The model is initialized with random weights and trained on a massive corpus (e.g., Common Crawl, Wikipedia) to acquire "world knowledge" and linguistic syntax [8]. This process is governed by the objective of *Causal Language Modeling (CLM)*, where the model learns to predict the probability of the next token x_t based solely on the sequence of previous tokens. Mathematically, this maximizes the likelihood:

$$P(x) = \prod_{i=1}^{T} P(x_i | x_1, \ldots, x_{i-1}) \tag{3}$$

At this stage, the model acts as a "stochastic parrot," capable of completing text patterns but lacking the ability to discern user intent. For example, if prompted with "How do I file a tax return?", a pre-trained model might continue with "...is a question many citizens ask every April," rather than providing instructions.

2. **Supervised Fine-Tuning (SFT):** To transform a text-completer into a helpful assistant, the model undergoes *Instruction Tuning*. This involves training on a curated dataset where humans have written both the prompt and the "gold standard" response (see [9]). The model learns to map specific input patterns to desired output behaviors via backpropagation, adjusting its weights to minimize the difference between its prediction and the human's example. For instance, an SFT dataset might include the pair: *(Prompt: "Summarize Article 5 of the ECHR", Response: "Article 5 protects the right to liberty and security...")*. By seeing thousands of these examples, the model learns the "persona" of an assistant that follows commands rather than just predicting the next word in a sentence.

3. **Alignment (RLHF):** Even after SFT, models may exhibit bias or "verbose" hallucinations. *Reinforcement Learning from Human Feedback (RLHF)* refines this by using a *Reward Model* [10]. In this stage, the LLM generates multiple answers for one prompt, and humans rank them (e.g., Output A is better than Output B). A Reward Model is trained to mimic these human preferences. The LLM then uses an optimization algorithm (like PPO) to play a "game" where it tries to generate text that gets the highest score from the Reward Model. An example of this in a legal context would be penalizing the model if it generates a "helpful" sounding answer that actually cites a non-existent law, teaching it to prioritize factual accuracy over sounding confident.

2.3 LLMs: From Generation to Reasoning and Agency

Modern Large Language Models (LLMs) such as GPT-5 [11], Llama 3 [12], DeepSeek-V3.2 [13], Claude 4.1 [14] and more have evolved beyond simple probabilistic text generation. In the context of this thesis, they are conceptualized not as word predictors, but as reasoning engines capable of autonomous agency. This shift is critical for automated compliance, where the system must apply static logic to dynamic, ever-changing regulations.

2.3.1 The LLM as a Thought Process Unit

A fundamental limitation of traditional Deep Learning models is their "black box" nature—inputs are processed into outputs via massive matrix multiplications with no visibility into the intermediate logic. To address this in high-stakes domains like public administration, this thesis treats the LLM as a *Thought Process Unit*.

In this paradigm, the LLM functions analogously to a CPU (Central Processing Unit) rather than a Hard Drive. It does not need to store the specific municipal regulations (*knowledge*) within its weights; rather, it processes external data (retrieved via RAG) using learned reasoning patterns.

2.3.2 Chain-of-Thought (CoT) and Reasoning Steps

The primary mechanism for enabling sophisticated logical processing within LLMs is *Chain-of-Thought (CoT)* prompting, as introduced by Wei et al. [15]. Standard prompting strategies require the model to map a query directly to an output, forcing all logical computations to occur implicitly within the model's hidden layers in a single forward pass. This "zero-shot" approach often collapses when applied to complex compliance tasks where multiple regulatory conditions must be evaluated sequentially.

CoT overcomes this by encouraging the model to generate intermediate reasoning steps—often initiated by the heuristic "Let's think step by step." By externalizing the reasoning process into a sequence of discrete tokens, the model can condition its final conclusion on its own prior logical derivations, significantly reducing the frequency of logical leaps and hallucinations.

- **Compliance Example:** Instead of an opaque "Non-Compliant" verdict, a CoT-enabled agent generates an explicit logical chain: *"Step 1: Article 4 requires a certified signature. Step 2: The provided document contains only a digital timestamp. Step 3: A timestamp does not meet the certification criteria of Article 4. Conclusion: Non-Compliant."*

More recent advancements have evolved CoT from a simple prompting technique into a core architectural paradigm known as *Inference-time Scaling* or *Test-time Compute* [16]. Models such as DeepSeek-R1 and OpenAI's o1 series are specifically trained using large-scale reinforcement learning to internalize these "thoughts," allowing them to spend more time processing a query before responding. These modern iterations introduce critical capabilities such as *Self-Correction*—where the model recognizes a contradiction in its own reasoning path and backtracks to find a correct solution—and *Verifiable Reasoning Chains*, which align the model's internal "hidden" thoughts with the structured requirements of an audit trail. For administrative compliance, this represents a shift from simple text generation to a deliberative process that mimics the meticulous review of a human legal auditor.

2.3.3 LLM Agents and The ReAct Paradigm

While CoT improves reasoning, it is limited to the information present in the prompt. To handle the complexity of public administration tasks—which require interacting with external databases, calculating sums, and verifying dates—LLMs are deployed as *Agents*. LLM Agents are LLMs that are prompted to achieve a better result on the actions promoted by the user by specifying how to reason and process.

The ReAct Loop This thesis adopts the *ReAct* (Reasoning + Acting) paradigm proposed by Yao et al. [17]. Unlike a standard chatbot that simply responds, a ReAct agent operates in a loop:

1. **Thought:** The agent analyzes the user's request (e.g., "Check if this invoice matches the contract").
2. **Action:** The agent decides to call an external tool (e.g., `search_database(invoice_id)`) this tool can be independent of the LLM.
3. **Observation:** The agent receives the output from the tool (e.g., "{Value: 5000 EUR}").
4. **Reasoning:** The agent processes this observation against the context (e.g., "The contract limit is 4000 EUR, so this is a violation").

This iterative process allows the system to ground its answers in reality, overcoming the LLM's inability to perform precise arithmetic or access private data. Instead of generating the output the agent will understand when it needs to call an external tool to execute the part of the prompt it needs. Using this paradigm allows to have a more standard way of prompting the agent to do the actions and reasoning that the system needs.

Multi-Agent Systems (MAS)
For complex workflows, a single agent often becomes overwhelmed by conflicting instructions (e.g., "be concise" vs. "be thorough"). Recent research by Li et al. [18] suggests that scaling the number of specialized agents improves performance on complex reasoning tasks. This thesis proposes a Multi-Agent architecture where specific roles are distributed:

- **Planner Agent:** Decomposes a high-level audit request into executable sub-tasks (e.g., "First retrieve the policy, then check the database").
- **Tool-Use Agent:** A specialized worker that interfaces with structured data. For example, when verifying a subsidy calculation, this agent does not "guess" the math; it extracts the numbers and passes them to a Python calculator tool, returning the deterministic result to the conversation.
- **Verifier/Critic Agent:** A crucial component for public administration. This agent reviews the output of the Tool-Use Agent against the retrieved regulation to ensure the logic holds, filtering out potential hallucinations before the final report is generated. One of the possible tools the agent can have access to is external formal reasoners such as the ones used in process mining in order to get better results in the compliance checking.

2.4 Retrieval-Augmented Generation (RAG) in Heterogeneous Environments

2.4.1 Foundational Mechanics of RAG

The standard RAG architecture operates as three layered system designed to decouple an LLM's reasoning capabilities from its static parametric memory. The process follows a "Retrieve-Augment-Generate" workflow:

1. **Retrieval:** Given a user query q, a retriever model identifies a subset of documents $D = \{d_1, d_2, ..., d_k\}$ from an external corpus that are most relevant to the request. In standard implementations, this is achieved by projecting both the query and the documents into a high-dimensional vector space using an *Embedding Model* $f(\cdot)$.
2. **Augmentation:** The retrieved documents are concatenated with the original query to create an enriched prompt $P = (q, D)$. This provides the LLM with a "context window" containing the specific facts required to answer the query.
3. **Generation:** The LLM processes the enriched prompt to produce a response y. Because the model has access to the retrieved context, it can generate grounded answers that cite specific evidence, significantly reducing the risk of "hallucination"—the generation of factually incorrect but plausible-sounding text.

This paradigm shifts the role of the LLM from a "knowledge base" to a "reasoning engine" that performs open-book examinations on provided data. However, while this works for general information retrieval, its reliance on flat-vector similarity creates the bottlenecks in specialized domains discussed below.

