VU VRIJE UNIVERSITEIT AMSTERDAM

Master Thesis

AuditRAG

An ARAG Approach for Auditing Dutch Annual Reports: Ensuring Compliance

by

Lars Bijvoet

2717625

Company Supervisor:	Anne Jonker
Supervisors:	Vincent François-Lavet
	Shujian Yu
Date:	02-06-2025
Credits & Period:	36, December 2024 - June 2025

Londen & Van Holland

AuditRAG

An ARAG Approach for Auditing Dutch Annual Reports: Ensuring Compliance

Lars Bijvoet
2717625
larsbijvoet@gmail.com

Master Thesis

Vrije Universiteit Amsterdam
Faculty of Science
Master Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

Host Organization:
Londen & Van Holland
Pedro de Medinalaan 39
1086 XP Amsterdam

June 2025

Preface

This thesis is written for the Master Project Business Analytics at the Vrije Universiteit Amsterdam. The purpose of this study is to investigate whether a locally deployed Agentic Retrieval-Augmented Generation (ARAG) system can assess compliance of Dutch annual financial reports.

This research was carried out in collaboration with Londen & Van Holland, a tax advisory and accounting firm in the Netherlands that offers a wide range of compliance and advisory services to companies and individuals.

First, I would like to express my sincere gratitude to my university supervisor, Vincent François-Lavet, for taking the time for our biweekly meetings, for the feedback he provided, and for the insights he offered that kept the project moving in the right direction.

A special thanks to Anne Jonker for being an outstanding daily supervisor. His support throughout this project has been invaluable. I am especially grateful for the willingness to help me, whether it was to discuss new insights, answer my questions, help me understand the company, or eliminate roadblocks so I could continue moving forward. The trust you showed in me, combined with your constant dedication, turned this project into a rewarding and enjoyable experience.

Finally, I would like to express my gratitude to the entire company Londen & Van Holland. I want to thank the company for not only supporting me with the research, but also for their warm welcome and making this experience joyful. The pleasant working atmosphere made it really nice to come to the office.

Abstract

This thesis examines the effectiveness of an Agentic Retrieval-Augmented Generation (ARAG) system to assess the compliance of Dutch annual reports with regulatory requirements. The study prioritises data privacy and security by focusing only on locally deployed solutions rather than cloud-based services like ChatGPT, because financial data is sensitive. The study creates a multi-agent framework that leverages Dutch regulatory documents (BW2T9 and the RJ) to automate compliance verification tasks. Using 100 selected compliance questions from the MKB checklist, four open-source language models of various sizes were evaluated. Using the Llama 3.3-70B-Instruct model resulted in the best performance (F_1-score of 0.74 and a precision of 89%), but smaller models also maintained high precision, offering viable options for organisations with limited computational resources. Human evaluation, by five professional auditors, validated the effectiveness of the system, achieving an average accuracy of 86% on an actual Dutch annual report. Ablation studies confirmed the essential contribution of each agent within the system, particularly the context evaluation agent and the retrieval expansion agent. The research provides empirical evidence on the viability of an on-premise ARAG system for financial compliance verification, offering audit firms a secure and efficient tool that respects the sensitive nature of financial documents. The GitHub repository can be found via: `https://github.com/Larsokillerz/AuditRAG`.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
RAG	Retrieval Augmented Generation.
ARAG	Agentic Retrieval-Augmented Generation
RJ	Richtlijnen voor jaarverslaggeving (Guidelines for annual reporting).
RJk	Richtlijnen voor de jaarverslaggeving voor micro en kleine rechtspersonen (Guidelines for annual reporting for small and micro companies).
BW2T9	Burgerlijk Wetboek 2 Titel 9 (Book 2 title 9 of the Dutch Civil Code).
SRA	Samenwerkende Registeraccountants en Accountants-Administratieconsulenten (Collaborating Registered Accountants and Accountants-Administrative Consultants).
MKB	Midden- en Kleinbedrijf (small and medium company).
IFRS	International Financial Reporting Standards.
GAAP	Generally Accepted Accounting Principles.
NLP	Natural Language Processing.
AI	Artificial Intelligence.
LLM	Large Language Model.
LVH	Londen & van Holland.
SBU	System Billing Units.
GPT	Generative Pre-training Transformer.
VM	Virtual Machine.

1

Introduction

1.1. Problem description

Londen & Van Holland is a tax advisory and accounting firm in the Netherlands, offering a wide range of compliance and advisory services to businesses and individuals. The firm specialises in financial auditing and tax compliance. They have a diverse clientele, including small and medium enterprises (SMEs), large corporations, and internationally active companies.

Ensuring compliance with Dutch accounting regulations is a complex and essential task for auditors at Londen & Van Holland. Annual reports must meet the requirements of standards such as the *Richtlijnen voor de jaarverslaggeving* (RJ) and *Book 2 Title 9 of the Dutch Civil Code* (BW2T9). To check whether an annual report adheres to these rules, Londen & Van Holland uses an MKB checklist created by the SRA. This checklist contains the requirements that must be in the annual report based on the RJ and BW2T9. Therefore, only adhering to the Dutch regulations and excluding IFRS and GAAP in this study. In addition, it specifies whether certain items need to be included, depending on the size of the company (small, medium, or large). Where the company size is determined based on Table 1.1. A company is classified into a specific category if it meets at least two of the applicable criteria for that category for two consecutive financial years.

Category	**Assets**	**Net Turnover**	**Number of Employees**
Micro	< €0.45	< €0.9	< 10
Small	€0.45 – €7,5	€0.9 – €15	10 ≤ 50
Medium	€7.5 – €25	€15 – €50	50 – 250
Large	> €25	> €50	> 250

Table 1.1: Overview of enterprise size categories in the Netherlands. Euros are in millions. Table from [1].

This process is often time-consuming, resource-intensive, and prone to human error, especially given the large volume of financial data and constantly changing regulations. Considering these challenges, Londen & Van Holland seek to explore whether a more automated solution can streamline compliance assessments and reduce the potential for human error.

Emerging technologies, particularly Large Language Models (LLMs), have the potential to significantly enhance the auditing of annual reports. LLMs excel in natural language processing, information extraction, and text analysis, enabling automated and systematic assessment of the extensive textual data contained in these reports. By integrating LLMs into the audit process, organisations can reduce the dependence on manual error-prone procedures, improving the efficiency, precision, and consistency of compliance evaluations.

However, using a public service such as ChatGPT is not a viable option due to the highly confidential and sensitive nature of financial data. All data must remain within the secure systems and infrastructure of Londen & Van Holland. To address the challenges that arise from the complexity of Dutch accounting regulations, the large volume of financial data, and the need to maintain stringent privacy standards, a locally deployed *Agentic Retrieval-Augmented Generation* (ARAG) system offers a promising solution.

1.1.1. Agentic Retrieval-Augmented Generation (ARAG)

Unlike standard LLMs, which solely rely on pre-trained parameters and knowledge, a Retrieval Augmented Generation (RAG) system, pioneered by Lewis et al. [2] and Guu et al. [3], dynamically fetches relevant information from external data sources (e.g., the RJ, BW2T9 and the MKB checklist) and incorporates these retrieved snippets into the prompt provided to the LLM. This mechanism ensures that responses are grounded in the most recent and context-specific regulatory texts, without requiring the model itself to be retrained every time new regulations or data become available.

An ARAG system extends this concept further by allowing the system to autonomously identify the required information and retrieve it from various knowledge bases. The system can examine its intermediate reasoning or results, assess what additional context might be necessary for a complete answer, and then actively query the relevant documents or databases, see Figure 1.1 for a general overview of an ARAG system. For example, suppose that an auditor asks how a specific part of BW2T9 applies to a particular section of a financial statement. In that case, the RAG agent can automatically locate the relevant regulatory text in BW2T9 and use that as context to generate a precise and explainable response. Using agent capabilities, the system can prioritise certain data sources, interpret multiple query results, or iterate retrieval prompts until sufficiently accurate or complete information is obtained [4].

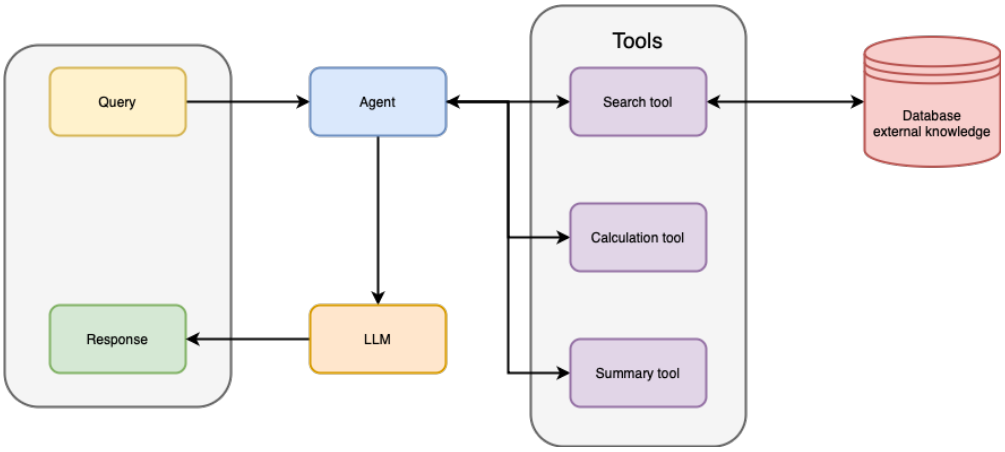


Figure 1.1: General overview of ARAG.

This approach ensures that compliance evaluations on annual reports are based on the most current and context-specific regulatory requirements, improving the precision, consistency, and explainability of automated audits without the need to retrain the LLM every time new data is available.

This research focuses on developing and evaluating a local ARAG system to assist accountants in automating the assessment of annual reports to ensure compliance with Dutch regulations. Using advanced retrieval techniques and language models, the aim is to improve the efficiency, accuracy, and interpretability of compliance verification in financial auditing.

1.2. Relevance of the problem

Accurately assessing financial compliance is crucial for businesses, auditors, and regulatory authorities. Companies must ensure that their annual reports comply with legal standards, while auditors are responsible for verifying this compliance. Mistakes or inconsistencies in this process can result in financial penalties, legal issues, and reputational harm.

An ARAG system could improve this process by providing insights while reducing the workload for auditors. By incorporating retrieval-based compliance verification, auditors can increase accuracy, reduce errors, and improve their decision-making.

From a scientific perspective, this research contributes to the field of applications of LLMs in Dutch financial auditing and regulatory compliance. Although LLMs are increasingly used in various domains, their potential for compliance assessment, particularly in the Dutch language, remains underexplored. By analysing the impact of different retrieval parameters and language models on compliance accuracy, this study aims to address this gap and provide practical recommendations for integrating LLMs into audit workflows by providing context-aware responses.

Furthermore, the findings of this research could drive future advancements in AI-assisted auditing, laying the groundwork for further innovation in automated compliance verification and the broader financial auditing sector.

1.3. Research question

Based on the description of the problem and its relevance, the following main research question can be stated:

> *How effectively can an Agentic Retrieval-Augmented Generation (ARAG) system, leveraging the Richtlijnen voor jaarverslaggeving, Burgerlijk Wetboek 2 Titel 9, and the Midden- en Kleinbedrijf Checklist, assess the compliance of Dutch annual financial reports, focusing on precision, interpretability, and suitability for auditing?*

To address this research question, this study is structured around the following sub-questions:

- What impact do different language models (e.g., Llama 3.1-8B-Instruct, Qwen3-8B, Qwen2.5-32B-Instruct, Llama 3.3-70B-Instruct) have on the precision and interpretability of compliance assessments?
- How does varying the number of retrieved documents (*top-k*) influence the system's ability to generate reliable compliance evaluations?
- How do different chunk sizes affect the performance of the ARAG system in retrieving and analysing regulatory content?
- What are the practical implications of implementing an ARAG system for financial auditors, and how can it be effectively integrated into existing audit workflows?

By examining these factors, this research aims to provide a clear understanding of how RAG-based LLMs can be used in compliance auditing and to offer practical insight for improving their use in regulatory assessments.

1.4. Research outline

The remainder of the research is outlined as follows: Chapter 2 reviews the existing literature on LLM-driven auditing, focusing on LLMs and RAG, and identifies the current gaps that motivate this study. Chapter 3 details the various data sources, including annual reports and Dutch regulatory documents, and describes the preprocessing steps undertaken to prepare them to be usable in the ARAG system. Chapter 4 presents the design of the ARAG system, explaining the selected retrieval techniques, the language models used, and the metrics used to measure performance in compliance auditing. Chapter 5 reports the experimental findings, comparing different configurations and evaluating their impact on precision and interpretability. Chapter 6 interprets these results in light of the research questions, considers their implications for audit practice, highlights the limitations of the study, and proposes directions for future work to extend or refine the system. Finally, Chapter 7 summarises key contributions and reflects on the effectiveness of RAG-based systems for regulatory compliance.

2

Literature Review

Compliance assessments in financial auditing have historically relied on a combination of rule-based systems and manual checks, often resulting in time-consuming and error-prone procedures. Recent advances in LLMs and RAG techniques have opened new opportunities to streamline these tasks [5]. This chapter reviews the existing literature on LLMs and RAG systems in the financial domain, examining their applications to tasks such as question answering and information retrieval in financial reports. The chapter also discusses recent developments in ARAG architectures and evaluates their potential applicability to financial auditing. Finally, it identifies the gaps in the current research that motivate the current study.

2.1. Evolution of RAG for Financial Auditing and Compliance

Recent advances in Natural Language Processing (NLP) have been driven by the rise of LLMs such as the GPT and LLama series, along with alternatives like Gemini [6–8]. As successful as these models are, there are still challenges they must face, such as outdated knowledge and a lack of domain-specific expertise [9, 10]. A significant problem with these LLMs is the generation of fabricated information, known as "hallucinations". This happens especially when the LLM must respond to a query that exceeds the scope of its training data [11].

RAG, introduced by Lewis et al. [2] and Guu et al. [3], offers a solution to these challenges. As mentioned in Section 1.1.1 within RAG, the LLM first queries an external data source to gather relevant information before generating an answer. Using this retrieval step, the generation phase uses the information extracted from the data source, improving the accuracy and relevance of the output, and thus reducing the so-called "hallucinations" [12, 13]. For this reason, the use of LLMs within an RAG system has made LLMs more suitable for practical applications, for example, financial auditing.

Recent research shows that RAG can improve the analysis of financial documents by grounding the LLM output in factual references. RAG systems have been applied to question-answering on financial reports to support decision making. For example, Iaroshev et al. [14] developed an RAG pipeline to answer queries about bank financial reports, finding that the system could provide accurate and contextually relevant answers by retrieving relevant report sections. This approach can be extended to compliance checks. The RAG system can indicate compliance or

deviations by retrieving relevant regulations (e.g., RJ guidelines or BW2T9) and comparing them with the report content.

2.2. Evaluation of different LLMs for Compliance Tasks

While RAG systems provide a robust framework for integrating external knowledge into LLM outputs, the overall performance of such systems is highly dependent on the underlying language model. Recent studies have compared state-of-the-art models on compliance and financial tasks, revealing significant differences in accuracy, interpretability, and domain-specific performance. For example, Zhao et al. [15] evaluated models such as GPT-4 and various LLaMA 2 configurations on financial question-answering benchmarks. Their results indicate that GPT-4 generally achieves higher factual accuracy and coherence, substantially outperforming smaller open models on domain-specific questions. The responses of GPT-4 were more accurate and relevant, even under imperfect retrieval conditions, while the LLaMA-2 models exhibited more variability and occasionally reproduced large text segments without fully addressing the query [15].