Retrieval-Augmented Generation (RAG) [4] addresses the "knowledge cutoff" and hallucination issues of LLMs by grounding their responses in external, verifiable data. However, the public administration domain introduces a specific challenge: *Data Heterogeneity*.

2.4.2 The Challenge of Heterogeneous Data

Standard RAG pipelines are optimized for unstructured text. Documents are chunked, embedded into dense vectors, and stored in a vector database. Retrieval is performed using Cosine Similarity.

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{4}$$

While effective for finding semantically similar text (e.g., matching a user query to a policy paragraph), this approach fails with **Structured Data** (SQL databases, Excel forms, JSON logs).

- **Loss of Relational Context:** Converting a database row into a simple text chunk often destroys the relational schema (e.g., knowing that "Value A" belongs specifically to "Column B" and is linked to "Table C").
- **Precision vs. Semantics:** Vector search is probabilistic. In compliance, a query for "contracts above €50,000" requires deterministic filtering, not semantic similarity. A vector search might return a contract for €49,000 simply because the text context is similar.

2.4.3 Hybrid Retrieval Strategies

To address RQ1, this thesis explores *Hybrid Retrieval* strategies that unify these disparate data types:

Text-to-SQL and Router Chains Instead of embedding structured data, a **Router** component classifies the user's intent. If the query requires structured data (e.g., "List all forms submitted last week"), the router directs the request to a Text-to-SQL agent which generates and executes a database query. If the query requires interpretative data (e.g., "What is the policy on sick leave?"), it routes to the Vector Store.

Serialization of Structured Data For cases where structured and unstructured data must be compared directly, structured data can be *serialized* into a natural language format to be compatible with vector search. Common techniques include:

- **Table-to-Text:** Converting a row `{id: 101, status: 'pending'}` into "The application with ID 101 is currently in pending status."
- **Parent-Child Indexing:** Storing the raw structured data (the child) but embedding a summary description (the parent) for retrieval. When the summary is matched, the full structured object is passed to the LLM.

3 Related Work

This section reviews state-of-the-art research in automated reasoning and retrieval. It identifies the multi-agent framework *PAKTON* as the primary architectural influence for this thesis, while examining advanced Graph-based retrieval methods and the ongoing trade-offs between RAG, Long-Context windows, and Prompt Engineering.

3.1 Multi-Agent Frameworks in Legal Domains

The most significant antecedent to this research is *PAKTON*, introduced by Raptopoulos et al. [19]. This framework represents a paradigm shift from standard "Retriever-Generator" loops to collaborative multi-agent workflows. Designed specifically for Question Answering over long legal agreements, PAKTON addresses the complexity of the legal domain—where a single compliance question often requires checking definitions, cross-referencing multiple clauses, and aggregating scattered information—by simulating the workflow of a human legal team.

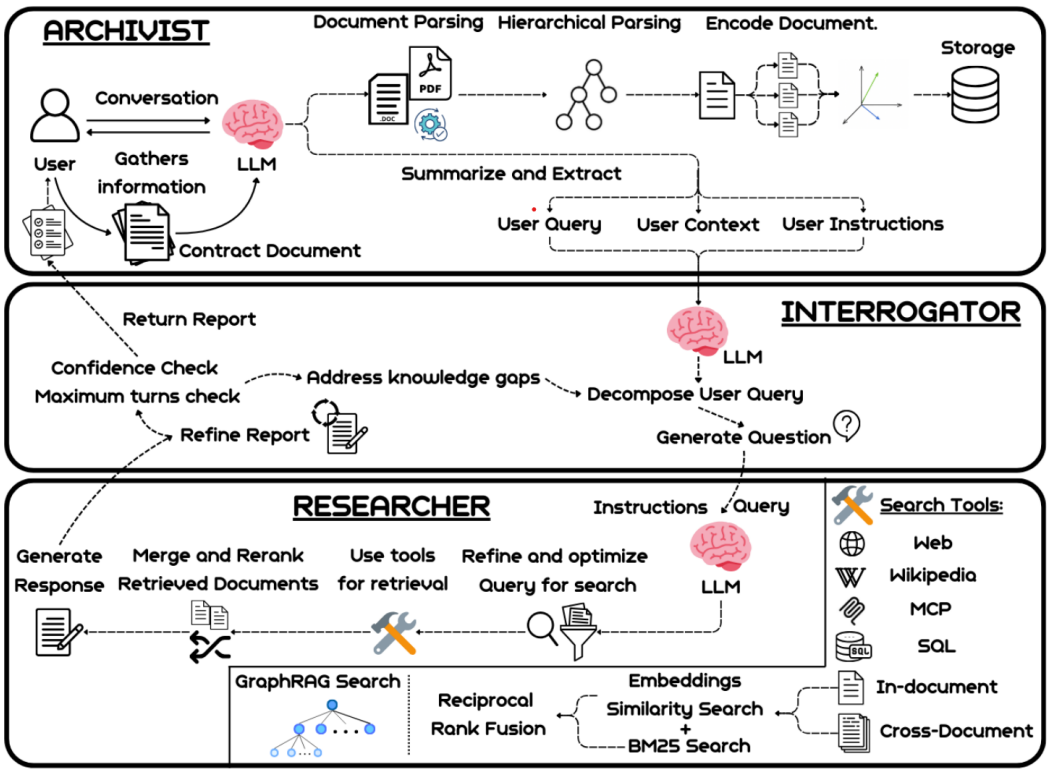

Figure 1: An overview of the proposed PAKTON framework and its internal components. Taken from [19]

3.1.1 Architectural Mechanics

The architecture of PAKTON is defined by the decomposition of reasoning tasks into specialized roles. The *Planner* acts as the orchestrator, decomposing complex user queries into a dependency graph of sub-questions, preventing the Large Language Model (LLM) from losing focus during multi-step reasoning. These sub-tasks are delegated to the *Archivist*, a dedicated retrieval specialist.

Crucially, the Archivist operates using a RAG pipeline rather than relying on the LLM's parametric knowledge. This RAG integration is essential because legal contracts and administrative regulations frequently exceed standard context windows; it allows the Archivist to dynamically fetch only the specific clauses relevant to the current sub-task, thereby minimizing "noise" and hallucination risks.

The retrieved information is passed to the *Analyst*, which synthesizes the answer. A critical innovation in this architecture is the iterative feedback loop; if the Analyst determines that the retrieved documents are insufficient, it signals the Planner to trigger a new search strategy with modified parameters. This self-correcting mechanism mimics the behavior of a human auditor and is essential for maintaining accuracy in high-stakes environments.

However, a critical gap remains in the application of PAKTON. The existing framework is designed for homogeneous unstructured text, comparing legal queries against contract documents. It lacks the capability to process structured data, which is ubiquitous in administrative compliance and similar processes, which we aim to achieve in this project.

3.2 Optimizing Retrieval: Graph-Based Approaches

Standard Retrieval-Augmented Generation (RAG) pipelines typically rely on vector similarity, a method that is computationally efficient but "structurally blind." By treating complex regulations as isolated text chunks, standard RAG often fails to capture the hierarchical and relational nature of legal texts, leading to fragmentation errors where a rule is retrieved but its exception is missed. To address this, recent research has turned to Graph-Based and Hierarchical optimizations.

3.2.1 GNN-RAG: Graph-Based Retrieval and Reasoning

Mavromatis and Karypis [1] introduced *GNN-RAG*, a framework that strategically combines the structural reasoning capabilities of Graph Neural Networks (GNNs) with the linguistic synthesis of LLMs. Unlike standard RAG, which treats documents as isolated vectors, GNN-RAG utilizes the GNN as a "structural reasoner" to identify relevant information within a Knowledge Graph (KG).

Architectural Mechanics The GNN-RAG workflow, as illustrated in Figure 2, follows a three-phase execution logic:

- **GNN-Based Candidate Retrieval:** The process begins with a dense retrieval step where the query is mapped to the graph. A GNN then reasons over the surrounding dense subgraph to identify potential answer candidates. By performing this at the graph level, the system identifies relevant nodes that might not share direct keyword overlap with the query but are topologically significant.
- **Shortest Path Extraction:** For the identified candidates, the system extracts the *shortest paths* connecting the query entities to the potential answers. This step provides the logical "connective tissue" required to understand the relationship between disparate data points.
- **Textualization and LLM Synthesis:** These paths are "verbalized" into natural language and passed to the LLM. The LLM acts as the final decision-maker, using the provided reasoning paths to filter out irrelevant information and generate a grounded response.