Emerging models such as Mistral 7B and Qwen2.5-32B [16, 17] are also promising, particularly in scenarios that demand multilingual support and efficient on-premise deployment. However, comprehensive evaluations of these models in the specific context of compliance verification and within an RAG system are still very limited. Domain-specific LLMs have shown clear advantages in financial contexts. Recent studies have highlighted the benefits of domain-specific fine-tuning in the financial sector. For example, Fatemi et al. [18] demonstrated that fine-tuning a smaller language model (e.g., Mistral-7B, Llama3-8B, and Phi3-mini) on different financial datasets improved its performance in financial text classification tasks.

2.3. Key Components of ARAG Systems

Although RAG has been shown to be effective, as discussed in Section 2.1, its implementation still requires some tuning. An RAG system has several components that influence the performance of the system. These include the retrieval model, which determines how relevant information is identified. The structure of the external knowledge base affects the efficiency with which the data is stored, indexed, and accessed. The complexity of the LLM shapes how responses are generated. Additional factors such as data preprocessing methods, vector encoding strategies, chunking approaches, and system orchestration also influence how effectively the various parts of the RAG pipeline function together [14].

According to Gao et al. [19], a traditional RAG system includes three fundamental steps: indexing, retrieval, and generation. During the indexing process, the raw data is converted into plain text, segmented into smaller chunks, and finally encoded as vector representations. These vectors, which are the numerical representation of the data, capture the semantic meaning and allow numerical operations. In the retrieval phase, we encode the query of the user with the same encoding as we did in the indexing phase. These query vectors are then compared against the chunk vectors to determine similarity scores. The chunks with the highest scores are retrieved and used to expand the context for the generation phase. Finally, in the generation phase, the original query and the selected chunks are combined into

a prompt for an LLM. This enables the LLM to generate contextually relevant and coherent responses [19].

Although non-agentic or naive RAG systems can retrieve relevant context, they typically follow a fixed retrieval generation pipeline without mechanisms for dynamic reasoning or iterative verification [4]. ARAG introduces an autonomous layer on top of the retrieval and generation components. These agents actively plan and execute the retrieval-generation workflow, rather than following a fixed pipeline. These agents can analyse the query, decide what knowledge sources or tools to consult, perform iterative retrievals, and even orchestrate multi-step reasoning before producing a final answer [4]. The iterative decision making and adaptive reasoning required for such agents align with the principles of Deep Reinforcement Learning (DRL). Here, agents learn optimal strategies through dynamic interactions with their environment. François-Lavet et al. [20] presents a framework for DRL, emphasising its ability to solve complex decision-making tasks by combining reinforcement learning with deep neural networks. These principles are closely related to the mechanisms that allow ARAG agents to adapt and make informed decisions.

Singh et al. [4] defines ARAG as the embedding of autonomous agents into the RAG pipeline. These agents can be deployed for a wide range of tasks, such as reflection, planning, tool utilisation, and multi-agent collaboration. This way they can dynamically manage retrieval strategies and refine the context for complex tasks [4]. In other words, the agent component oversees and optimises the interaction between retrieval and generation, making real-time decisions about when to retrieve, what to retrieve, and how to integrate retrieved information into the answer.

The architecture of the ARAG is fundamentally similar to that of traditional RAG. It still consists of indexing, retrieval, and generation, but introduces an agent controller that intervenes in the loop. This agent receives the user query and can perform actions such as: analysing the query intent, selecting among multiple knowledge sources, issuing search queries, calling external tools or APIs (for instance, a web search or calculator), and deciding when the gathered information is sufficient to answer the query. Crucially, the agent can operate in an iterative fashion: it may retrieve some documents, evaluate their relevance, then refine the query, or retrieve additional information if needed. This loop of retrieving, reading, and refining continues until the agent judges that it has enough evidence to generate a coherent and contextually relevant response [4]. The agent can also include self-reflection steps, such as criticising its initial response and refining the response based on the feedback or retrieving additional information [4].

2.4. Evaluation for RAG systems

When evaluating RAG systems, three important aspects must be taken into account: context relevance, answer faithfulness, and answer relevance [21, 22]. Context relevance refers to how well the retrieved information matches the user's question. It checks whether the system finds information that is truly useful for answering the query while avoiding unnecessary or unrelated content that could make the response less clear or harder to process. The faithfulness of the answers looks at whether the generated answer correctly reflects the retrieved information. It tries to ensure that the response of the model is based on evidence

and does not add incorrect details, misinterpret information, or contradict sources. Finally, answer relevance measures how well the generated answer responds to the original question. A relevant answer should not only be factually correct, but should also fully and clearly address what the user is asking, showing that the system understands the main purpose of the question [21, 22].

Different frameworks, such as Retrieval-Augmented Generation Assessment Suite (RAGAS) by Es et al. [22] and Automated RAG Evaluation System (ARES) by Saad-Falcon et al. [21] provides automated metrics for context relevance, answer relevance, and faithfulness, using an LLM to judge each aspect [23]. Such LLM-based evaluators (e.g., GPT-4 used as a judge) have been used to score how well responses are grounded in the provided documents and how directly they answer the query [23]. Other metrics include string overlap measures like Recall Orientated Understudy for Gisting Evaluation (ROUGE) or Bilingual Evaluation Understudy (BLEU), for answer quality, and recall at k or MRR (Mean Reciprocal Rank), for retrieval quality, when ground-truth answers or documents are available. Human evaluation continues to play a crucial role. Ultimately, many studies rely on human judgement to assess the correctness and faithfulness of responses, particularly in critical applications [23].

2.5. RAG systems within Finance

Question answering (QA) in financial reports has received significant research attention. Iaroshev et al. [14] built an RAG system to assist private investors in querying half-yearly and quarterly reports of banks. Their system retrieves relevant passages from the financial report and feeds them to GPT-4 to answer questions, aiming to improve decision-making for investors. They evaluated different versions of the pipeline and found that using high-quality retrieval and generation components is essential. The best setup used OpenAI's Ada model for embedding-based retrieval and GPT-4 for generation, resulting in the most accurate and relevant responses [14]. A weaker set-up, which used the smaller MiniLM embedder, scored significantly lower. The MiniLM embedder lacks the capacity to capture the nuanced information necessary for effective retrieval in complex financial documents. This resulted in lower "context relevance" and negatively affected both the "answer faithfulness" and the "answer relevance" [14].

This highlights that high-quality embedding models and language models improve performance [14]. They also observed that well-structured and coherent reports resulted in better results than poorly written ones. The system handled qualitative questions, such as requests for explanations or descriptions, better than quantitative ones [14]. Their evaluation explicitly measured the same three metrics mentioned earlier: the relevance of the retrieved context, the faithfulness of the response, and the relevance of the answer itself. This demonstrated the value of RAG in producing grounded answers from financial texts [14].

Another study by Kim et al. [24] focused on financial QA with RAG and optimised the retrieval stage for finance documents like SEC 10-K filings. They noted that the specialised language and numerical data in these reports can challenge standard retrieval methods. To overcome these issues, Kim et al. [24] introduced a three-phase pipeline: pre-retrieval, retrieval, and post-retrieval. In the pre-retrieval phase, the system applies several query and corpus preprocessing techniques. These include query expansion and the addition of metadata to

both queries and documents, thereby enriching the financial context and reducing ambiguity [24]. During the retrieval phase, the authors fine-tuned state-of-the-art embedding models by incorporating domain-specific knowledge. They further enhanced retrieval performance through a hybrid strategy that merges dense embeddings with traditional sparse retrieval signals. This fusion effectively overcomes the limitations of relying solely on one retrieval modality, ensuring that both the nuanced semantics and the exact terminology found in financial documents are accurately represented [24]. Finally, the post-retrieval phase involves refining the initially retrieved results. Techniques such as Direct Preference Optimisation (DPO), a form of reinforcement learning, are used to rerank and filter the candidate documents. This final selection step ensures that only the most relevant and contextually appropriate documents are used to generate the final answer [24]. This shows that domain-adapted retrieval and hybrid search are key to RAG success in finance.

Besides QA, RAG has also been explored for financial sentiment analysis. Zhang et al. [25] presented an RAG framework designed for financial sentiment analysis. This framework incorporates an instruction-tuned LLM module that leverages pre-trained LLMs and human-like text to guide the model's execution based on task descriptions and desired outputs, which are typically labelled by humans.

Singh et al. [4] describes an example in insurance claims processing: an ARAG system retrieves the details of a customer's policy and the relevant regulations. Then it compares them with an accident report to determine eligibility for claims. The system's planning agent decides which documents to fetch (policy, prior claims, regulatory clauses) and a decision agent ensures the final recommendation (approve or flag the claim) is consistent with all rules and evidence. This illustrates how multistep retrieval and reasoning can automate compliance verification tasks that traditionally require human experts. Although this specific example is in insurance, the same principles could apply to verifying financial reports against accounting standards.

2.6. Research Gap

While existing literature has shown the potential of RAG systems in financial applications, several critical gaps remain.

Although RAG systems have been successfully applied to tasks such as financial question answering and semantic analysis [14, 24, 25], these implementations are primarily based on static, one-shot RAG pipelines. They lack agentic capabilities such as iterative retrieval, dynamic reasoning, tool use, or self-correction. As highlighted in recent surveys [4], ARAG introduces important innovations for multistep and adaptive reasoning, but no published studies have yet explored their application to financial document auditing or compliance verification. Specifically, no studies have investigated the use of ARAG systems to verify compliance of financial statements, such as annual reports, with regulatory standards. Furthermore, no research to date has focused on Dutch financial reporting, including the application of Dutch accounting regulations like the RJ and BW2T9.

Most existing RAG research assumes access to large-scale cloud-based LLMs (e.g., GPT-4). However, due to privacy and confidentiality constraints in auditing environments, such

models are unsuitable for deployment. This creates a need for open source alternatives, such as Llama 3-8B, and Qwen2.5-32B. In addition, evaluating the performance of smaller models is important to explore options that offer lower deployment costs. Limited research has been conducted on how smaller models perform in financial compliance verification tasks within an ARAG framework. Understanding the capabilities and limitations of these models is critical for developing practical, secure, and effective compliance auditing solutions.

This study addresses these gaps by designing, implementing, and evaluating a local ARAG system for the compliance verification of Dutch annual reports. The study investigates the effectiveness of smaller language models in a privacy-preserving on-premise setting.

3

Data

Preparing and auditing annual reports in the Netherlands must adhere to strict standards, as defined in BW2T9 and the RJ. The MKB checklist, provided by the SRA, can be used as a guide to assess whether an annual report complies with these standards. This chapter presents a detailed overview of the available data and the pre-processing steps taken to prepare the data such that it can be used within the ARAG system.

3.1. The Available Data

The available data consist of BW2T9, the RJ, an MKB checklist, and two annual reports.

3.1.1. Burgelijk Wetboek 2 Titel 9 (BW2T9)

BW2T9 is part of the Dutch civil law and contains detailed legal requirements for the preparation of annual reports. This includes provisions on presentation, valuation, and disclosure of financial information, as well as specific rules applicable to different company sizes, as discussed in Section 1.1. The text is structured into sections and subsections, each addressing specific aspects of annual reporting.

3.1.2. Richtlijnen voor de Jaarverslaggeving (RJ)

The RJ provides practical guidelines and interpretations for applying the legal requirements of BW2T9. These guidelines are more detailed and offer additional explanations, including best practices and specific examples to ensure compliance. The RJ is organised into chapters, sections, and subsections. The RJ covers topics such as revenue recognition, asset valuation, and disclosure requirements for various financial items.

3.1.3. MKB Checklist

The MKB Checklist is a tool designed by the SRA to help companies and auditors verify compliance with reporting standards. Each sheet in the checklist represents a specific category of reporting requirements. Within each sheet, every row represents an individual checklist item. Table 3.1 illustrates the structure of the MKB checklist. The checklist includes a

description of the requirement (checklist item x), the conditions under which the item applies (based on the size of the company), and references to relevant legal provisions or guidelines.

Main subject	Groot (big)	Midden (medium)	Klein (small)	Bron (source)	Controle (Audit)	Gewijzigd in 2024 (changed in 2024)
Checklist item 1	x	x	x			
Checklist item 2	x	x	x	source		
Checklist item 3	x	x				x, changed

Table 3.1: Example of MKB checklist 2024 format.

3.1.4. Annual Reports

This research uses real annual reports. The specific reports analysed in this research belong to medium-sized companies, which means that the companies meet at least two of the three criteria outlined in Table 1.1 for a minimum of two consecutive financial years. Due to privacy reasons, the names of the companies and any identifying details, such as personal names, addresses, or numbers, will not be disclosed. The reports include a balance sheet, profit and loss account, explanatory notes, and a management report. These reports serve as the primary dataset for evaluating the ARAG system's ability to identify compliance with the standards set in BW2T9 and the RJ, guided by the MKB checklist.

3.2. Data Extraction and Structuring

This section describes the process used to extract and structure text from the provided files. The tools and methods used to convert the data from its original format into a structured form are described. The structured data is then ready for further analysis within the ARAG system.

3.2.1. Burgelijk Wetboek 2 Titel 9 (BW2T9)

BW2T9 is provided in PDF format [26]. As a legal document, its structure is crucial for proper interpretation. Therefore, it is necessary to extract the text while preserving the hierarchical organisation of the document. Now, since the content of a PDF is positioned in a non-deterministic manner, it can be hard to programmatically extract text, tables, or images in a way that retains the original meaning or structure.

Looking at BW2T9, one can conclude that there are no tables or images in this PDF. Also, since it is a legal document, there is a clear structure in the document. The document is organised into sections, called "Afdeling x," each representing a distinct part of the legislation. Within these sections are subsections that detail the specific laws, identified as "Artikel x." Using these keywords and their consistent use of bold formatting, the system was able to extract the text associated with each section.

To extract the content of BW2T9 while maintaining its legal structure, a parser was developed that processes the PDF page by page. The primary goal was to isolate each article ("Artikel x") and capture its associated text.

First, the entire PDF is read using the PyMuPDF library [27], which allows access to the raw text of each page. In cases where a page appeared empty, often due to scanned or image-based content, OCR (Optical Character Recognition) was applied using Tesseract [28] to extract visible text. This ensures that no content was lost during parsing.

Once the complete text was assembled, regular expressions were used to identify the boundaries of the article by matching patterns such as "Artikel 360". The full text was then split based on these article markers, enabling us to segment the document into meaningful parts. Each segment was structured into a dictionary with the number of the article, its corresponding content, and associated metadata, including the source of the document and the reference to the article. An example is shown in Appendix A in Figure A.1 and the code of the process is given in Figure A.4.

3.2.2. Richtlijnen voor de Jaarverslaggeving (RJ)

The RJ, like BW2T9, is also provided in PDF format [29], and its legal structure once again plays a crucial role. The extraction process for the RJ begins by analysing the font properties of the text, such as font size and boldness, to identify hierarchical levels. This ensures that chapters, sections, and subsections that must be extracted to maintain the hierarchical structure could be distinguished. Additionally, text elements at the bottom of each page with smaller font sizes are flagged as potential footers and excluded to ensure that only relevant content is preserved.

As the document is processed, a nested structure is built dynamically. When a new chapter is identified, the current chapter is saved, along with any sections and subsections it contains, and a new chapter element is created. The same logic applies to sections and subsections. This allows for a clean and organised representation of the original structure.

Between structural headers, the content is added to the appropriate part of the hierarchy. For example, text located after the title of a subsection is stored as part of the content of that subsection. This ensures that the original logical structure of the RJ is preserved and that each part of the text is associated with the correct heading.

At the end of the process, any remaining chapters, sections, or subsections are added to the final output. The result is a structured representation of the RJ, shown in Appendix A within Figure A.2, and the code of the process is given in Figure A.5.

3.2.3. MKB Checklist

The checklist, created by the SRA, is provided in Excel format and contains 38 sheets. For this thesis, our focus will be mainly on sheets 4 through 16, which are about the balance sheet of the report. The first three sheets are excluded as they contain the table of contents and the legend. Sheets beyond 16 are related to the profit and loss statement in the annual report, whereas we will mainly concentrate on the balance sheet section of the annual report in this thesis.

The checklist consists of several columns, see Table 3.1. The first column contains the checklist item itself, followed by three columns labelled "Groot", "Midden", and "Klein", indicating whether the item applies to large, medium, or small companies. The checklist also contains a

column called “Bron”, which shows the source of BW2T9 and/or RJ. The final two columns, “Controle” and “Gewijzigd in 2024”, contain minimal information and are therefore excluded from further analysis.

Compared to the RJ and BW2T9, the checklist is less clear. The checklist items, particularly the sentences in the first column, often lack sufficient context to be understood on their own. Additionally, some text is split across multiple rows, which means that these rows need to be merged to get the complete text.

To automatically process the entire checklist, the first three rows of each sheet were discarded. This is because these three rows contain metadata and are not part of the checklist itself. The remaining rows were analysed and structured, with the column headers dynamically assigned based on the content of the Excel sheet. The “Gewijzigd in 2024” column simply flags items that were updated in 2024 relative to 2023, as can be seen in Table 3.1. Since this is not relevant for this research, it is removed. Likewise, the “Controle” (Audit) column, used by auditors to track which items they’ve reviewed, is not relevant to this study and has also been dropped.

To normalise the checklist text, split rows were merged, and unnecessary white spaces or special characters were removed. The checklist entries were categorised according to the columns “Groot”, “Midden” and “Klein”. Rows without relevant entries in these columns were removed to maintain consistency in the dataset. Furthermore, the “Bron” column, which contains references to legal articles or accounting standards, was cleaned and expanded to replace abbreviations or shorthand references with their full versions. For example, references such as “BW 2:275,5” were rewritten to “BW 2 Artikel 275 lid 5” for better clarity. Similarly, the ranges of references like “RJ 212.301 t/m 212.305” were expanded to include all individual references within the range, ensuring completeness and eliminating ambiguity.

To enhance the text of the checklist, the main subject and its corresponding subheaders were extracted and used to create new sentences that provide additional context. In doing so, the checklist entries became clearer and more informative, improving their usability for the LLM.

The processed checklist was further enriched by integrating information from external sources such as the RJ and BW2T9. The previously extracted dictionaries of the RJ and BW2T9 were parsed and linked to the checklist entries using the “Bron” column. This mapping was achieved by matching legal references or accounting guidelines in the checklist with the corresponding content in external sources. For example, if a checklist entry referred to “RJ 275.505,” the relevant content of the RJ was retrieved and included in a new column called “Bron_text.”

The final output of this process is a well-structured dataset, saved as a dictionary. This dataset includes the cleaned checklist text, the associated context of the subheaders, and additional information from the RJ and BW2T9. See Figure A.3 in Appendix A for an example of the structured BW2T9.

3.2.4. Annual Reports

Annual reports are provided in PDF format. These reports typically contain numerous tables, such as the balance sheet, profit and loss account, and explanatory notes. However, these

tables are not consistently structured and often vary in formatting between reports and even within the same report. As a result, programmatic extraction of tabular data can be challenging. Figure 3.1 illustrates an example of a balance sheet that can be included in an annual report.

To address these challenges, PDF files were first converted to Markdown using Docling [30], an open source document conversion tool developed by IBM. Docling is a tool that helps turn documents like PDFs into a more usable format while keeping their original layout. It uses intelligent AI models to understand how the document is organised, for example, where the tables are or how the text is split into columns. When Docling converts a document, it creates a Markdown version that keeps the meaning and structure of the original PDF. This makes it easier to divide the content into sections and use it in other parts of the system, such as for searching or analysis.

ACTIVA

	31-12-2024 €	**31-12-2023** €
Vaste activa		
Immateriële vaste activa	xxxxx	xxxxx
Materiële vaste activa	xxxxx	xxxxx
Financiële vaste activa	xxxxx	xxxxx
Totaal vaste activa	xxxxx	xxxxx
Vlottende activa		
Voorraden	xxxxx	xxxxx
Vorderingen	xxxxx	xxxxx
Liquide middelen	xxxxx	xxxxx
Totaal vlottende activa	xxxxx	xxxxx
Totaal activa	**xxxxx**	**xxxxx**

PASSIVA

	31-12-2024 €	**31-12-2023** €
Eigen vermogen	xxxxx	xxxxx
Voorzieningen	xxxxx	xxxxx
Langlopende schulden	xxxxx	xxxxx
Kortlopende schulden	xxxxx	xxxxx
Totaal passiva	**xxxxx**	**xxxxx**

Figure 3.1: General balance sheet for 31 December 2024.

3.3. Resulting data

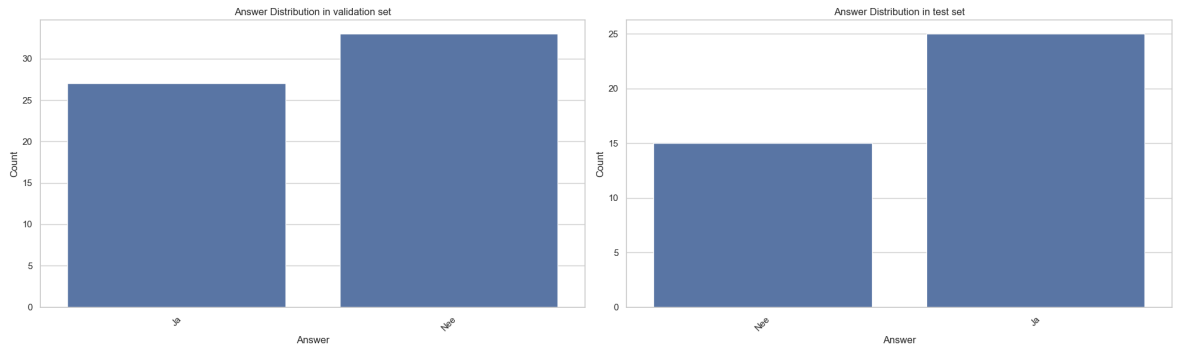
After extracting text from various sources (e.g., BW2T9, RJ, and annual reports), a normalisation step was applied to ensure consistency. This step involved standardising Unicode characters (e.g., replacing non-breaking spaces, accented quotes, and special punctuation), removing unnecessary whitespace, and correcting spacing around punctuation marks. All text was also converted to lowercase to reduce variation during text comparison. When tables were detected, they were left intact, while the rest of the content was flattened by replacing line breaks with spaces. This normalisation process ensured that content from different documents could be processed uniformly within the ARAG system.

After data processing, we are left with four structured data sources: BW2T9, the RJ, the MKB checklist, and the annual reports. Among these, annual reports are the primary documents that the ARAG system evaluates for compliance. The MKB checklist serves as the basis for this assessment. BW2T9 and the RJ are included as additional context sources to support the interpretation of legal and accounting terminology. From the checklist, 100 items were selected that are specifically relevant to medium-sized companies. These checklist items were manually reformulated into clear yes/no questions, with the help of compliance officers and accountants, to allow for binary classification.

3.3.1. Validation and Test Dataset

To test whether the ARAG system can check annual compliance reports, a validation and a test set are needed. The validation set is used to obtain the correct parameters for the system, and the test set will be used to get the final results of the models.

To obtain the validation and test set, a random 60/40 split was made on the 100 checklist items that were created. For all these 100 items, the correct answers were predefined. Figure 3.2 illustrates the answer distribution in both sets. In the validation set, 45% of the answers are "yes" and 55% are "no," whereas in the test set 62.5% are "yes" and 37.5% are "no."



(a) Distribution of validation set answers Yes/No (Ja/Nee). **(b)** Distribution of test set answers Yes/No (Ja/Nee).

Figure 3.2

An example of the structure of the validation and test set is shown in Figure 3.3.

```
{
    "text": question asked to the system,
    "expected_answer": yes or no,
    "category": Compliance (meaning we are in compliance mode)
}
```

Figure 3.3: Example of the structure validation and test set.

3.3.2. Final Dataset Overview

The result of all preprocessing steps is a clean and consistent dataset composed of three key components.

1. The normalised texts of BW2T9 and the RJ, structured by article and paragraph, respectively.
2. An annual report in Markdown format.
3. A validation and test set containing 60 and 40 pre-labelled checklist questions, respectively.

These components are used during evaluation and testing on the Snellius supercomputer in the Netherlands [31]. Given the constraints of the Snellius environment, particularly concerning persistent database services, we simplified the checklist to the format illustrated in Figure 3.3. This approach ensures compatibility with the available resources and avoids potential issues related to database management in a high-performance computing setting.

In a production environment, where such constraints are absent, the complete checklist as shown in Figure A.3 would be used. This version includes detailed references and metadata to facilitate more robust compliance checks and analyses.

The decision to use a simplified checklist on Snellius stems from the need to align with the system's operational parameters, ensuring efficient and effective use of computational resources during the evaluation phase.

4

Methodology

This chapter presents the methodology to systematically investigate how an Agentic Retrieval-Augmented Generation (ARAG) system can support compliance checking of Dutch annual reports.

4.1. Research Design Overview

This research follows a design science approach, building an ARAG system and evaluating it for compliance checking on Dutch annual reports. The focus is on developing a local AI-powered auditing assistant. This assistant, called Krissie, autonomously retrieves relevant regulatory information and sections of annual reports to answer compliance questions.

The overall research design covers two major phases:

1. **Development Phase:** constructing the ARAG system with multiple specialised agents and a knowledge base of Dutch regulations and annual reports.
2. **Evaluation Phase:** conducting experiments to asses its accuracy, interpretability, and efficiency.

To answer the research question, mentioned in Section 1.3, different configurations (e.g., various language models, document chunk sizes, number of retrieved documents) are compared. To evaluate the ARAG system, a baseline is also included. The baselines are the plain LLMs without any retrieval or agents, into which the entire annual report is fed directly. In this way, the ARAG system can be directly compared with the non-RAG system. The following sections outline the system architecture and components, data storage, model selection, experimental setup, and evaluation protocol to measure performance.

4.2. The ARAG System

The **Agentic Retrieval-Augmented Generation (ARAG)** system is a structured network of collaborating agents developed to assess compliance of Dutch annual financial reports. Beyond compliance verification, the system can answer user queries related to Dutch accounting by retrieving and synthesising information from BW2T9 and the RJ. Within the system, there are

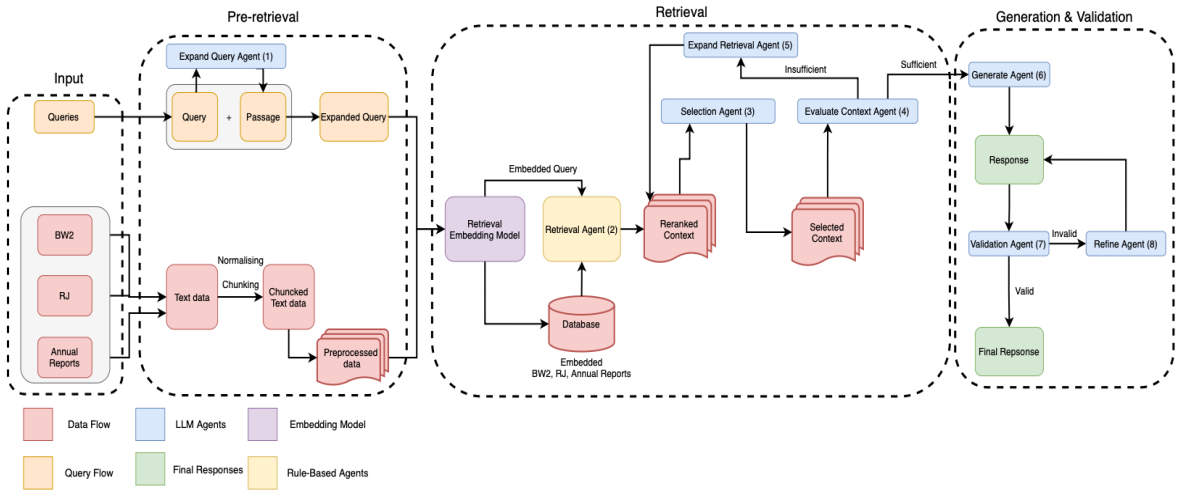


Figure 4.1: The ARAG Pipeline.

two types of agents: language models and rule-based agents. Together, the agents follow a clear step-by-step workflow to answer questions aimed at assessing whether an annual report is compliant. Figure 4.1 shows the complete ARAG pipeline, where the agents are numbered.

Each agent is responsible for a specific part of the workflow. One agent (3) selects the most relevant information from the retrieved data, while another (1) expands the query to improve clarity. Other agents (2, 5) handle information retrieval through vector search and traditional text-based search, and a different agent (4) evaluates the context it retrieved. Once the necessary context is collected, a different agent (6) generates an initial response, which is then reviewed and refined by the validation agents (7, 8).

To control the agents, `LangGraph` [32] is used, a framework for orchestrating multi-agent workflows in a graph-based structure. Each agent is represented as a node in the graph, and the edges define how data flows between them based on conditions or outcomes. This setup allows for clear control over the sequence of steps, branching logic, and collaboration between language models and rule-based components. LangGraph provides both flexibility and transparency, which are essential for structured reasoning tasks such as compliance checking in annual reports.

4.2.1. Agent Roles

Within the ARAG system, as shown in Figure 4.1, each numbered component corresponds to a distinct agent. These agents work together to process compliance questions about Dutch annual reports. The different agents used are described below, and the prompts used for the different agents can be found in Appendix B in Section B.1.

ExpandQueryAgent

This agent applies the Query2Doc method of Wang et al. [33]. In this approach, the agent is asked to respond to the query with a query-related passage, denoted p. This passage is intended to surface synonyms, related terms, and implicit context that may not appear in the raw query. Next, the passage is concatenated with the original user

query, denoted as q, to form an expanded query. The new expanded query q^+, is formed as follows:

$$q^+ = \text{concat}\left(\{q\} \times n, p\right),$$

where $n = 5$, as mentioned by Wang et al. [33]. The prompt used to generate p can be found in Appendix B.1.1

RetrievalAgent

This agent, which is a rule-based agent, implements a multistage retrieval pipeline to collect relevant documents for both compliance assessment and query answering. The agent performs four key functions:

1. **Initial retrieval**: Executes parallel searches using both dense and sparse methods across three distinct document collections: BW2T9, RJ guidelines, and annual report sections. For each collection, the agent retrieves the top 100 documents.
2. **Domain separation**: Maintains a clear separation between regulatory sources (BW2T9 and RJ) and the content of annual reports.
3. **Re-ranking**: Using a cross-encoder model, it performs a computationally more expensive but more accurate re-ranking of the initially retrieved documents. After re-ranking, it will return the top-k annual report sections and the top-k regulatory content.
4. **Adaptive expansion**: When directed by the `EvaluateContextAgent`, it can generate alternative query formulations and perform additional retrieval rounds to enhance recall.

The agent leverages both semantic similarity (through embeddings) and lexical matching (through BM25), which will be discussed in Section 4.5. These scores are then combined through a weighted sum as detailed in Section 4.5.3.