Example As shown in the "Retrieval" box of Figure 2, when the system processes the query regarding the languages of Jamaica, the GNN retrieves several candidates, including "English," "French," and "Caribbean." However, the "Reasoning" phase (visible in the bottom blue box of Figure 2) demonstrates how the extraction of shortest paths creates a deterministic link: `Jamaica` → `official_language` → `English`. By providing these explicit relations (*e.g.,* "Jamaica is located in the Caribbean Sea" and "Jamaica's official language is English"), the LLM can ignore the "French" candidate, which was only retrieved because it was geographically close to Jamaica via the "Haiti" node. This mechanism directly addresses the issue of **Semantic Drift** by forcing the model to rely on explicit relational paths rather than vague vector proximity.

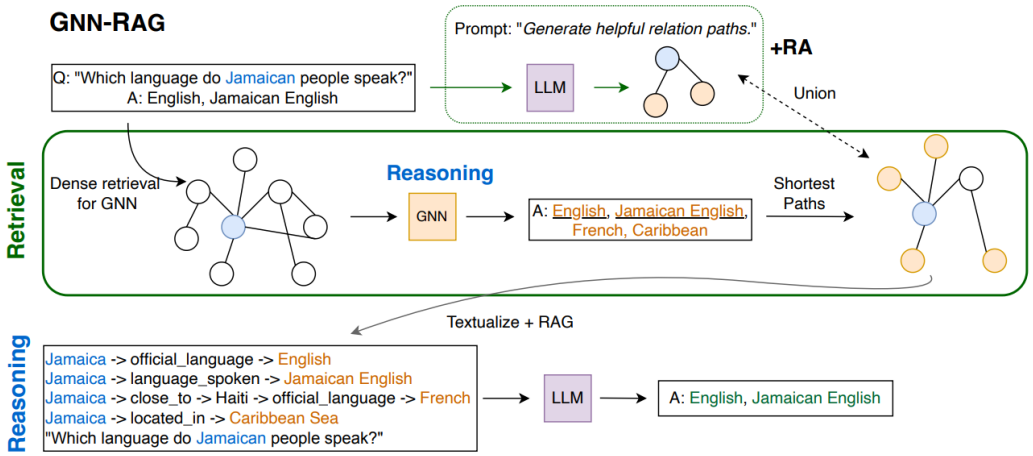

Figure 2: GNN-RAG: The GNN reasons over a dense subgraph to retrieve candidate answers, along with the corresponding reasoning paths (shortest paths from question entities to answers). The retrieved reasoning paths –optionally combined with retrieval augmentation (RA)– are verbalized and given to the LLM for RAG. Taken from [1]

Advantages of GNN-Driven Retrieval Utilizing a GNN-based approach offers a significant performance advantage over standard embedding-based RAG by enabling the system to move beyond probabilistic semantic similarity toward deterministic structural reasoning. By traversing graph edges, the GNN resolves the issue of *Multi-hop Fragmentation*, allowing the system to logically connect a "Payment Event" to a "Regulatory Article" through multiple intermediate steps—such as invoices or inspection reports—that a vector search would likely overlook. Furthermore, the GNN's *Topological Awareness* ensures that hierarchical weights in public administration data are respected, while the extraction of explicit shortest paths provides a transparent audit trail. This transforms the retrieval process from a "black box" keyword match into a verifiable reasoning chain, ensuring that the LLM's final conformance verdict is grounded in a complete and logically sound context.

3.2.2 Med-Graph-RAG: Triple-Layered Evidence and U-Retrieval

In parallel to general frameworks, Wu et al. [20] introduced *Med-Graph-RAG*, a system designed for the medical domain which shares the zero-error tolerance requirements of public administration. The framework addresses the limitations of standard Graph-RAG—specifically high computational costs and "context flooding"—through two key innovations: Triple Graph Construction and the U-Retrieval strategy.

Triple Graph Construction Unlike naive graphs that only link document chunks, Med-Graph-RAG builds a hierarchical "Med-MetaGraph" consisting of three layers:

- **Raw Data Layer:** Segments of private user documents (e.g., patient records or local administrative files).
- **Scientific Paper Layer:** Credible medical literature that provides evidence-based backing for the raw data.
- **Vocabulary Layer:** Foundational medical dictionaries (e.g., UMLS) that provide standardized definitions.

This triple-linking ensures that every retrieved entity is not only connected to a document but is also grounded in authoritative definitions and scientific evidence, creating a high-fidelity "reasoning chain."

The U-Retrieval Strategy To optimize retrieval, the authors propose a "U-shaped" process that avoids the expensive community detection used in other Graph-RAG systems. As defined in the workflow of

3.2.3 Med-Graph-RAG

, the process follows a dual-sweep logic:

1. **Top-Down Precision (The "Down" of the U):** The system uses an LLM to generate medical tags for the user query. It then traverses a hierarchical tag structure (from broad categories to specific medical codes) to index the most relevant subgraph. This pinpoints the "core" evidence without scanning the entire database.
2. **Bottom-Up Refinement (The "Up" of the U):** Once the specific nodes are found, the system refines the initial response by progressively integrating broader, higher-level summaries from the parent tags. This "zooming out" ensures the LLM understands the global context (e.g., a patient's overall history) while focusing on the specific local evidence (e.g., a recent lab result).

Illustrative Example Consider a query regarding a specific drug contraindication in an administrative health claim:

- **Top-Down:** The system identifies the tag `Medication` → `Anticoagulants` → `Warfarin`. It retrieves the specific subgraph for Warfarin, finding a conflict with a patient's recent `Aspirin` prescription.
- **Bottom-Up:** The system then pulls the higher-level summary of the patient's `Chronic Conditions`, realizing the patient has a history of `GI Bleeding`.
- **Result:** Instead of just saying "Don't mix these drugs," the system provides a grounded answer: "Warfarin and Aspirin conflict (Specific Evidence), which is particularly dangerous given the patient's history of GI Bleeding (Broad Context), as defined by Medical Guideline X (Scientific Layer)."

This approach directly benefits public administration systems by ensuring the LLM receives the minimum necessary context to be accurate, thereby optimizing reasoning speed while maintaining a verifiable audit trail back to regulatory "dictionaries."

3.2.4 LATTICE: Multi-Layer Knowledge Graphs

A significant advancement in graph-augmented retrieval is the *LATTICE* framework proposed by Wang et al. [21]. While traditional Graph-RAG methods often focus on a single flat knowledge graph, LATTICE introduces a **Multi-Layer Knowledge Graph** structure that organizes information into three distinct tiers: the *Concept Layer* for high-level abstractions, the *Entity Layer* for specific subjects, and the *Evidence Layer* for raw textual grounding.

The framework utilizes a "Cross-Layer Traversal" mechanism, which allows the system to navigate from broad regulatory concepts down to granular factual evidence. This hierarchical partitioning directly addresses the "semantic gap" often found in complex domains, where high-level rules must be mapped to low-level administrative data. By structuring the graph in layers, LATTICE demonstrates a marked improvement in handling multi-hop queries that require both conceptual understanding and specific evidence retrieval. This multi-tiered approach provides a strong theoretical foundation for the *Forest of Graphs* architecture proposed in this thesis, particularly in its ability to separate regulatory logic from administrative event data while maintaining a traversable link between them.

3.3 The Context Debate: RAG vs. Long-Context Windows

A central debate in modern Natural Language Processing concerns the necessity of RAG given the emergence of Long-Context (LC) LLMs capable of processing over one million tokens. Zhu et al. [22] provided a comprehensive benchmark comparing these approaches, offering critical insights for this thesis.

Their study demonstrates that while Long-Context models excel at global summarization tasks, they suffer from the "Lost-in-the-Middle" phenomenon when tasked with precise information extraction. When the relevant "needle" (e.g., a specific fine amount) is buried in the middle of a massive context window, the model's retrieval accuracy degrades significantly. Consequently, the authors conclude that RAG remains superior for tasks requiring high precision and cost-efficiency.

Furthermore, the study highlights that prompt engineering techniques, specifically Chain-of-Thought (CoT), are orthogonal to the retrieval method. The highest performance benchmarks were not achieved by RAG or Long-Context models in isolation, but by a hybrid approach combining RAG with CoT prompting. RAG provides the factual grounding to reduce hallucinations, while CoT provides the reasoning structure to minimize logical errors. This finding validates the proposed methodology of this thesis, which integrates robust hybrid retrieval with the use different LLM agents to achieve the performance that is expected of the system.