SelectionAgent

The idea of the selection agent is adopted from Kim et al. [24]. The selection agent filters the documents to those necessary for answering the query. It processes the top-k sections from the annual report and selects only those sections that are relevant for compliance verification. Similarly, from the top-k retrieved regulatory documents, the agent selects only the essential documents needed to support the compliance check and to formulate a correct answer to the query. The selection agent is an LLM-based agent that returns the indices of the selected context passages. The prompts used to select the essential content can be found in Appendix B.1.2.

EvaluateContextAgent

This agent determines whether the currently retrieved and selected documents provide sufficient context to answer the query. It evaluates both the coverage of all aspects of the query and the level of detail of the available context. If it determines that the context is insufficient, it indicates the need for expanded retrieval. The prompt used can be found in Appendix B.1.3.

ExpandRetrievalAgent

When the `EvaluateContextAgent` determines that more context is needed, this agent executes an expanded retrieval strategy. The agent generates alternative formulations

of the original query to capture different semantic aspects and then performs additional retrieval rounds, using these alternative queries, by calling the `RetrievalAgent`. Finally, the agent combines and deduplicates the newly retrieved documents with the original set, ensuring a more comprehensive context while avoiding redundancy. The prompt used can be found in Appendix B.1.4.

GenerateAgent

This agent processes the selected documents and formulates a coherent, accurate response to the user's query. For compliance queries, it explicitly states whether the annual report meets the relevant requirements (with a clear conclusion: yes or conclusion: no), provides a concise explanation of its reasoning and cites specific passages from the annual report that support its conclusion. The agent follows a consistent response structure to ensure clarity and auditability, making it clearer for accountants to verify the assessments. The structure used in the response is as follows:

```
Conclusion: Yes or Conclusion: No
Explanation: [Maximum 4 sentences why this is your conclusion]
Justification from annual accounts: [exact passages from ANNUAL
REPORT SECTIONS, if it is not in the annual report sections then
say 'It is not in the annual report']
```

To ensure that the agent responds in the correct format, a set of examples is provided in the prompt. This helps the agent to follow the desired structure consistently. The entire prompt can be found in Appendix B.1.5.

ValidationAgent

The following two agents are inspired by the self-refine method of Madaan et al. [34]. This agent acts as an internal critic that judges the response on the following:

- Factual accuracy: Ensure that all statements are supported by the retrieved documents.
- Completeness: Verify that all aspects of the query are addressed.
- Direct Answering: Confirming that a clear conclusion is provided.
- Consistency: Verify that the explanation aligns with the stated conclusion.

If any issues are detected, the agent produces specific feedback for refinement, identifying exactly what needs to be corrected or improved. The prompt used for this agent can be found in Appendix B.1.6

RefinementAgent

To complete the self-refine loop mentioned in Madaan et al. [34], the `RefinementAgent` ingests the feedback from the `ValidationAgent` and rewrites the response. It focuses on addressing the specific issues identified, whether they involve factual corrections, adding missing information, clarifying reasoning, or restructuring the response format. The refined response is then sent back to the `ValidationAgent` for re-evaluation. This iterative refinement process continues until the response meets the quality standards or the maximum number of refinement attempts is reached. The prompt can be found in Appendix B.1.7.

4.2.2. Architecture

The ARAG system supports two operating modes, each designed for different use cases. In **single-query mode**, the pipeline is run end-to-end on one query, with every intermediate result stored in a single `AgentState` object. In **batch mode**, multiple queries are processed in parallel through a shared `BatchState` object. This mode enables vectorised LLM calls, achieving approximately a 7× speed-up on the Snellius supercomputer. The architecture of both modes is implemented using the `LangGraph` framework.

Single Mode
In single-query mode, the ARAG system creates a single `AgentState` object that carries all intermediate data. This object keeps track of the query, retrieved documents, selected passages, LLM responses, and validation flags throughout each step of the LangGraph as presented in Figure 4.2. Within this workflow, every node in the graph represents an agent, and each arrow indicates the flow of control or data. Solid arrows denote the normal progression, while dashed arrows indicate conditional branching.

One can see from Figure 4.2 that the user's query enters the system at the `START` node and flows through the sequence of agent nodes, as described in Section 4.2.1. First, the `ExpandQueryAgent` enriches the query by generating related passages that capture synonyms and implicit context. Based on this enriched query, the `RetrievalAgent` fetches the relevant information. The `SelectionAgent` then filters the information to the most relevant information that is used to answer the query. Next, the `EvaluateContextAgent` inspects whether the accumulated context fully covers the question. If `EvaluateContextAgent` finds the context insufficient, the `ExpandRetrievalAgent` will expand the search for documents. Once the context is deemed sufficient, the `GenerateAgent` produces a first response. This response is passed to the `ValidationAgent`, which confirms the factual basis, complete coverage, and clarity of conclusions. If any validation criteria fail, the `RefinementAgent` applies targeted corrections.

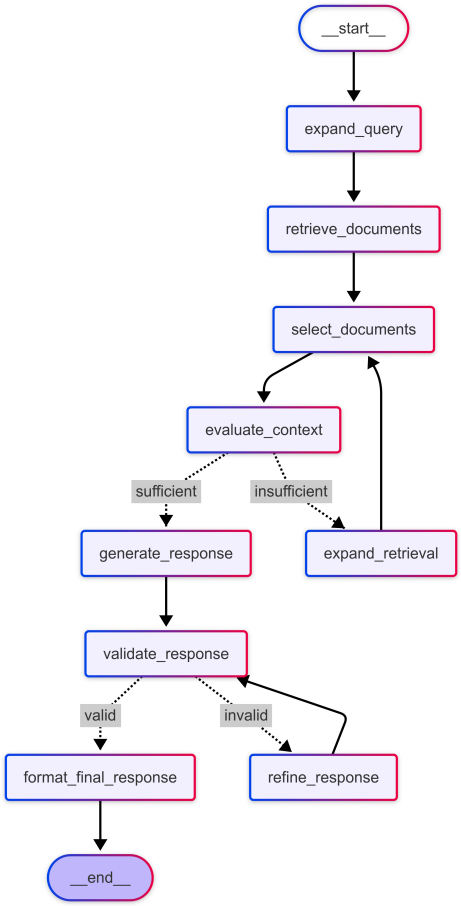


Figure 4.2: The single-mode ARAG system.

Batch Mode

In real-world auditing scenarios, compliance verification often requires checking dozens or hundreds of requirements against an annual report. Processing these sequentially would be time-consuming and inefficient. To address this issue, batch mode is introduced.

In batch mode, the ARAG system processes multiple queries in parallel by passing a single `BatchState` through a LangGraph whose nodes are vectorised (`_batch`) versions of the single-query agents. Unlike the simpler flow shown in Figure 4.2, the batch mode in Figure 4.3 contains additional routing nodes that manage the flow of multiple queries simultaneously. The routing nodes are "update_state_for_mixed_context", "split_and_process_mixed_context", "update_validation_state", and "split_and_process_mixed_validation". These nodes serve an important function: they partition queries based on their current processing status and direct them to their appropriate agents.

For example, when the `EvaluateContextAgent` determines that some queries have sufficient context, while others require additional retrieval, the agent splits the batch (e.g., mixed state). Queries with insufficient context will need more information and are directed to the `ExpandRetrievalAgent`, while the queries that are sufficient will wait un-

til these queries are sufficient as well. The `ExpandRetrievalAgent` is called within the "split_and_process_mixed_context" node. If all contexts of all queries are insufficient, they will be directed to the "expand_retrieval_batch" node. The same happens with the validation agents.

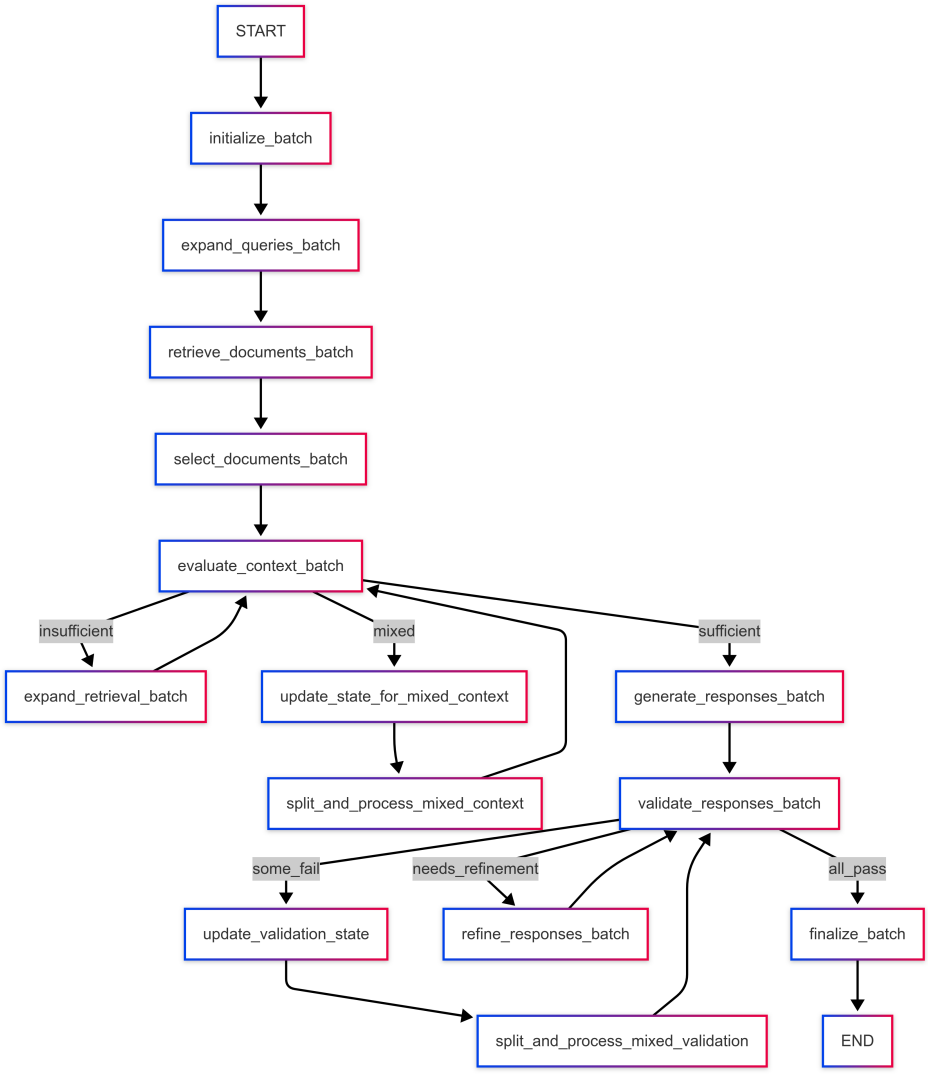


Figure 4.3: The batch mode ARAG system.

The batch mode allows for processing queries in groups based on their current state. Efficiency is improved through vectorised LLM calls. The batch mode uses the batch processing capabilities of modern LLMs to run multiple prompts in a single forward pass, reducing GPU idle time and context switching. On the Snellius supercomputer, the batch mode yields an approximately 7× speed-up improvement over the execution of a single query.

This performance improvement makes ARAG practical for real-world audit firms that need to perform comprehensive compliance checks on multiple annual reports with tight deadlines.

Batch capabilities also enable the overnight processing of large compliance checklists, with results ready for auditor review the next morning.

4.3. Chunking

Breaking documents down into smaller parts, a process known as chunking, is essential for improving retrieval accuracy and avoiding length-related issues in large language models. Chunking can be performed at different levels of detail, including tokens, individual sentences, or based on semantic meaning [35].

Token-level chunking is the most straightforward approach. The text is cut after a fixed number of tokens without taking into account the sentence boundaries. While this approach is easy to implement, this often splits sentences in unnatural places and therefore it may affect retrieval quality.

Sentence-level chunking maintains the integrity of sentence boundaries. Adjacent sentences are combined until a target length is reached. In this way, semantic continuity is preserved and massive context windows are avoided. Because it preserves meaning and it remains computationally efficient, sentence-level chunking offers a good trade-off for many RAG pipelines.

Semantic-level chunking goes further by using an LLM or another semantic parser to locate breakpoints. Using this technique, the context is preserved and the resulting chunks are highly coherent. However, this method is considerably more time-consuming and requires additional computation during preprocessing.

In this research, a sentence-level chunking strategy is used, as it balances simplicity and preserves semantic continuity.

4.3.1. Chunk Size

The length of a text chunk is a critical hyperparameter in RAG pipelines. If the chunks are too long, the individual chunks may exceed the context window of the LLM and increase the processing time. If the chunks are too small, the index will grow rapidly, and many chunks will contain insufficient evidence to support a faithful answer. An effective chunk must be large enough to preserve semantic coherence, yet small enough to fit within the context window of the model and keep the retrieval index manageable.

To determine the chunk size S that balances context preservation and index efficiency, S was treated as a tunable hyperparameter and optimised using Bayesian optimisation [36]. In this hyperparameter search, different chunk sizes were evaluated to maximise the F_1-score. For each candidate chunk size, the overlap of the chunk was fixed at $0.2S$, ensuring a consistent proportion of shared context between consecutive chunks. Because the splitter is aware of the sentences, every boundary ends in a complete sentence. The 20% overlap ensures that the adjacent sentences are duplicated, ensuring that the meaning at the boundaries remains visible in both chunks.

4.3.2. Chunking Technique

Based on the findings of Kim et al. [24], this study uses a sliding-window technique. Each document is divided into sequential chunks that do not exceed the chunk size. Successive chunks are produced by shifting the window according to:

$$\textit{chunk\_size} - \textit{overlap},$$

so every new segment repeats the final 20% of the preceding chunk.

In addition, the chunking process is made sentence-aware so that chunk boundaries align with natural language sentence boundaries whenever possible. Rather than strictly splitting at a fixed character or token count, the algorithm checks if the end of a chunk falls in the middle of a sentence. If so, the boundary is adjusted to include the full sentence in the current chunk (if the sentence is very long, the chunk may end just before that sentence to keep within the size limit). In practice, this means that each chunk contains only whole sentences, leading to semantically coherent segments of text.

4.3.3. Embedding Model Selection

After the documents are chunked, each chunk is converted into a vector representation using an embedding model. For this purpose, the **BAAI/bge-m3** model [37], a state-of-the-art multilingual embedding model, was chosen. The choice was guided by the strong performance of the model on the Massive Multilingual Text Embedding Benchmark (MMTEB) [38, 39], a comprehensive evaluation suite for embedding models across various tasks and languages. In particular, bge-m3 has been reported to achieve top-tier results on the MMTEB leaderboard for multilingual tasks, outperforming other open-source embedding models in various languages, including Dutch. The MMTEB evaluation spans dozens of datasets in nine task categories and more than 250 languages, providing evidence that bge-m3 can effectively capture semantic nuances in Dutch text.

Bge-m3 is suitable for several reasons. First, it is a truly multilingual model (supporting more than 100 languages), which means that it can embed Dutch sentences with the same encoder used for other languages, benefiting from cross-lingual learnt representations. This can especially be useful when auditing annual reports from companies from different countries. Second, it is designed for multiple retrieval functionalities (dense retrieval and even hybrid sparse signals) and can handle long input texts (up to 8192 tokens), aligning with the need to encode lengthy regulatory sections. Third, the model is open source and contains only 568 million parameters. Thus, it is small enough to run on a single high-memory GPU or even modern CPU servers, keeping deployment costs modest and allowing full on-premise processing of confidential financial data.

4.4. Database

Using the pre-processed material from Chapter 3, the data is ingested into two distinct database instances. All regulatory texts, BW2T9 and the RJ, are stored in the first instance, whereas the annual report corpus is stored in the second. The separate database instances allow for independent retrieval. The system can return the top-k regulatory passages and the

top-k report sections for any checklist query. This ensures that evidence from legislation and evidence from the target document are handled in parallel but remain logically isolated.

Each text chunk is converted into a numerical vector representation (embedding) using the bge-m3 embedding model. The resulting vectors, together with their metadata, are then ingested into the appropriate database instance.

4.4.1. Vector Store on HPC (FAISS) for Validation and Testing

For the validation and testing phase, the Facebook AI Similarity Search (FAISS) [40] library is used. FAISS is a specialised vector database library designed for efficient similarity search over high-dimensional vectors. It provides algorithms for indexing and searching large collections of vectors (even those that may not fit entirely in RAM) with support for GPU acceleration. This makes FAISS well-suited for a high-performance computing (HPC) environment. In the validation and testing phase, the ARAG system is deployed on the Snellius supercomputer in the Netherlands [31]. Given the constraints of the HPC setting, notably the lack of persistent database services on compute nodes, the use of FAISS allowed fast semantic searches without requiring a running database server. Instead, the index stays in memory during the job execution, which aligns with batch-oriented supercomputer usage.

The MKB checklist data are loaded from a simple JSON file, mentioned in Section 3.3.1 and presented in Figure 3.3, avoiding the need for networked database connections.

4.4.2. Production Database Integration (PostgreSQL)

The ARAG system being developed in this study is intended to be deployed on-premises to help accountants audit annual reports. Therefore, a production environment is needed. For production deployment, data is stored in a traditional relational database using PostgreSQL [41]. PostgreSQL is chosen because it is a powerful and scalable relational database management system. Also, it supports JSONB fields, full-text search, and vector storage through the 'pgvector' extension. Furthermore, PostgreSQL is open source, widely used, and community-driven.

In the production database, unlike the simplified HPC setup, each checklist item entry can include full metadata and reference links to the regulations. In production, the checklist items, unlike the simplified HPC setup, are stored as shown in Figure A.3 in Appendix A.

4.5. Retrieval Methods

A key component of the ARAG system is its retrieval module, which fetches relevant information from the prepared knowledge base to support compliance checking queries. The knowledge base consists of chunked textual segments from Dutch accounting regulations (BW2T9 and RJ guidelines) as well as Dutch annual reports. Given a compliance question, the system must identify both the applicable sections of the annual report that need to be examined and the corresponding regulatory text that provides context for assessing compliance. To accomplish this, we employ a hybrid retrieval approach that combines sparse retrieval (BM25) with dense retrieval based on original text embeddings. The motivation for using this approach is to take advantage of the strengths of both retrieval strategies. Dense

retrieval excels at capturing semantic similarities, which is useful when the query is phrased differently from the text. Sparse retrieval is effective for exact keyword matching, ensuring that precise terms are not missed. According to Wang et al. [35], such hybrid methods effectively balance the precision of lexical matching with the generalisation capabilities of dense representations, achieving strong retrieval performance with low latency. This makes it a suitable choice for our compliance checking ARAG system. The number of documents retrieved, denoted as top-k, is a tunable hyperparameter. The top-k determines how many candidate text segments are returned for further processing. In the following subsections, a detailed theoretical foundation and implementation of the retrieval methods are given.

4.5.1. Embedding-Based Retrieval (FAISS)

Embedding-based retrieval, often referred to as dense retrieval, uses vector representations of text to capture semantic meaning beyond exact keyword matches. The ARAG system employs this technique to find text segments that are semantically relevant to a given query, even if they do not share the exact wording. For example, the words "salaries" and "wages" mean semantically almost the same, but the writing is different. Theoretically, this approach is grounded in distributional semantics, the idea is that the meaning of text can be represented in a continuous vector space such that semantically similar texts have vectors that are close together (highly correlated).

Each text chunk d is encoded into a d-dimensional embedding vector $\mathbf{v}_d \in \mathbb{R}^d$ using the `BAAI/bge-m3` model (see Section 4.3.3). Given a query q, the same encoder produces $\mathbf{v}_q \in \mathbb{R}^q$. The relevance of each chunk d to the query q is scored with *cosine similarity*:

$$\text{sim}_{\cos}(q, d) = \frac{v_q \cdot v_d}{||v_q|| \cdot ||v_d||}, \tag{4.1}$$

where $||\cdot||$ is the ℓ^2-norm, with $||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ and $x \in \mathbb{R}^n$. Because all embeddings are ℓ^2 normalised, during indexing, Equation (4.1) reduces to the inner product, making lookups efficient in FAISS [40]. For each query, the system retrieves the top-k chunks with the highest cosine scores from the regulation index and the annual report index, respectively.

4.5.2. Sparse Retrieval (BM25)

In addition to dense retrieval, the system also uses a sparse retrieval strategy based on the BM25 algorithm, which is a classical information retrieval method for lexical matching [42]. The motivation for including BM25 is to capture cases where exact or near-exact keyword overlap is crucial. Compliance questions often contain specific terminology (e.g., financial terms or legal references) that, if present in a document, strongly indicate relevance. Sparse retrieval treats documents as bags-of-words and ranks them according to how well they textually match the query terms, without embedding them in a semantic vector space.

Let $f(t, d)$ be the frequency of term t in document d, $|d|$ the length of d, and $\bar{l}$ the average length of the document in the collection. With free parameters k_1 (term-frequency saturation)

and b (length normalisation), the BM25 score of a document d for query q is

$$\text{BM25}(q,d) = \sum_{t \in q} \text{IDF}(t) \cdot \left(\frac{f(t,d) \cdot (k_1 + 1)}{f(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\bar{l}}\right)} \right). \tag{4.2}$$

Here $\text{IDF}(t) = \ln\left(\frac{N - n_t + 0.5}{n_t + 0.5} + 1\right)$, where N is the number of documents and n_t is the number that contains the term t. In this study, the library named rank-bm25 is used [43]. This library uses $k_1 = 1.5$ and $b = 0.75$ as default values, and these values do not change in this study.

4.5.3. Hybrid Retrieval

To take advantage of both dense and sparse retrieval, the ARAG system uses a hybrid strategy. The hybrid score is the convex combination:

$$S_{\text{hyb}}(q,d) = \lambda \text{sim}_{\cos}(q,d) + (1 - \lambda)\text{BM25}(q,d), \tag{4.3}$$

where $\lambda \in [0,1]$. In this study, λ is set to 0.5, giving the dense and sparse scores equal influence in the final hybrid ranking. Kim et al. [24] investigated various values of λ across multiple financial datasets. The mean value of λ over all these datasets tends to be close to 0.5. Given that the primary focus of this study lies in evaluating model choice, chunk size, and the number of documents retrieved, rather than fine-tuning fusion parameters, $\lambda = 0.5$ is used as a balanced default.

Retrieval workflow
The retrieval workflow from the ARAG system works as follows:

1. For every query, retrieve the top-$R = 100$ chunks from the dense index and the top-$R = 100$ from the BM25 index for each knowledge base instance (regulatory and annual report).
2. Apply normalisation and compute the hybrid score using Equation (4.3). The chunks that appear in only one list receive the other score as 0. For example, if a chunk is retrieved by dense retrieval and is not found in the retrieved documents retrieved by BM25, then the score is manually set to 0 as the BM25 score.
3. Keep the best H candidates per index, where $H = 100$. These are forwarded to the cross-encoder described in Section 4.5.4. In practice, there are often fewer annual report sections than $H = 100$, which means that most of the time, all annual report sections are retrieved and passed to the cross-encoder. The choice for $H = 100$ is motivated by the computational cost of cross-encoders, which scale poorly with larger inputs.
4. The cross-encoder returns, after re-ranking, the top-k regulatory documents and the top-k annual report sections.

4.5.4. Cross-Encoder

To re-rank the retrieved documents obtained by the hybrid approach, a cross-encoder is used. In a bi-encoder, the query and the document are embedded independently. In contrast, a

cross-encoder concatenates them as ⟨[CLS] q [SEP] d [SEP]⟩ and feeds this single sequence through a transformer. This enables self-attention layers to compute pairwise interactions between all token positions across both the query and the document.

Using the **BAAI/bge-reranker-v2-m3** [37, 44] model, the 100 candidates retrieved with the hybrid retriever are re-ranked. The model is a multilingual cross-encoder fine-tuned for re-ranking. The bge-reranker-v2-m3 is preferred for three reasons. First, it shares the same bge-m3 backbone chosen for dense retrieval in Section 4.3.3, ensuring identical tokenisation and eliminating encoder mismatch effects. Second, bge-reranker-v2-m3 is trained in more than 100 languages, an important property for audits that must also process English or multilingual disclosures in Dutch annual reports. Third, the model footprint of 568 million parameters fits on a single A100 GPU or an 8-bit-quantised CPU server, facilitating a secure on-premises deployment. In practice, re-ranking the 100 candidates adds under 60 milliseconds per query, a negligible latency increase relative to the overall pipeline.

4.6. Language Models

A critical choice is to select the Large Language Models (LLMs) to power the various agents. For example, the `GenerateAgent` is responsible for the formulation of answers, while other agents handle tasks such as query expansion and validation. Since all data must remain on a secure infrastructure, the focus in this research is on open-source LLMs that could be locally deployed. Given the requirement for the Dutch language, models with strong multilingual or specifically Dutch capabilities are prioritised. In this research, the following models are considered: Llama-3.1-8B-Instruct, Llama-3.3-70B-Instruct [45], Qwen2.5-32B-Instruct [17, 46], and Qwen3-8B [47]. Each model was adapted to the financial domain through prompt engineering rather than further fine-tuning, since rules around auditing and annual reports change yearly.

For all models, the same prompts and agent frameworks are used. This allows for a direct comparison between the different models in terms of performance. Also, the `ValidationAgent` and `RefinementAgent` use the same LLM (the system does not rely on an external model for validation, instead, it uses the primary model itself to self-critique). This is a conscious decision to keep the system self-contained. One of the parameters of the models is called the temperature. The temperature is a parameter that controls the randomness of a model's output, where lower values make responses more focused and deterministic, while higher values encourage more diverse and creative generation. The temperature of all models is set to 0 to favour consistent deterministic outputs over creativity, since factual accuracy is preferred in auditing.

Model characteristics and rationale
Llama-3.1-8B-Instruct (8B parameters) is the "lightweight" member of Meta's third-generation family, offering a multilingual context window of 128k tokens and competitive precision for mixed tasks, while remaining deployable on a single high-memory GPU (e.g., around 21GB of RAM)[45].

Llama-3.3-70B-Instruct expands the capacity to 70B parameters. This model also has a context window of 128k tokens. The Llama 3.3 model, fine-tuned for instructions and text-based

tasks, is specifically designed for multilingual conversations. According to Meta, it performs better than many other open-source and proprietary chat models on widely used industry benchmarks [45]. The model needs around 140GB of RAM with its 70B parameters, making it the largest model used in this research due to resource constraints.

Qwen2.5-32B-Instruct delivers strong multilingual performance (over 29 languages) and also a 128k context window, making it well suited to long Dutch annual reports. The benchmarks report state-of-the-art scores on MMLU (83.9), GSM8K (95.9), and HumanEval (88.4) [17]. The 32B size, resulting in around 64GB of RAM, still fits on a single H100 GPU and even on an A100 GPU when quantised in 8-bit.

Qwen3-8B is a state-of-the-art multilingual model that supports more than 100 languages. With the Qwen3 series, a hybrid reasoning engine is introduced with two runtime modes. In thinking mode, the model generates an internal chain of thought encapsulated in `<think>` tags before emitting the final answer, leading to markedly higher scores on maths, coding, and common sense tasks than the Qwen2.5 and previous QwQ models of similar size [47]. Switching to non-thinking mode results in a small decrease in accuracy for faster responses. To make sure that the model can be directly compared with the other models used, the thinking mode is disabled. With 8B parameters and a context window of 32k tokens, it remains easily deployable on commodity hardware.

4.7. Experimental Setup

The experiments were conducted on the Dutch National Supercomputer Snellius, leveraging its high-performance multi-GPU infrastructure to run the ARAG system in batch mode [31]. In this research, an ARAG pipeline is developed using LangGraph [32], as discussed in Section 4.2. The batch mode allows for parallel processing of multiple queries through a shared state. Using batch mode provides significant speed-ups (approximately 7× on Snellius, based on local experiments) compared to single-query mode. The single-query mode was used only during early development and debugging. All reported results use the batch pipeline for efficiency and consistency.

For model tuning, a validation set of 60 compliance questions is used, as discussed in Section 3.3.1, targeting Dutch annual report requirements for medium-sized companies. All these questions were created and checked by auditors, and each question has a known ground-truth answer (yes or no). This validation set represents a realistic mix of compliance checks under BW2T9 and the relevant RJ guidelines for medium enterprises. The goal of the validation phase is to use these known answer queries to optimise two critical retrieval parameters for each language model, namely, the chunk size S and the number of retrieved chunks (top-k).

4.7.1. Hyperparamter Tuning

The hyperparameter tuning is performed as a Bayesian optimisation [36] problem to efficiently search for the optimal chunk size and the top-k values. Specifically, Scikit-Optimise's [48] gp_minimize function (Gaussian process-based optimisation) is used to explore the two-dimensional hyperparameter space. The search space was defined as:

- The number of chunks retrieved from the database for each query, top-$k \in [5, 25]$.
- The chunk size $S \in [128, 1024]$.

Each candidate combination uses a chunk overlap of $0.2S$, as stated in Section 4.3.1. The Bayesian optimiser was configured to run for 35 calls. The optimiser was set to start with eight random trials to sample the space broadly, followed by 27 sequential iterations. The F_1-score was set to be the objective to maximise, as the F_1-score provides a balanced measure of accuracy considering both precision and recall. An important note is that precision is especially critical in this domain. A false positive (the system incorrectly marking a non-compliant item as compliant) is more problematic than a false negative (the system missing a compliant item). In practice, an auditor can catch a false negative upon review, but a false positive could lead to an oversight of an actual compliance issue. Using the F_1-score as an objective, the system tries to maintain high precision while also not sacrificing recall unduly. As a result of this validation phase, each model was paired with its optimal chunk size and top-k.

The hyperparameter optimisation process is repeated for every candidate language model evaluated in this study. In this research, four different LLMs of varying sizes and capabilities are examined, which are discussed in Section 4.6. Each model was integrated into the ARAG pipeline and underwent its own 35-trial Bayesian optimisation on the same 60-question validation set. The result is an optimal configuration per model. For example, model A might perform best with smaller chunks and higher top-k, whereas model B might need larger chunks but fewer of them. All experiments leveraged the computational resources of Snellius (multiple H100 GPUs and high-memory nodes) to handle the large models and the parallel query processing efficiently.

4.7.2. Testing

After tuning, the main evaluation is carried out on a held-out set of 40 questions, with known answers, to measure each model's performance on unseen queries. These 40 test questions were curated similarly by domain experts, but were not used during the tuning phase, see Section 3.3.1. This provides an unbiased assessment of how well the ARAG system responds to new compliance checks. Each model, with its optimised chunk size and top-k value based on the results of the validation tuning, runs on all 40 test queries in batch mode, on Snellius.

4.7.3. Baseline: Non-RAG Setup

To determine the effectiveness of the ARAG pipeline, a non-RAG baseline is included. This baseline setup uses the same underlying language models but removes the retrieval and agentic reasoning components. Instead, the model receives the entire annual report (or as much as can fit within its context window) directly in its prompt, along with the compliance question. This setup isolates the contribution of retrieval and step-by-step reasoning from the raw language modelling capacity of the LLM.