4 Proposed Solution: A Multi-Agent Graph-RAG Framework

To address the limitations of current compliance checking systems—specifically data heterogeneity (**RQ1**), context loss in long-documents (**RQ2**), structural dependency discovery (**RQ2**), agentic reasoning performance (**RQ3**), and the lack of process awareness (**RQ4**)—this thesis proposes a novel **Process-Aware Multi-Agent Graph-RAG Framework**.

Unlike monolithic RAG pipelines that treat compliance as a simple retrieval task, this solution models compliance checking as a *collaborative reasoning process* between specialized AI agents operating over a unified Knowledge Graph.

4.1 System Architecture

The proposed architecture extends the PAKTON framework by introducing three critical innovations designed to move beyond simple document retrieval toward adversarial process validation. While PAKTON established a foundation for multi-agent coordination, it often suffers from structural blindness, which is the inability to maintain semantic integrity across deep document hierarchies or enforce rigid procedural sequences.

The first innovation is a **Hybrid Graph Retrieval Method**, which addresses the *Semantic Dilution* found in standard flat RAG where critical edge cases are often averaged out in vector space. By implementing a **Forest of Graphs** structure that utilizes LLM-generated **Natural Language Resumes** at each cluster level, the system provides a semantic map for the agents. This allows the framework to solve the problem of context fragmentation by ensuring that agents navigate through high-level concepts while preserving granular, specific facts within their broader regulatory context.

Secondly, we introduce a **Multi-Agent Reasoning** workflow to solve the issue of *Context Flooding*, which typically occurs when a single LLM is forced to handle both data retrieval and legal interpreta-

tion simultaneously. By decoupling these responsibilities into specialized roles—including the Archivist, Clerk, Specialists, and a central Report Agent—the framework significantly reduces the cognitive load on individual agents. This separation of concerns enables iterative refinement loops where agents can critique each other's logic, leading to a measurable reduction in hallucination rates and more robust interpreted outcomes.

Finally, the architecture incorporates a **Process Alignment Module** to address a fundamental deficiency in current compliance systems: the inability to validate *sequence* alongside existence. While standard RAG architectures can successfully retrieve isolated facts—confirming that a document exists—they typically lack the temporal awareness required to detect if a mandatory procedural step was skipped or executed out of order, a critical failure point in administrative law. To solve this, our framework introduces a **Process Graph** extraction phase that converts unstructured regulatory text into machine-readable logic constraints (e.g., *Payment* must follow *Approval*). This module acts as a strict logical validator that intercepts agent reasoning; if an agent proposes a "Compliant" status for a transaction that violates these extracted temporal dependencies, the system triggers a mandatory halt. This mechanism effectively constrains the probabilistic nature of the LLM with deterministic procedural rules, preventing the generation of "False Compliance" reports and ensuring that the final output adheres to the rigid legal workflows required by public administration.

4.2 Layer 1: Hierarchical "U-Retrieval"(Addressing RQ1 and RQ2)

To address data heterogeneity and context fragmentation, we replace standard flat retrieval with a **Hierarchical U-Retrieval** architecture, inspired by the "dual-sweep" strategy of Med-Graph-RAG [20]. Instead of a flat list of documents, the system constructs a **Forest of Graphs**—a tree-like structure where leaf nodes are individual document graphs, and parent nodes are semantic clusters.

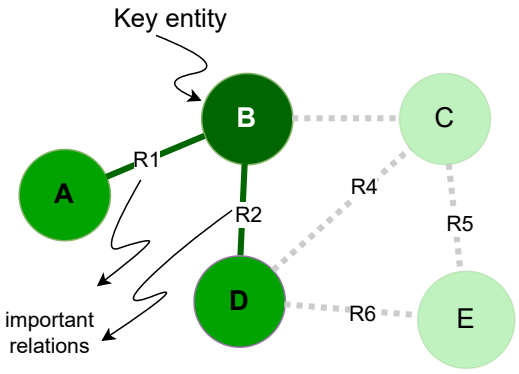

Figure 3: Connected Graph showing the Selection of the most important Entities and relations

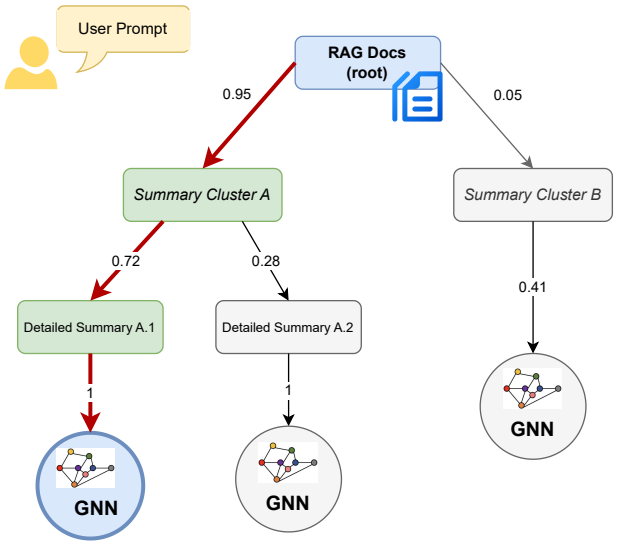

Figure 4: Tree graph containing the summaries and graphs for each docs, choosing the correct path using GNN weights following a specific prompt

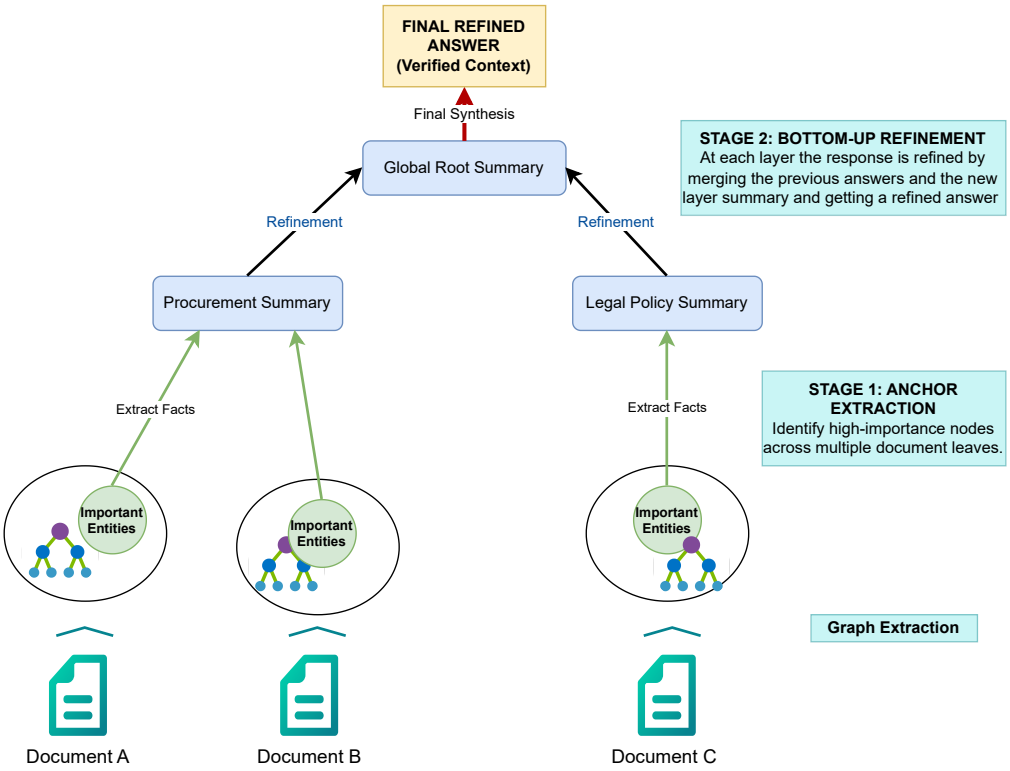

Figure 5: Figure shows the operation that after getting the most important entities, for each layer use the previous result and merge them it the summary of the current layer

This layer operates in three distinct phases:

Phase 1: Bottom-Up Indexing and Resume Generation

At ingestion time, the system builds the tree from the bottom up to ensure higher layers retain critical entity information. Unlike standard vector clustering, we utilize **Natural Language Resumes** to support the reasoning capabilities of the Planner Agent:

- **Level 0 (Document Graphs):** Each document (unstructured PDF or structured Form) is converted into a local knowledge graph. Key entities (e.g., "Vendor A", "Article 5") are extracted.
- **Level 1+ (Cluster Propagation):** Documents are clustered based on semantic similarity. Crucially, the system identifies the *Most Important Entities* (via centrality measures) within each cluster.
- **Example Summary:** Instead of opaque vectors, each cluster generates a Natural Language Resume dynamically constructed from the propagated entities (e.g., *"This cluster contains 50 invoices related to Vendor A and Public Works Contracts"*). This allows specific signals to bubble up to the root, preventing important details from being lost in broad averages.