All evaluations are conducted on the same 40-question test set and use the same evaluation metrics as for ARAG. Comparing the ARAG system with the non-RAG baseline reveals whether retrieval and agent-based validation offer added value.

4.7.4. Human Evaluation of ARAG System

To assess real-world applicability, the best-performing model is selected and deployed on a second annual report for human evaluation. In this human study, the same 100 questions (test and validation set) will be used, but on a different annual report and without predefined answers. Instead, company auditors review the model responses to judge the correctness. This mimics a realistic use case: the ARAG system analysing a fresh annual report and an expert verifying if its justification and conclusions are satisfactory.

During the case study, an annual report from a medium-sized Dutch company is used. For privacy reasons, the company name and sector are not disclosed and will be referred to as "Company X". In contrast to the validation and test environments, where predefined ground-truth answers were available, the case study simulates a real-world scenario in which professional auditors assess the output of the ARAG system. The annual report in this case is not prepared by the company Londen & Van Holland (LVH), but by a client. This means that this report has a different style and layout from the ones LVH creates. This distinction is important, as it tests the system's ability to process and analyse financial documents with varying formats and presentation styles. Furthermore, this also reduces any bias towards the quality of the report, resulting in a more real-world audit scenario where documents come from diverse sources.

All 100 compliance questions from the validation and test set are processed by the ARAG system using the annual report of Company X as input. For each question, the system produces a binary compliance assessment (Yes or No) based on whether what the questions ask is in the annual report. The response also includes an explanation and evidence drawn from the annual report. These responses are evaluated by five LVH auditors, all with more than 10 years of financial auditing experience. The auditors assess the system's output based on two criteria. First, auditors are asked to answer the question independently, without viewing the system's response. Their answers will be used as ground truth. Secondly, the quality of the justification refers to the relevance and sufficiency of the explanation and the supporting evidence provided by the system. Justification quality was rated on a five-point scale, with higher scores indicating greater clarity and alignment with professional audit standards.

To evaluate the consistency among the auditors' assessments, Fleiss' kappa [49] is calculated. Fleiss' kappa is a statistical measure that assesses the reliability of agreement between multiple raters when assigning categorical ratings to several items. It is especially useful when there are more than two raters, as in this case with five auditors. The kappa coefficient ranges from -1 to 1, where a value of 1 indicates perfect agreement and 0 indicates an agreement equivalent to chance, and negative values indicate poor agreement. A kappa between 0.61 and 0.80 is generally considered "substantial agreement", while values above 0.80 are considered "almost perfect agreement".

4.7.5. Ablation Experiments

To understand the contribution of the individual agents described in Section 4.2.1 within the ARAG system, a series of ablation experiments is conducted. These experiments systematically disable specific agents or agent groups to measure their impact on the performance of the

ARAG system. By comparing these results with the results of the validation phase and the test phase, it can be determined which agents are the most influential in the ARAG system.

In this experiment, there are four ablation configurations, each of which removes a specific component or a set of components from the ARAG pipeline. First, the `ExpandQueryAgent` is disabled, bypassing the Qeury2Doc method from Wang et al. [33]. This tests whether query expansion through contextual enrichments is beneficial for compliance verification. Secondly, the `SelectionAgent` is disabled, this will cause the system to use all retrieved documents without filtering for relevance. This tests the importance of filtering the retrieved documents before final answer generation. Third, a group of agents is disabled. Here, the `EvaluateContextAgent` and the `ExpandRetrievalAgent` are disabled. This removes the system's ability to evaluate the context sufficiency and perform additional retrievals, which evaluates whether iterative retrieval contributes to answer quality. Finally, the self-critic loop is disabled. Here, the `ValidationAgent` and the `RefinementAgent` are disabled. This tests whether the system's ability to validate and refine initial answers enhances performance.

Each ablation configuration is evaluated on both the validation set and the test set, using the metrics described in Section 4.8.

4.8. Evaluation Protocol

This section details how the ARAG system is evaluated in both the validation and test phases. As noted in Section 3.3.1, all the checklist items are formulated as yes/no questions. The system must decide whether the annual report is (yes) or is not (no) compliant with a given question.

4.8.1. Metrics

In this study, multiple metrics are used to measure performance: accuracy, precision, recall, and F_1-score.

- **Accuracy**: proportion of questions answered with the correct compliant/non-compliant decision.
- **Precision**: $\frac{TP}{TP+FP}$, the proportion of times the system correctly identified compliance among all instances labelled compliant. In other words, out of all the cases where the model indicated that "the report complies with requirement X," how many were correct? A low precision score suggests that the model frequently produces false positives.
- **Recall**: $\frac{TP}{TP+FN}$, ability to identify all truly compliant cases. That is, of all truly compliant items in the test, how many did the model successfully recognise as compliant? A lower recall would mean that the model misinterpreted compliant cases (false negatives).
- **F_1-score**: harmonic mean of precision and recall, calculated by $F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$ (optimisation objective).

These metrics provide a clear picture of the effectiveness of each model in answering compliance questions correctly. Accuracy summarises the overall success rate, while precision

and recall diagnose the model's errors. The F_1-score combines these aspects, which is useful for comparing models with different precision/recall trade-offs.

In addition to the standard metrics above, there are also LLM-based metrics. These metrics evaluate the quality and usefulness of the model's answers beyond just correctness. The metrics used are:

- **Context Relevance**: the ratio of relevant statements to all statements in the context. In other words, it measures how many statements in the context help answer the question [22].
- **Answer Relevance**: measures how similar the user's question is to the generated questions, based on the ARAG response. It is calculated by averaging the cosine similarity scores, showing how well the answer aligns with what the user was asking [22].
- **Faithfulness**: measures how well the generated statements match the retrieved context. It evaluates whether the context supports each statement from the ARAG response. The score is the ratio of the supported (valid) statements to the total number of statements in the response [22].

These three metrics are evaluated using an LLM to act as a judge. In practice, for each answer the system produces, a separate prompt is given to the LLM with the question, the model's answer, and the set of retrieved documents that the model used. The LLM is instructed to analyse these and produce an assessment for each of the above aspects. The LLM used for this purpose is GPT-4o-mini from OpenAI [50], a state-of-the-art yet cost-effective model. It is chosen for its strong performance and is 8 times cheaper than the full GPT-4o model, making it suitable for efficient processing at scale. To ensure data privacy and prevent the exposure of sensitive information to external servers, the content of the annual report is masked before submission. All names, numerical values, and other identifiable entities are replaced with dummy placeholders. All models are compared on these metrics, to see not just which model is the most "correct", but also which one provides the most relevant and well-supported answers. For details of the computations of the LLM-based metrics, see Appendix B.2.

Finally, a human evaluation is performed on the best model, as mentioned in Section 4.7.2. The auditors were asked to rate each answer according to correctness (whether the system's conclusion about compliance was correct, in their opinion) and justification quality (whether the answer was well explained and supported by appropriate references to the report or the law). Together, the combination of automatic metric evaluation on a held-out test set, LLM-based scoring, and human expert judgement provides a comprehensive evaluation of the performance of the ARAG system on the task of Dutch annual report compliance evaluation.

5

Results

This chapter presents the findings of evaluating the ARAG system for auditing Dutch annual reports. The results are organised to address the research questions outlined in Chapter 1, examining how effectively an ARAG system leveraging Dutch regulatory documents can assess compliance in financial reporting. The findings are presented in six main sections, covering hyperparameter optimisation, test set evaluation, real-world study, ablation studies, computational cost analysis, and a production application.

5.1. Hyperparameter Optimisation and Validation

The first phase of the experimental evaluation focused on exploring promising configurations for each language model, in the ARAG system, through Bayesian optimisation on the validation set of 60 compliance questions. All LLM-based agents in the ARAG system used the same LLM during validation. As described in Section 4.7.1, 35 optimisation trials were conducted for each model. The optimisation trials explored a two-dimensional search space with chunk sizes $S \in [128, 1024]$ tokens and top-$k \in [5, 25]$.

5.1.1. Validation Results for Models in the ARAG System

This section presents the configurations that perform the best during the Bayesian optimisation process for each language model in the ARAG system. For each model, the top five configurations ranked by the F_1-score and their corresponding evaluation metrics are reported.

The configurations are defined by three key parameters:

- **k**: The number of documents retrieved from the database.
- **cs**: Chunk size in tokens.
- **co**: Chunk overlap in tokens, which is always 20% of the chunk size, as stated in Section 4.3.1.

All configurations are evaluated using standard classification metrics (e.g., Accuracy, Precision, Recall, F_1-score) and LLM-Based evaluation metrics (e.g., Faithfulness, Answer relevance, Context relevance) as described in Section 4.8.1.

Llama 3.1-8B-Instruct
One of the smallest models in this study, Llama 3.1-8B-Instruct [45], demonstrated the following performance across its top 5 configurations, see Table 5.1:

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=9, cs=763, co=153	**0.717**	0.731	**0.704**	0.750	**0.836**	0.723	0.502
k=6, cs=772, co=154	0.708	0.810	0.630	**0.767**	**0.836**	0.723	0.502
k=9, cs=760, co=152	0.706	0.750	0.667	0.750	0.818	0.735	0.529
k=6, cs=775, co=155	0.694	0.773	0.630	0.750	0.800	0.724	0.529
k=9, cs=768, co=154	0.682	**0.882**	0.556	**0.767**	0.835	**0.739**	**0.531**

Table 5.1: Top 5 Parameter Configurations by F1 Score for model: Llama 3.1-8B-Instruct.

In Table 5.2 are the top 5 configurations shown sorted by precision.

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=9, cs=768, co=154	0.682	**0.882**	0.556	**0.767**	0.835	**0.739**	0.531
k=25, cs=437, co=87	0.636	0.824	0.519	0.700	0.835	0.727	0.526
k=9, cs=766, co=153	0.636	0.824	0.519	0.733	**0.846**	0.734	0.478
k=6, cs=772, co=154	**0.708**	0.810	**0.630**	**0.767**	0.836	0.723	0.502
k=5, cs=770, co=154	0.652	0.790	0.556	0.733	0.826	0.727	**0.549**

Table 5.2: Top 5 Parameter Configurations by precision for model: Llama 3.1-8B-Instruct.

For the Llama 3.1-8B-Instruct model, Bayesian optimisation identified a configuration with a chunk size of 763 tokens, a chunk overlap of 153 tokens, and a top-k value of 9 that achieved the highest F_1-score observed during the search, as shown in Table 5.1.

A particularly notable observation in the Llama 3.1-8B-Instruct results is the dramatic shift in the precision-recall balance between configurations with minimal differences in chunk size. The configuration with $k = 9$ and chunk size = 768 achieved the highest precision (88%) among all the configurations tested but at the cost of a lower recall (56%), despite being only five tokens larger than the best configuration found ($k = 9$, chunk size = 763) that more effectively balanced both metrics.

The precision is the primary concern in financial compliance, since falsely approving non-compliant items carries greater regulatory risks than missing compliant ones. As shown in Table 5.2, the configuration with a chunk size of 768 and a value of $k = 9$ achieves the highest precision of all configurations. In practice, auditors will thoroughly check all items flagged as non-compliant, making the minimisation of false positives crucial.

All results of the 35 Bayesian optimisation calls for Llama 3.1-8B-Instruct can be found in Appendix C in Table C.1.

Qwen3-8B
The other model of the same size as Llama 3.1-8B-Instruct is the newer model Qwen3-8B [47]. Qwen3-8B demonstrated the following performance across its top 5 configurations, see Table 5.3.

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=6, cs=128, co=26	**0.809**	**0.950**	**0.704**	**0.850**	**0.893**	**0.747**	0.364
k=6, cs=192, co=38	0.723	0.850	0.630	0.783	0.892	0.736	0.402
k=21, cs=306, co=61	0.708	0.810	0.630	0.767	0.850	0.728	0.543
k=21, cs=339, co=68	0.708	0.810	0.630	0.767	0.846	0.739	0.549
k=21, cs=340, co=68	0.696	0.842	0.593	0.767	0.840	0.745	**0.556**

Table 5.3: Top 5 Parameter Configurations by F1 Score for model: Qwen3-8B.

In Table 5.4 are the top 5 configurations shown sorted by precision.

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=6, cs=128, co=26	**0.809**	**0.950**	**0.704**	**0.850**	**0.893**	**0.747**	0.364
k=21, cs=375, co=75	0.667	0.933	0.519	0.767	0.858	0.740	0.535
k=21, cs=347, co=69	0.667	0.933	0.519	0.767	0.847	0.742	0.559
k=17, cs=414, co=83	0.634	0.929	0.482	0.750	0.869	0.746	0.520
k=25, cs=681, co=136	0.634	0.929	0.482	0.750	0.876	0.734	**0.669**

Table 5.4: Top 5 Parameter Configurations by precision for model: Qwen3-8B.

For the Qwen3-8B model, Bayesian optimisation identified a promising configuration with a chunk size of 128 tokens, a chunk overlap of 26 tokens, and a top-k value of 6. This results in an F_1-score of 0.81 on the validation set, as shown in Table 5.3.

The Qwen3-8B model, while having a parameter count similar to Llama 3.1-8B-Instruct, exhibited different hyperparameter preferences, suggesting architectural differences between these models despite their comparable size. The greatest difference appears in the best-found chunk size. Although Llama 3.1-8B-Instruct performed best with chunk sizes between 763-775 tokens, Qwen3-8B achieved its peak performance with a substantially smaller chunk size of 128 tokens. This difference in chunk size suggests that Qwen3-8B processes information more effectively with a precise, targeted context, rather than broader text spans. The smaller chunk size likely allows the model to focus on the most relevant chunks without being distracted by the surrounding content.

Qwen3-8B achieves a remarkably high precision (95%) but a more moderate recall (70%). This suggests that when Qwen3-8B identifies compliant items, it does so with high confidence. In particular, the best configuration found for Qwen3-8B also provides the highest precision among all the tested configurations, as presented in Table 5.4. This is particularly valuable for compliance verification, where false positives have a greater regulatory risk than false negatives. The ability to maintain high precision while still achieving moderate recall makes this model an interesting candidate.

All results of the 35 Bayesian optimisation calls for Qwen3-8B can be found in Appendix C in Table C.2.

Qwen2.5-32B-Instruct
Bayesian optimisation identified the following top 5 parameters for Qwen2.5-32B-Instruct, see Table 5.5:

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=14, cs=441, co=88	**0.880**	**0.957**	**0.815**	**0.900**	0.841	0.748	0.528
k=14, cs=449, co=90	0.857	0.955	0.778	0.883	0.846	**0.751**	0.485
k=14, cs=452, co=90	0.840	0.913	0.778	0.867	0.856	0.743	**0.547**
k=14, cs=447, co=89	0.840	0.913	0.778	0.867	**0.860**	0.741	0.518
k=14, cs=446, co=89	0.840	0.913	0.778	0.867	0.828	0.739	0.499

Table 5.5: Top 5 Parameter Configurations by F1 Score for model: Qwen2.5-32B-Instruct.

In Table 5.6 are the top 5 configurations shown sorted by precision.

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=14, cs=441, co=88	**0.880**	**0.957**	**0.815**	**0.900**	0.841	0.748	**0.528**
k=14, cs=449, co=90	0.857	0.955	0.778	0.883	**0.846**	**0.751**	0.485
k=14, cs=427, co=85	0.809	0.950	0.704	0.850	0.814	0.744	0.483
k=18, cs=879, co=176	0.783	0.947	0.667	0.833	0.829	0.744	0.456
k=14, cs=442, co=88	0.783	0.947	0.667	0.833	0.829	0.750	0.501

Table 5.6: Top 5 Parameter Configurations by precision for model: Qwen2.5-32B-Instruct.