The construction of the Cluster Resume is a recursive **merging and synthesis** process. As the system moves up the hierarchy, the resumes of child nodes are aggregated into a higher-level summary. This process compresses the informational content of thousands of sub-nodes into a single, agent-readable narrative, preserving only the distinct patterns and critical anomalies required for high-level reasoning.

Phase 2: Top-Down Retrieval (The Reasoning Flow)

At query time, the **Planner Agent** traverses this tree using a top-down decision process (see figure 4) :

1. **Global Context Check:** The agent reads the Root Resume to understand the global scope.
2. **Branch Selection:** Using the detailed Resumes, the agent explicitly selects relevant sub-clusters (e.g., *"I will ignore 2023 and focus on the Vendor A branch"*).
3. **Graph Drills:** Once it reaches the target Document Graph, it performs granular extraction.

Phase 3: Bottom-Up Response Refinement

To generate "deep connections," we implement a final refinement phase that propagates granular findings back up the hierarchy.

- **Local Subgraph Extraction:** The system activates the local GNNs of the selected leaf nodes to extract specific facts (e.g., specific dates, monetary values) showed in figure 3.
- **Upward Semantic Propagation:** These facts are pushed up to refine the parent Cluster Resumes. For example, a generic tag like "International Contracts" is refined to state: *"Contains International Contracts, specifically involving Vendor X (Foreign Entity)."* This operation is detail in figure 5
- **Global Synthesis:** The **Report Agent** uses these refined resumes to identify cross-cutting concerns (e.g., linking a delay in Document A to a clause in Document B) that would be invisible to a standard RAG system.

4.2.1 The Role of Graph Neural Networks in Structural Retrieval (Addressing RQ2)

While standard RAG relies on the similarity of isolated text embeddings, the integration of Graph Neural Networks (GNNs) enables the system to capture structural dependencies inherent in administrative data. In this framework, the GNN operates over the Knowledge Graph via *message passing*, where the representation of a specific node is iteratively updated with information from its topological neighbors.

This process computes *relational embeddings* that allow the system to identify relevant subgraphs based on the probability of a "path" existing between an administrative case and a regulatory constraint. As discussed in Section **??** this graph-based approach ensures that the retrieval phase is context-aware. For instance, retrieving a specific invoice automatically prioritizes the legal articles governing that vendor category, effectively mitigating errors by the loss of critical context.

Design Challenge: Subgraph Topology and Information Loss

A critical challenge in constructing the leaf layer (Level 0) is the trade-off between *semantic impact* and *connectivity*. We analyzed two structural paradigms: anchoring all entities to a central "Document Node" versus constructing independent subgraphs connected solely by extracted relations.

While the latter approach (Relational Subgraphs) significantly amplifies the impact of specific entities—allowing agents to reason over high-confidence logical chains without traversing metadata noise—it introduces a risk of *structural information loss*. By relying exclusively on explicit relations, the system risks creating disconnected "islands" of data. In this scenario, latent contextual links (e.g., two entities appearing in the same paragraph without a syntactically extractable relationship) are severed, potentially hiding subtle dependencies that a document-centric connection would have preserved.

4.2.2 The Synthesis Duality: Vector-Grounding vs. Summaries (Addressing RQ2)

A core investigative objective of this research is to evaluate the trade-off between **Raw Vector-Grounding** and **Summary-Guided Synthesis** during the final reasoning stage. This duality presents a choice between computational precision and human-centric explainability.

Paradigm A: Raw Vector-Grounding In the Vector-Grounding paradigm, the system generates responses by operating directly on granular data chunks retrieved via embedding similarity. Utilizing embedding vectors as the primary connective tissue makes the system significantly quicker and simplifies the search operations through standard nearest-neighbor algorithms. This approach maintains high technical fidelity and minimizes the risk of loss of information inherent in multi-layer abstraction. However, it risks *contextual blindness*, where the system fails to synthesize cross-document dependencies that are not explicitly captured in a single vector space.

Paradigm B: Summary-Guided Reasoning Conversely, we explore a **Summary-Guided** approach, where the LLM uses the "Natural Language Resumes" as the primary context for reasoning. While this method provides superior insight into the system's internal logic and makes the output more interpretable for human auditors, it introduces significant research risks:

- **Semantic Drift:** The recursive summarization process might omit low-frequency but high-impact details found in the leaf nodes.
- **Justification Hallucination:** The system may generate a compelling narrative that serves as a logical justification, yet we cannot be strictly certain that the LLM is using this narrative as its actual functional path rather than an rationalization of the solution it came up with.

By implementing both pathways as experimental variables, this thesis aims to quantify whether the qualitative "process-awareness" provided by LLM-synthesized summaries justifies the increased computational overhead and the potential loss of granular precision compared to pure vector-based retrieval.

4.3 Layer 2: Multi-Expert Agentic Workflow

To move beyond standard "Retrieve-and-Summarize" pipelines, this framework adopts a federated multi-agent architecture inspired by the PAKTON framework (Section 3.1). The core philosophy is the decoupling of **reasoning** (domain experts), **orchestration** (report synthesis), and **data navigation** (retrieval). This separation ensures that complex legal reasoning remains focused and explainable, avoiding the "context flooding" common in monolithic LLM applications **(Addressing RQ3)** .

4.3.1 Agentic Taxonomy and Responsibilities

The workflow is distributed across five specialized roles, each operating with specific system prompts and defined tool access:

- **The Archivist (Ingestion Agent):** Operates asynchronously to the user query. Its primary function is document processing, entity extraction, and the continuous construction of the Unified Knowledge Graph. It handles the "Bottom-Up Indexing" described in Layer 1 to ensure new legislation or records are immediately navigable.
- **The Clerk (Context & Triage Agent):** Acts as the first point of contact for user prompts. It performs intent classification and entity recognition to define the "Audit Mission." It identifies which legal domains are triggered (e.g., procurement, tax, labor) and passes a structured context object to the Report Agent.
- **The Report Agent (Orchestrator):** The central authority of the system. It manages the lifecycle of a query by delegating sub-tasks to Specialist Agents. It is responsible for the *Refinement Loop*, where it evaluates expert outputs for consistency and depth. If an explanation is insufficient, it re-triggers the agents with corrective feedback until a satisfaction threshold or maximum iteration count is reached.
- **Specialist Agents (Domain Experts):** These agents (e.g., Public Procurement Expert, Financial Auditor) utilize the **ReAct (Reasoning + Acting)** paradigm. They receive specific sub-tasks and use an internal "thought" process to decide which information is required from the Retriever.
- **The Retriever Agent:** A centralized interface for data access. It executes the "U-Retrieval" strategy (Top-Down navigation and Bottom-Up refinement) to provide specialists with verified ground truths. By centralizing retrieval, the system maintains a consistent "source of truth" and allows for the integration of external computational tools (e.g., Python math interpreters, SQL queries, etc).

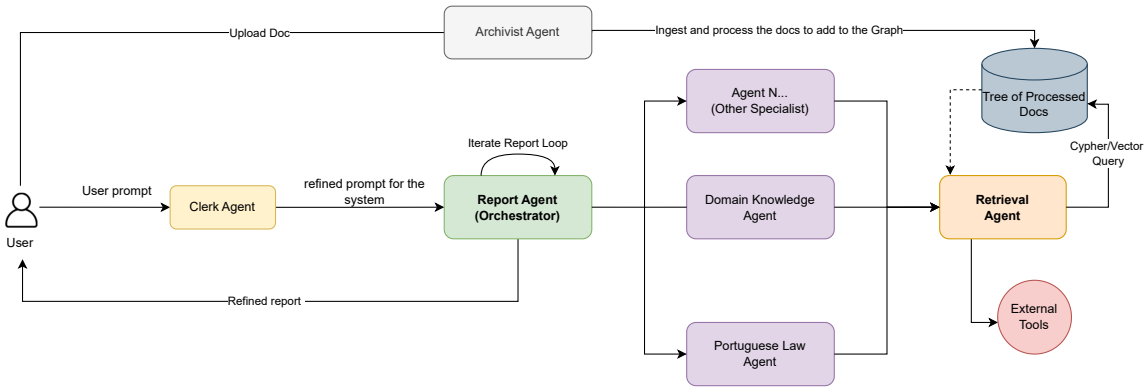

Figure 6: The Proposed Process-Aware Multi-Agent Graph-RAG Architecture. The User initiates an audit, which is orchestrated by the Report Agent. This agent delegates reasoning to specialist agents (Legal, Domain), which query the Unified Knowledge Graph via a translation layer. The system iterates until a consensus is reached.

4.4 Layer 3: Process Alignment via Compliance Checking and Feedback (Addressing RQ4)

High-stakes compliance failures in regulated workflows often arise from procedural deviations—missing prerequisites, misordered steps, absent approvals, or violated deadlines—even when relevant documents exist. Moreover, LLM-based pipelines can overlook critical evidence in long contexts, motivating explicit mechanisms that verify procedural validity rather than assuming it, yet these mechanisms are not always available [23].

Layer 3 introduces a **process-alignment (compliance-checking) layer** that treats procedural validity as a verifiable property of the system's output. The layer operationalises conformance checking ideas from process mining, where observed executions are compared against an explicit process specification to identify deviations [24, 25]. This layer can also be viewed as an extension to the specialist agents of layer 2, focusing on process alignment with constraints.

Constraint specification. The framework maintains an explicit **process constraint specification**, encoding precedence relations, mandatory checks, role/approval requirements, timing constraints, and exception clauses. Constraints can be (i) extracted from normative text (regulations, policies, guidelines) by a specialised process extraction agent, and/or (ii) refined using operational traces when event logs are available, consistent with conformance checking practice [24, 26]. These are converted into *latent process graphs* to be used as ground truth proxies.

Compliance checking over agent traces. During execution, the system records a structured **audit traces** (retrieved evidence, inferred steps/events, ordering, and citations). A compliance checker compares this trace to the constraint specification to detect and localise deviations such as missing steps/documents, ordering violations, missing approvals, and temporal breaches [25].

Verification-and-correction feedback loops (Compliance guardrails). When non-conformance or uncertainty is detected, the layer applies **compliance guardrails** by checking the agent-produced audit trace against the process constraint specification. If the agent proposes a conclusion that violates procedural constraints (e.g., approving a payment before the Receipt Date), the module flags the deviation and triggers a **verification-and-correction loop**: targeted re-retrieval, plan refinement (e.g., add a check or handle an exception), and report revision. The output either restores conformance with grounded evidence or explicitly reports non-conformance with an appropriate constraint-to-evidence explanation.

4.5 Implementation Setup

The proposed solution will be implemented as a functional prototype to validate the research questions.

4.5.1 Technology Stack

- **Orchestration:** LangGraph will be used to implement the cyclic multi-agent workflow (PAKTON adaptation). Unlike traditional Directed Acyclic Graph (DAG) frameworks, LangGraph supports *cyclic workflows*. This is critical for **RQ2**, as it enables the "Refinement Loop" where the Report Agent can recursively critique and improve the outputs of Specialist Agents until deterministic compliance is reached.
- **Graph Database:** Neo4j will store the Unified Knowledge Graph, supporting both vector search (for text) and Cypher queries (for relationships). Standard vector databases are "structurally blind." Neo4j is utilized to store relationships as first-class citizens, enabling the *multi-hop reasoning* required to link a specific administrative act to its governing legislation (Addressing **RQ1**). Its support for *index-free adjacency* ensures that deep contextual explorations across disconnected documents remain computationally efficient.
- **LLM Backbone:** The system will be tested with state-of-the-art models (gpt-4o, gemini-3-pro, etc.) to find the LLM with the best results. The architecture of the system is very dependent on the ability of LLMs to output good results. Due to the multiple available state-of-the-art models that are available, it is necessary to test the system in order to choose the ones with the best results.

4.6 Datasets

4.6.1 LegalBench-RAG

To test and improve the system, we need a dataset that actually challenges the way our architecture works. We have chosen **LegalBench-RAG** [27] as our primary benchmark. This dataset is specifically built for RAG systems and focuses on the legal field, which is closely related to the administrative compliance tasks our system is designed to handle. **LegalBench-RAG** is a comprehensive synthesis of multiple specialized legal datasets, including *CUAD* (contract understanding), *MAUD* (merger agreements), and *ContractNLI* (natural language inference for contracts). By combining these sources, the benchmark provides a diverse range of legal documents that vary in length, complexity, and linguistic style.

One of the most important reasons for choosing this dataset is its focus on **multi-hop reasoning**. In many RAG benchmarks, the answer is usually hidden in a single paragraph. However, in real-world compliance, you often need to connect the dots between different sections or even different documents—such as checking a specific contract clause against a general regulation. This is what we call a "multi-hop" problem.

By using LegalBench-RAG, we can see if our **Forest of Graphs** and **Multi-Agent** setup are actually better at following these complex trails than a standard search. Since the dataset provides clear ground truths across various tasks (summarized in Table 1), we can measure exactly where the system succeeds or fails. It also helps us find the right balance between the system trying to generate an answer and knowing when to "refuse" because the evidence isn't clear enough.

Table 1: Statistics of the LegalBench-RAG benchmark components (adapted from [27]).

Dataset Component	Task Type	Doc Count	Avg. Tokens/Doc
CUAD	Extraction	510	9,234
MAUD	Classification	47	28,450
ContractNLI	Inference	607	2,140
LegalBench (Selection)	Multi-hop QA	156	4,820

Data Structure Example The following extract 1 illustrates the Question-Answering (QA) pair structure within the LegalBench-RAG dataset, showcasing the mapping between natural language queries, source file paths, and the specific text spans containing the grounded answers.

```
{
  "tests": [
    {
      "query": "Consider the Non-Disclosure Agreement (...) to the Confidential
          Information?",
      "snippets": [
        {
          "file_path": "contractnli/CopAcc_NDA-and-ToP-Mentors_2.0_2017.txt",
          "span": [11461, 11963],
          "answer": "Any and all proprietary rights, including but not limited to
              rights to and in inventions(...) with the Copernicus Accelerator 2017
              ."
        }
      ]
    },
    {
      "query": "Consider the Non-Disclosure Agreement (...) include technical
          information?",
      "snippets": [
        {
          "file_path": "contractnli/CopAcc_NDA-and-ToP-Mentors_2.0_2017.txt",
          "span": [7752, 8016],
          "answer": "Confidential Information means any Idea disclosed to Mentor,
              (...) and/or its description and any conclusions."
        }
      ]
    }
  ]
}
```

Listing 1: Extract of the QA pairs in LegalBench-RAG. Adapted from Pipitone et al. [27].

4.6.2 ECHR: Compliance Audit Simulation

To evaluate the proposed framework's capability in automated compliance auditing, this research utilizes the **European Court of Human Rights (ECHR)** dataset [28]. This dataset is uniquely suited for *Compliance Audit Simulation* because it provides a direct mapping between unstructured factual narratives and specific regulatory outcomes (Articles of the Convention).

The dataset is implemented within the framework as follows:

- **The Input (Audit Data):** The "Facts" sections of judicial cases serve as the target audit data—comprising long-form, unstructured narratives describing administrative or state actions.
- **The Knowledge Graph (The Law):** The ECHR Articles and their associated case law are indexed as the regulatory Knowledge Graph within the *Forest of Graphs* architecture. Each article is represented by a *Natural Language Resume* that summarizes its legal scope and precedents.
- **The Retrieval Task:** Given the facts of a new case, the *Retriever Agent* must navigate the hierarchy to identify which specific Articles are relevant to the alleged conduct.
- **The Conformance Check:** The *Specialist Agent* performs a conformance check to determine if the facts constitute a violation (*non-conformance*) or adherence (*conformance*).

By comparing the system's output against the Court's actual judicial decisions (Ground Truth), we can quantitatively measure the framework's precision in complex legal reasoning and its ability to mitigate hallucinations. The statistics of the ECHR corpus used in this study, as defined in [28] and [29], are summarized in Table 2.

Table 2: Comparative Statistics of the ECHR 2019 and ECHR 2021 (Rationale) Datasets

Metric	ECHR 2019 (LJP)	ECHR 2021 (Rationale)
Primary Task	Judgment Prediction	Alleged Violation Prediction
Total Cases	11,478	11,000
Train / Dev / Test Split	7,100 / 1,380 / 2,998	9,000 / 1,000 / 1,000
Label Set (Articles)	66	40
Avg. Facts per Case	41 paragraphs	45 paragraphs

4.6.3 BPI Challenge 2018: Multi-Document Process Log

The **BPI Challenge 2018 (BPIC18)** dataset [30] provides a structured event-based foundation for evaluating the framework's ability to handle interdependent process lifecycles. Derived from the German agricultural subsidy application process, this dataset reflects a *document-centric* workflow where a single case consists of multiple synchronized document types, including land parcel declarations, inspection reports, and payment applications.