One can see from Table 5.5 that the configuration for the Qwen2.5-32B-Instruct model with a chunk size of 441, a chunk overlap of 88, and $k = 14$ is the best configuration found by Bayesian optimisation. Based on classic classification metrics, this configuration yields the strongest performance on the validation set with an F_1-score of 0.88. This model achieved an accuracy of 90%, which means that from the 60 questions it answered 54 questions correctly, with a precision of 96% and a recall of 81%. The best performing configurations for this model consistently featured chunk sizes in a narrow range (441-452 tokens) and all required a top-k value of 14.

All results of the 35 Bayesian optimisation calls for Qwen2.5-32B-Instruct can be found in Appendix C in Table C.3.

Llama 3.3-70B-Instruct
As shown in Table 5.7, Llama 3.3-70B-Instruct, the largest model evaluated in this study with 70 billion parameters, performed very well on the validation set.

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=11, cs=164, co=33	**0.920**	**1.000**	**0.852**	**0.933**	0.862	**0.753**	0.427
k=25, cs=163, co=33	**0.920**	**1.000**	**0.852**	**0.933**	0.831	0.747	**0.444**
k=24, cs=129, co=26	0.902	0.958	**0.852**	0.917	**0.909**	0.750	0.426
k=25, cs=195, co=39	0.902	0.958	**0.852**	0.917	0.847	0.743	0.442
k=11, cs=187, co=37	0.902	0.958	**0.852**	0.917	0.855	0.746	0.411

Table 5.7: Top 5 Parameter Configurations by F1 Score for model: Llama-3.3-70B-Instruct.

In Table 5.8, the top 5 configurations are shown sorted by precision.

Config	F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
k=7, cs=929, co=186	0.898	**1.000**	0.815	0.917	**0.881**	0.741	**0.501**
k=12, cs=161, co=32	0.898	**1.000**	0.815	0.917	0.851	0.747	0.431
k=25, cs=163, co=33	**0.920**	**1.000**	**0.852**	**0.933**	0.831	0.747	0.444
k=11, cs=164, co=33	**0.920**	**1.000**	**0.852**	**0.933**	0.862	**0.753**	0.427
k=20, cs=128, co=26	0.898	**1.000**	0.815	0.917	0.877	0.746	0.397

Table 5.8: Top 5 Parameter Configurations by precision for model: Llama-3.3-70B-Instruct.

Bayesian optimisation identified two configurations that resulted in the same classification metrics, as shown in Table 5.7. The first configuration has a chunk size of 164 tokens, a chunk overlap of 33 tokens, and a top-k value of 11. The second configuration with identical performance metrics has a chunk size of 163 tokens, a chunk overlap of 33 tokens, and a top-k value of 25. Despite the significant difference in the number of documents retrieved (11 versus 25), both configurations achieved the same F_1-score of 0.92 on the validation set.

Both configurations reached an accuracy of 93% on the validation set, correctly answering 56 out of the 60 validation questions. In particular, both achieved perfect precision (100%) while maintaining a high recall of 85%. This perfect precision score is unique among all models tested and suggests that Llama 3.3-70B-Instruct makes extremely reliable judgments when identifying compliance.

The perfect precision is further confirmed in Table 5.8, which shows that all top five configurations, sorted by precision, achieved an ideal 100% score. This consistency in achieving perfect precision scores across different configurations demonstrates that the model is robust in avoiding false positives, which is critical for compliance verifications.

Although the classification metrics were identical, the LLM-based evaluation metrics showed slight differences between these two configurations. The configuration with $k = 11$ achieved a faithfulness score of 0.86 and an answer relevance of 0.75, while the configuration with $k = 25$ had a slightly lower faithfulness score of 0.83 but a marginally higher context relevance of 0.44 compared to 0.43 for the $k = 11$ configuration. This relationship between retrieval quantity and context relevance follows a logical pattern, namely, the more you retrieve, the more potentially relevant context you include. However, the faithfulness suggests that

simply retrieving more content does not necessarily lead to better answer quality. Too much information can introduce noise or distract the model from the relevant details.

What distinguishes Llama 3.3-70B-Instruct from the other models is its perfect precision on the validation set, combined with strong recall. This suggests that when this model identifies a compliance requirement as being met, it is always correct (within the validation set), while still identifying a high proportion of the truly compliant cases. For auditing purposes, this balance of precision and recall is particularly valuable, as it minimises the risk of overlooking non-compliant items.

Like with the other models, it is important to emphasise that these results are based solely on the validation set. Performance on the separate held-out test set of 40 questions may differ, as this test set represents a true evaluation of the model's generalisation capabilities and is not used during hyperparameter optimisation.

All results of the 35 Bayesian optimisation calls for Llama 3.3-70B-Instruct can be found in Appendix C in Table C.4.

5.2. Evaluation on the Held-out Test Set

After identifying the hyperparameters for each model in the ARAG system during the validation phase, the next step is to evaluate their performance on a held-out test set. This held-out test set consists of 40 compliance questions, as mentioned in Section 3.3.1. Using the held-out test set provides an unbiased assessment of the ARAG's generalisation capabilities per model. During this evaluation, the best configurations for each model from the validation phase, sorted by the F_1-score, are used.

5.2.1. Performance of the Models with the ARAG System

Llama 3.1-8B-Instruct
As shown in Table 5.9, the use of the Llama 3.1-8B-Instruct model in the ARAG system achieved an accuracy of 60% on the test set, with a precision of 85% and a recall of 44%, resulting in an F_1-score of 0.58. The confusion matrix, shown in Figure 5.1, reveals that although the system correctly identified 13 out of the 15 non-compliant items, it struggled to identify compliant items. One can see from Figure 5.1 that the system only correctly classified 11 out of the 25 compliant cases. Despite this classification imbalance, the system maintained good faithfulness, answer relevance, and decent context relevance. This suggests that the system effectively used the retrieved context in formulating a response.

F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
0.579	0.846	0.440	0.600	0.837	0.739	0.581

Table 5.9: ARAG results on the test set for model: Llama 3.1-8B-Instruct, , with $k = 9$, chunk size set to 763 tokens, and a chunk overlap of 153 tokens.

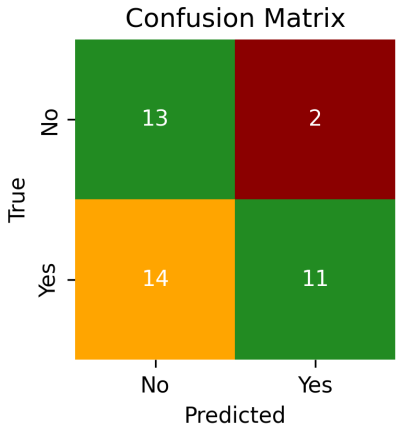


Figure 5.1: Confusion matrix for the model Llama 3.1-8B-Instruct on the test set.

Qwen3-8B

Using the Qwen3-8B model in the ARAG system demonstrated the lowest overall performance on the test set, as detailed in Table 5.10. The system achieved an accuracy of 55%, a precision of 82% and a recall of 36%. Especially the low recall resulted in an F_1-score of 0.50. Looking at the confusion matrix, shown in Figure 5.2, shows similar patterns to the use of the Llama 3.1-8B-Instruct model. The system has a strong performance on identifying non-compliant items, which are again 13 out of 15, but even poorer performance on compliant items. The system could only correctly identify 9 of the 25 compliant questions. This indicates an even stronger bias towards predicting non-compliance. Using Qwen3-8B in the ARAG system achieved the highest faithfulness score (0.89) and context relevance (0.60) among all models used. This suggests that while its classification decisions were often incorrect, the explanations of the system were factually aligned with the retrieved context.

F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
0.500	0.818	0.360	0.550	0.888	0.755	0.600

Table 5.10: ARAG results on the test set for model: Qwen3-8B, with $k = 6$, chunk size set to 128 tokens, and a chunk overlap of 26 tokens.

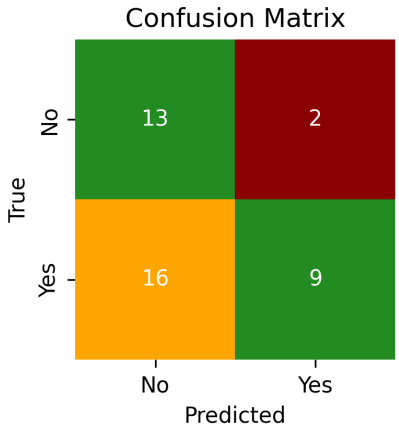


Figure 5.2: Confusion matrix for the model Qwen3-8B on the test set.

Qwen2.5-32B-Instruct

Using the Qwen2.5-32B-Instruct model in the ARAG system showed improved performance, shown in Table 5.11, compared to the 8B models. On the test set, it achieved an accuracy of 63%, a precision of 86% and a recall of 48%, resulting in an F_1-score of 0.62. Although the metrics showed improvement, the confusion matrix in Figure 5.3 still shows that the model has a strong performance on non-compliant questions and a weaker performance on compliant questions. One can see from Figure 5.3 that 13 of the 15 non-compliant questions were correctly answered, while only 12 of the 25 compliant questions were correctly answered. The system maintained a high faithfulness of 0.88, although its answer relevance of 0.61 is the lowest among all models tested. This can indicate that, while their responses were factually grounded, they sometimes did not directly address the specific compliance question asked.

F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
0.615	0.857	0.480	0.625	0.882	0.606	0.567

Table 5.11: ARAG results on the test set for model: Qwen2.5-32B-Instruct, with $k = 14$, chunk size set to 441 tokens, and a chunk overlap of 88 tokens.

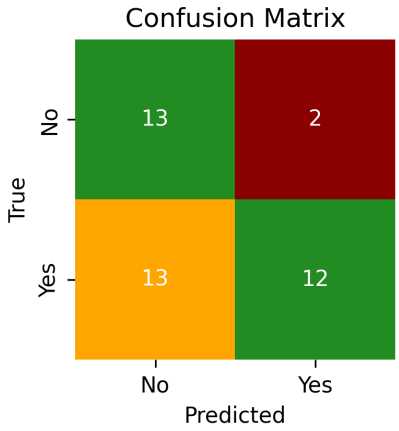


Figure 5.3: Confusion matrix for the model Qwen2.5-32B-Instruct on the test set.

Llama 3.3-70B-Instruct

As detailed in Table 5.12, using the Llama 3.3-70B-Instruct model demonstrated the best performance on the test set. The system achieved an accuracy of 73%, a precision of 89%, and a recall of 64%, resulting in an F_1-score of 0.74. The confusion matrix, shown in Figure 5.4, reveals a significantly different pattern compared to the other models used. Although it maintained a strong performance on non-compliant questions, where it still correctly answered 13 out of 15, it showed substantially better performance in correctly identifying compliance. The system correctly answered 16 of the 25 compliant questions. The system achieved the highest faithfulness of 0.90 and an answer relevance of 0.76 among all models tested.

F_1	Prec	Rec	Acc	Faithful	AnsRel	CtxRel
0.744	0.889	0.640	0.725	0.897	0.761	0.543

Table 5.12: ARAG results on the test set for model: Llama 3.3-70B-Instruct, with $k = 11$, chunk size set to 164 tokens, and a chunk overlap of 33 tokens.

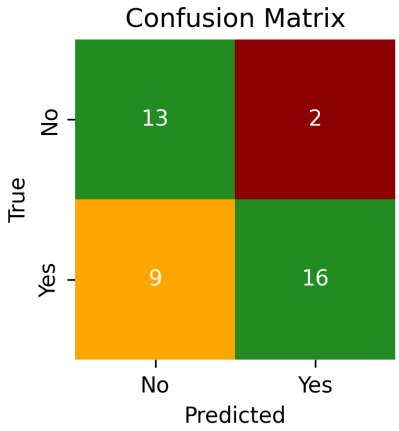


Figure 5.4: Confusion matrix for the model Llama 3.3-70B-Instruct on the test set.

5.2.2. Model Comparison

When comparing the confusion matrices across all the models used (see Table 5.13), a clear pattern can be seen. All models show consistent performance in correctly identifying non-compliant questions, which are for all models 13 out of the 15 questions. The main differences between models appear in their ability to identify compliant items, see Table 5.13

Model	**Confusion Matrix**
Llama 3.1-8B-Instruct	$\begin{pmatrix} \text{TN} = 13 & \text{FP} = 2 \\ \text{FN} = 14 & \text{TP} = 11 \end{pmatrix}$
Qwen3-8B	$\begin{pmatrix} \text{TN} = 13 & \text{FP} = 2 \\ \text{FN} = 16 & \text{TP} = 9 \end{pmatrix}$
Qwen2.5-32B-Instruct	$\begin{pmatrix} \text{TN} = 13 & \text{FP} = 2 \\ \text{FN} = 13 & \text{TP} = 12 \end{pmatrix}$
Llama 3.3-70B-Instruct	$\begin{pmatrix} \text{TN} = 13 & \text{FP} = 2 \\ \text{FN} = 9 & \text{TP} = 16 \end{pmatrix}$

Table 5.13: Confusion matrices for all models on the test set. TN = True Negative (correctly identified non-compliant), FP = False Positive (incorrectly identified as compliant), FN = False Negative (incorrectly identified as non-compliant), TP = True Positive (correctly identified compliant).

The results presented in Figure 5.5 and Figure 5.6 show the performance metrics for all models. Llama 3.3-70B-Instruct achieved the highest values in most metrics, with an F_1-score of 0.74, precision of 0.89, a recall of 0.64, and an accuracy of 0.73. On the LLM-based metrics, the model achieved a faithfulness of 0.90, an answer relevance of 0.76, and a context relevance of 0.54. The Qwen3-8B model achieved the highest context relevance of 0.60, despite having

the lowest F_1-score and recall. All models maintained relatively high precision scores ranging from 0.82 to 0.89.

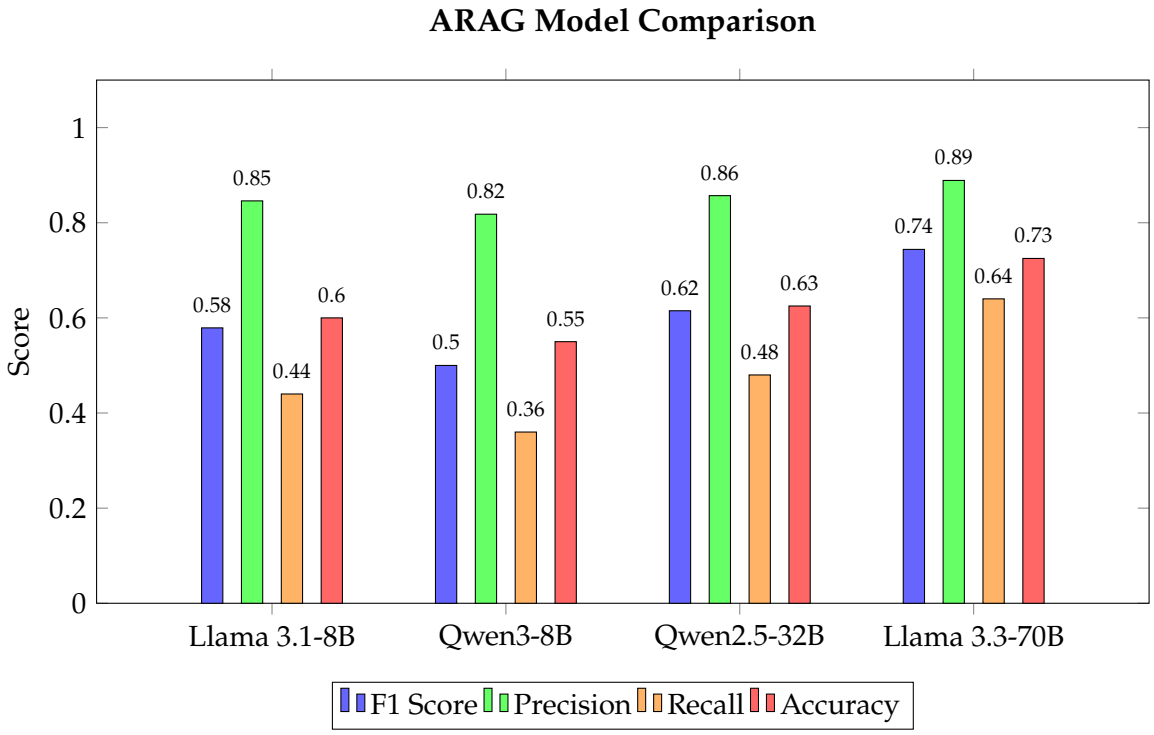


Figure 5.5: ARAG Model Performance Comparison.