The **BPI Challenge 2018** dataset captures a real-world **public administration** workflow for handling **EU Common Agricultural Policy (CAP) subsidy applications** by a German paying agency. It is released as **event logs in XES format** (the standard in process mining), where each **trace corresponds to a single application** and each trace contains a time-ordered sequence of **administrative events** (procedural steps). This structure enables direct analysis of procedural ordering, repetitions, and delays, and supports **conformance checking** by comparing observed executions against a reference or inferred procedure.

Information is conveyed at two levels. **Case-level (trace) attributes** summarize the application context (e.g., identifiers, year, department/program indicators, scheme participation flags, and binned financial aggregates), and include **outcome-related indicators** (e.g., rejection and other "undesired outcome" signals such as late or reopened cases) that can serve as an **accept/reject** proxy. **Event-level attributes** describe each step (activity name, timestamp, document type, subprocess, resource/role, and success flags). A distinctive feature is the dataset's **document-centric organization**: besides the integrated application log, it provides **separate logs per document type** (linked back to the parent application), enabling conformance analysis at both application and document-subprocess granularity.

Within the proposed *Forest of Graphs* architecture, each document type is modeled as a distinct subgraph, while the governing **Common Agricultural Policy (CAP)**—specifically EU Regulation No. 1306/2013—serves as the regulatory Knowledge Graph. The evaluation focuses on the *Specialist Agent's* capacity to perform conformance checking across these disparate entities. Specifically, the system must verify that procedural dependencies are met, such as ensuring that payment authorization nodes are only reached after the corresponding inspection sub-graphs have been closed without findings. By utilizing BPIC18, the research demonstrates how the Graph-RAG framework integrates structured administrative event nodes with high-level regulatory summaries to identify procedural non-conformance in complex, high-volume government workflows.

The statistics of the BPI corpus used in this study are summarized in Table 3

4.7 Overall System Capabilities

This layered approach improves the final response by generating an explicit reasoning trace. The core contribution is an end-to-end architecture that goes beyond "retrieve-then-generate" by combining structured representations, agentic decomposition, and process-alignment checks to improve reliability, traceability,

Table 3: Summary of the BPI Challenge (BPIC) 2018 dataset.

Attribute	Value
Workflow process	Handling applications for EU direct payments
Start date	4 May 2014
End date	19 January 2019
# Application (Traces)	43,809
# Events	2,514,266
# Activities	165
Avg. of events per trace	57 events
The shortest trace	24 events
The longest trace	2973 events

and procedural validity in complex auditing tasks. Instead of an opaque binary flag, the output provides a transparent audit trail that identifies specific procedural breaks—such as a payment authorization node being reached without a corresponding "closed" status on an inspection report. This allows for a "proof of non-conformance" that maps administrative events directly to regulatory requirements, significantly enhancing interpretability for public administration audits.

At a high level, the system provides four key capabilities. **First**, it unifies heterogeneous evidence by linking unstructured normative text (laws, regulations, policies, guidelines) with structured operational records (forms, databases, event logs) in a connected graph representation. **Second**, it improves retrieval reliability in hierarchical and cross-referenced corpora by combining hierarchical semantic indexing and graph-based navigation, reducing semantic drift and multi-hop fragmentation relative to flat-vector retrieval. **Third**, it introduces a multi-agent workflow that decomposes auditing into specialised roles (planning, retrieval, tool-based verification, and report synthesis), enabling explicit reasoning traces and reducing hallucination risk via cross-checking. **Fourth**, it incorporates a **process-alignment (compliance-checking) layer** that evaluates the system's audit trace against explicit procedural constraints, and triggers **verification-and-correction feedback loops** (targeted re-retrieval, plan refinement, and report revision) when non-conformance or uncertainty is detected.

5 Evaluation

We'll be using the different datasets proposed in the previous Dataset section and using different metrics to be able to test and gain a deeper understanding of the different sections of the system architecture.

5.1 Direct Evaluation of RAG

We employ a combination of traditional statistical metrics and RAG-specific indicators to quantify performance. Like the paper PAKTON [19] we use recall and precision to quantify the performance of the architecture and we also use F1-score to get better understanding between the two.

Precision, Recall, and F1-Score: These will be our primary metrics for measuring *Retrieval Accuracy*, providing a quantitative assessment of the system's ability to navigate the Knowledge Graph.

- **Precision** measures the proportion of retrieved documents or nodes that are truly relevant to the query. High precision indicates a reduction in "data noise," ensuring the agents are not overwhelmed by irrelevant context:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

- **Recall** is critical for legal compliance, as it measures the system's ability to find *all* mandatory regulatory clauses. In the context of an audit, a high recall score ensures that no "gold standard"

evidence is omitted from the final reasoning path:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

- **F1-Score** provides the harmonic mean of Precision and Recall, serving as a single metric to evaluate the overall effectiveness of the retrieval strategy. It is particularly useful for identifying the optimal trade-off between being thorough (Recall) and being concise (Precision):

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

Where **TP** (True Positives) represents correctly retrieved relevant evidence, **FP** (False Positives) refers to irrelevant data mistakenly retrieved, and **FN** (False Negatives) indicates mandatory evidence that the system failed to capture.

In our framework, the triad of **Precision, Recall, and F1-Score** is interpreted as a collective measure of operational trust. **Recall** acts as the primary safety metric, ensuring all mandatory regulatory evidence is captured to prevent "False Compliance" errors. **Precision** serves as an efficiency filter, mitigating agent reasoning fatigue by removing irrelevant data noise. A high **F1-Score**, specifically weighted toward Recall, defines our "system success" threshold.

5.2 End-to-End Decision Evaluation:

Upon evaluating the quality of the RAG-based retrieval and reasoning components, we further assess the impact of the specialised **multi-agent architecture** on the final system decision (verdict). To this end, we conduct comparative evaluations on both datasets introduced in Section 4.6. For the **ECHR dataset** (Section 4.6.2), the system is tasked with predicting whether a case results in a violation or non-violation of the Convention, based on the case description and supporting legal material. For the **BPIC18 dataset** (Section 4.6.3), the system predicts whether a process instance is approved or rejected with respect to the prescribed procedural constraints. Performance is measured using **decision accuracy**, and **F1-score**, allowing us to quantify improvements attributable to the multi-agent setup over single-agent or flat RAG baselines.

The framework's end-to-end performance is evaluated as a binary classification task using the ECHR dataset [28], where the system predicts a verdict of either "Violation" or "No-Violation" for specific Articles. To move beyond simple metrics measuring performance on a binary decision we would also like to assess the reasoning and understanding of the process. To that end, we aim to ask the agents to provide an explanation on the final decision and measure the grounding of the generated explanations against the "gold standard" evidence provided in the different datasets. This can include the passages provided in the LegalBench-RAG benchmark, the curated facts provided in ECHR and the event traces identified in BPIC18. This evaluation utilizes semantic relationship metrics to calculate the proximity between the system's output and the expected rationale. By analyzing how closely the model's cited evidence aligns with human-annotated legal precedents, we can quantitatively verify that the binary decision is derived from correct factual grounding rather than stochastic generation. To that end, we can use models such as BERTSCORE to compute the similarity between a machine-generated and a reference (gold) text, using contextual token embeddings [31].

The structure of the BPIC18 dataset [30] additionally allows us to perform a finer-grained analysis and evaluation focused on the measurable improvement provided by the three-layer Graph-RAG architecture over standard flat models, validated against the events described in the application traces.

5.3 Efficiency Considerations

In addition to effectiveness, the evaluation includes an analysis of **computational efficiency** to characterise the overhead introduced by structure-aware retrieval, multi-agent coordination, and process-alignment checks. Rather than optimising for raw latency, the goal is to quantify **trade-offs between robustness and cost** in compliance-oriented settings.

Efficiency is assessed along three dimensions: (i) **retrieval latency**, comparing flat-vector RAG against hierarchical and graph-based retrieval; (ii) **agentic overhead**, measuring the cost of multi-step planning, verification, and feedback loops relative to single-pass generation; and (iii) **process-alignment cost**, capturing the additional runtime incurred by compliance checking and verification-and-correction loops. Metrics will include end-to-end response time, number of retrieval calls, and token usage per audit task.

This analysis will aim to clarify where additional computational cost is incurred, how it scales with document size and process complexity, and under which conditions the proposed architecture offers a favourable balance between efficiency and procedural reliability.