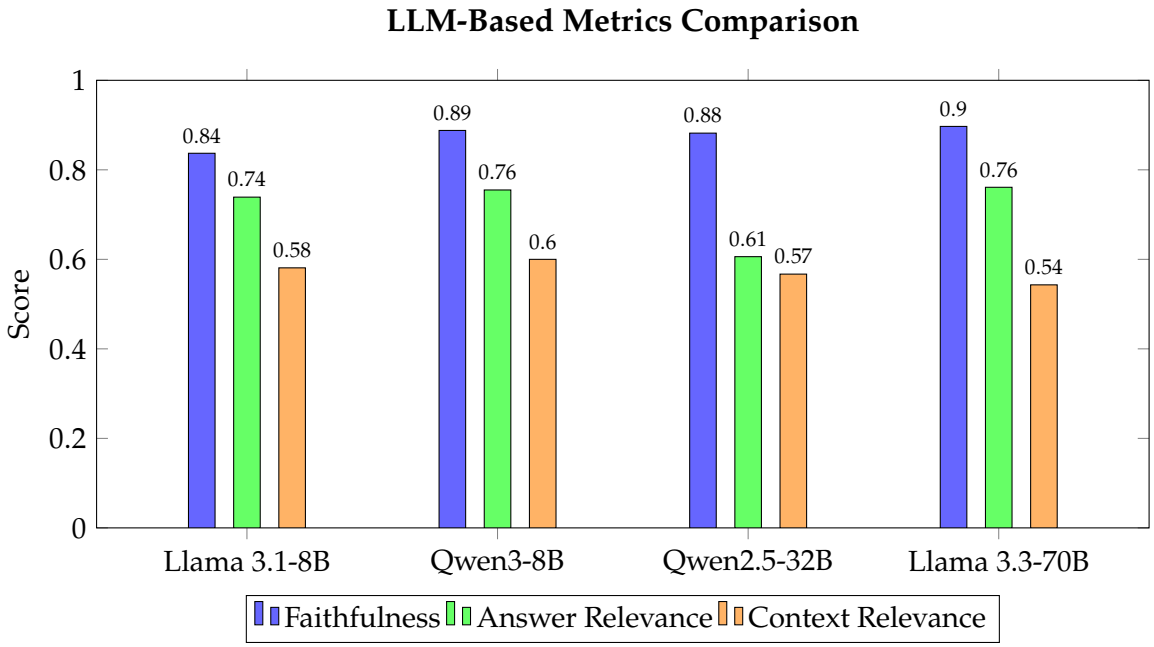


Figure 5.6: LLM-Based Quality Metrics Comparison.

5.2.3. Comparison with Baseline (Non-RAG) Performance

To evaluate the effectiveness of the ARAG system, the performance of each model is compared with its corresponding non-RAG baseline. For the non-RAG baseline, the LLM is provided with the entire annual report, which was used in both validation and testing, and the compliance question without the benefit of retrieval or agent-based reasoning.

As shown in Figure 5.7, the ARAG system significantly improves the performance of most models. Llama 3.3-70B-Instruct demonstrates the most substantial improvements, as shown in Figure 5.7, with improvements across all classification metrics (+20% in F_1-score, +16% in precision, +23% in recall, +21% in accuracy). The smaller Llama 3.1-8B-Instruct model within the ARAG system shows a great improvement in precision (+38%), while maintaining the same recall. The Qwen2.5-32B-Instruct model within the ARAG system improved moderately in all metrics, particularly in recall (+20%). Using the Qwen3-8B model in the ARAG system shows mixed results, while precision improves (+19%), the recall decreased by −18%, resulting in a slight drop in F_1-score, as shown in Figure 5.7.

The complete comparison data are available in Appendix C.2, Table C.5.

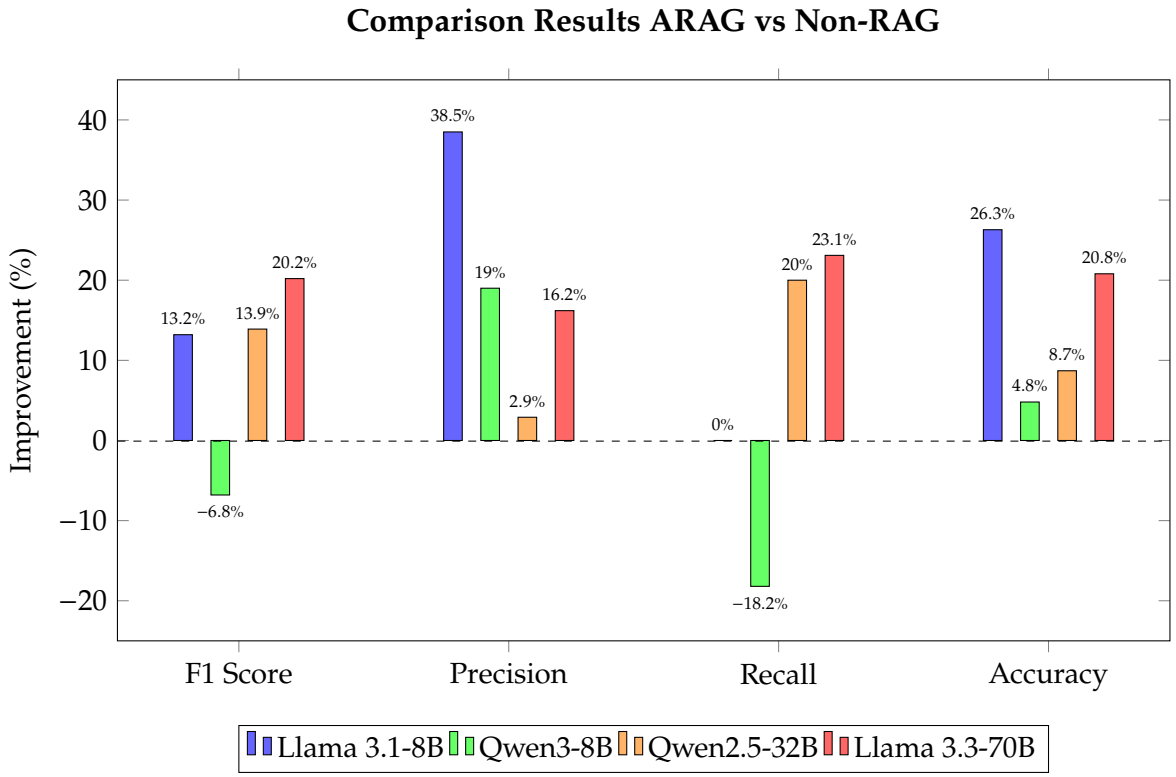


Figure 5.7: Comparison of ARAG vs Non-RAG performance across all models. Improvement is calculated as the percentage increase from Non-RAG to ARAG performance.

5.3. Human Evaluation of ARAG System

A case study is conducted to assess the real-world applicability of the ARAG system, as described in Section 4.7.4. This case study uses a complete annual report from a medium-sized Dutch company. This section presents the findings of the real-world case study, where all the 100 compliance questions from the combined validation and test set were applied to a different annual report without predefined answers. The Llama 3.3-70B-Instruct model with its best configuration, which achieved the highest performance in previous evaluations, see Section 5.2, is selected for this case study.

Figure 5.8 presents the results of the compliance assessments of the ARAG system evaluated by human auditors.

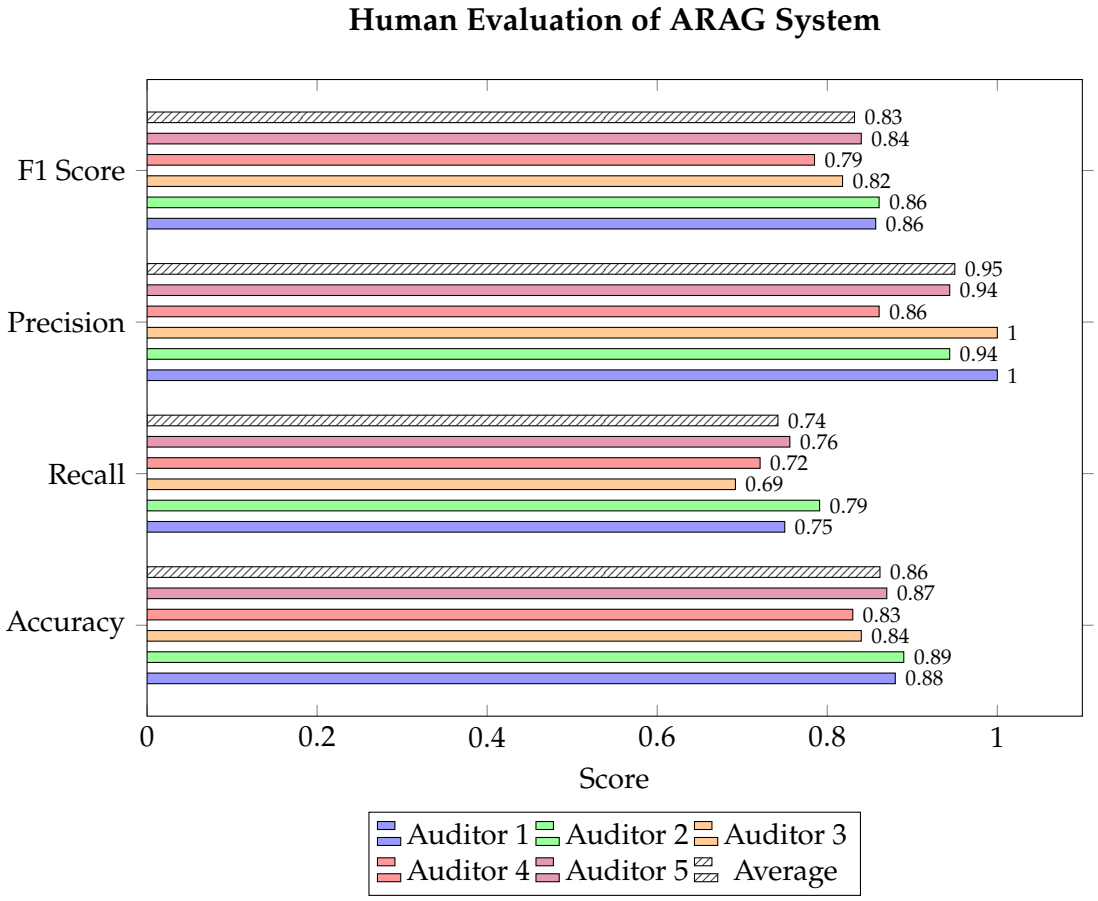


Figure 5.8: Result of ARAG system on the annual report of "Company X" evaluated by human auditors.

The ARAG system demonstrated strong performance across all five auditors. The system achieves an average F_1-score of 0.83, with individual auditor scores ranging from 0.79 to 0.86. The precision scores are particularly high, with two auditors (Auditor 1 and Auditor 3) rating the system's precision at a perfect 1.00, and an average precision of 0.95 across all auditors. Recall scores show more variation, ranging from 0.69 to 0.79, with an average of 0.74. The overall accuracy of the system is 0.86, with individual scores ranging from 0.83 to 0.89. For

the binary compliance assessments (Yes/No) made by the five auditors, the Fleiss' kappa coefficient is 0.73.

The quality of the explanations and supporting evidence is rated on a scale of 1-5, with the results shown in Figure 5.9. The percentages reported in Figure 5.9 are calculated by calculating the mean proportion of responses falling into each rating category among all auditors.

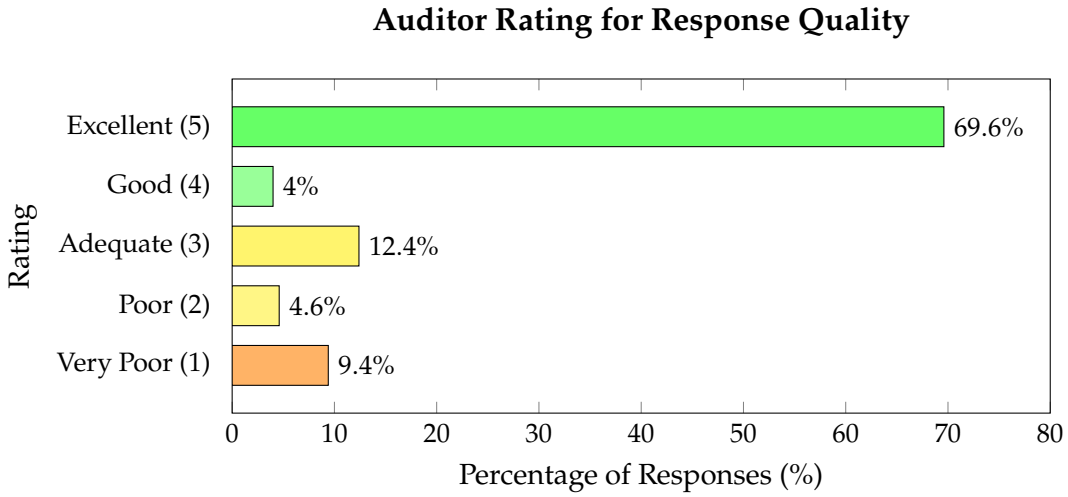


Figure 5.9: Auditor ratings of justification quality for the responses of the ARAG system

The auditors' assessment of the justification quality showed that 70% of the ARAG system's responses were rated as "Excellent". The quality of the justification is rated as "Good" in 4% of the cases, "Adequate" in 12%, "Poor" in 5%, and "Very Poor" in 9% of the cases. In total, 86% of the responses received ratings 3 or higher.

5.4. Ablation Experiments

To understand the contribution of the individual agents described in Section 4.2.1 within the ARAG system, a series of ablation experiments is conducted, as described in Section 4.7.5. These experiments systematically disable specific agents or groups of agents to measure their impact on the performance of the ARAG system. By comparing these results with the results of Section 5.1 and Section 5.2, it can be determined which agents are most critical in the ARAG system.

The ablation experiments focused on using the Llama 3.3-70B-Instruct model with the best configuration found (chunk size of 164 tokens, chunk overlap of 33, and top-k value of 11). This model is chosen because it demonstrates the best performance in both the validation and the test phases. Ablation experiments are performed on both the validation and test sets.

Impact on Validation Set

The results of the ablation experiments on the validation set are shown in Figure 5.10. The results show that removing the `ExpandQueryAgent` or the `SelectionAgent` resulted in

identical precision (0.96) and recall (0.78), resulting in an F_1-score