6 Work Schedule

The research will follow a structured timeline focused on the iterative development of the multi-agent layers and the final benchmarking phase as well as the thesis writing. The current plan can be seen in Figure 7.

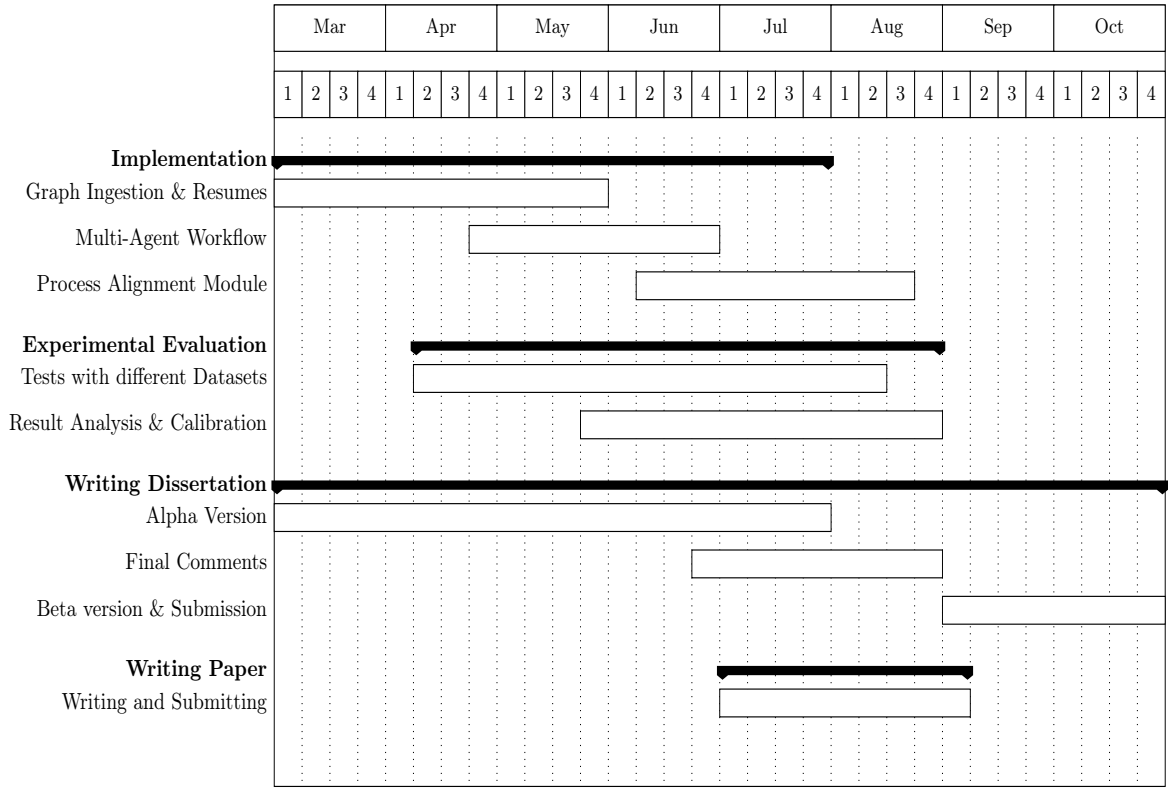

Figure 7: Project Timeline (Gantt Chart)

7 Conclusion

The goal of this thesis is to address a central challenge in high-risk, documentation-heavy domains—most prominently public administration, but also legal and clinical compliance settings: providing automated decision support in contexts where conclusions must be justified against complex normative sources and assessed with respect to explicit procedural requirements. While recent LLM- and RAG-based approaches can retrieve relevant text and produce fluent explanations, they often remain structurally

brittle. In particular, they struggle with deep cross-references and exceptions, fragment evidence across multi-step reasoning chains, and lack principled mechanisms for checking whether generated conclusions are consistent with the underlying process.

To address these limitations, this work aims to develop a Process-Aware Multi-Agent Graph-RAG framework that goes beyond monolithic retrieval-and-generation pipelines. The first intended contribution is a structure-aware retrieval layer that organizes information into hierarchical and graph-based representations, improving robustness to semantic drift and multi-hop fragmentation in cross-referenced corpora. The second contribution is an agentic workflow that decomposes auditing into specialized roles such as planning, retrieval, verification, and synthesis, enabling iterative cross-checking and more controlled intermediate outputs. The third contribution is a process-alignment (compliance-checking) layer that evaluates intermediate traces and final conclusions against explicit procedural constraints and triggers verification-and-correction feedback loops when non-conformance or insufficient support is detected. Together, these components are intended to support more traceable and process-aware compliance reports, while making the trade-offs between robustness and computational cost explicit.

The thesis will empirically assess the proposed architecture on complementary legal and process-oriented datasets, analyzing both effectiveness and efficiency relative to standard RAG baselines.

Bibliography

[1] C. Mavromatis and G. Karypis, "Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[5] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[7] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[8] A. Radford *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[9] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[10] P. F. Christiano *et al.*, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[11] OpenAI, "GPT-5 System Card," OpenAI Technical Report, Aug. 2025, published August 7, 2025. [Online]. Available: https://cdn.openai.com/gpt-5-system-card.pdf

[12] A. Dubey, A. Jauhri, A. Pandey, A. Keshwam *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[13] DeepSeek-AI, "Deepseek-v3.2: Pushing the frontier of open large language models," 2025. [Online]. Available: https://arxiv.org/abs/2512.02556

[14] Anthropic, "System card addendum: Claude opus 4.1," Anthropic, Tech. Rep., Aug. 2025, released August 5, 2025. Supplements the original Claude 4 System Card from May 2025. [Online]. Available: https://www.anthropic.com/claude-opus-4-1-system-card

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 24 824–24 837. [Online]. Available: https://arxiv.org/abs/2201.11903

[16] C. Snell, J. H. Sheen, K. Xu, A. Kumar, and S. Levine, "Scaling llm test-time compute optimally can be more effective than scaling model size," 2024. [Online]. Available: https://arxiv.org/abs/2408.03314

[17] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023. [Online]. Available: https://arxiv.org/abs/2210.03629

[18] J. Li, Q. Zhang, Y. Yu, Q. Fu, and D. Ye, "More agents is all you need," *arXiv preprint arXiv:2402.05120*, 2024.

[19] P. Raptopoulos, G. Filandrianos, M. Lymperaiou, and G. Stamou, "Pakton: A multi-agent framework for question answering in long legal agreements," *arXiv preprint arXiv:2506.00608*, 2025.

[20] J. Wu *et al.*, "Medical graph rag: Evidence-based medical large language model via graph retrieval-augmented generation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

[21] X. Wang *et al.*, "Lattice: A multi-layer knowledge graph framework for enhanced retrieval-augmented generation," *arXiv preprint arXiv:2410.13217*, 2024. [Online]. Available: https://arxiv.org/abs/2410.13217

[22] J. Lee, A. Chen, Z. Dai, D. Dua, D. S. Sachan, M. Boratko, Y. Luan, K. Guu, and M.-W. Chang, "Can long-context language models subsume retrieval, rag, sql, and more?" *arXiv preprint arXiv:2406.13121*, 2024. [Online]. Available: https://arxiv.org/abs/2406.13121

[23] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, 2024, originally released as arXiv:2307.03172.

[24] W. M. P. van der Aalst, *Process Mining: Data Science in Action*, 2nd ed. Springer, 2016.

[25] J. R. Rehse *et al.*, "A task taxonomy for conformance checking," *Information Systems*, 2025.

[26] A. Burattin, F. M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models," *Expert Systems with Applications*, 2016.

[27] N. Pipitone and G. H. Alami, "Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain," *arXiv preprint arXiv:2408.10343*, 2024. [Online]. Available: https://arxiv.org/abs/2408.10343

[28] I. Chalkidis, I. Androutsopoulos, and N. Aletras, "Neural legal judgment prediction in english," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4317–4323. [Online]. Available: https://aclanthology.org/P19-1424

[29] I. Chalkidis, M. Fergadiotis, D. Tsarapatsanis, N. Aletras, I. Androutsopoulos, and P. Malakasiotis, "Paragraph-level rationale extraction through regularization: A case study on European court of human rights cases," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 226–241. [Online]. Available: https://aclanthology.org/2021.naacl-main.22

[30] B. van Dongen and F. Borchert, "BPI challenge 2018," Dataset, 2018. [Online]. Available: https://doi.org/10.4121/uuid:3301445f-95e8-4ff0-98a4-901f1f204972

[31] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkeHuCVFDr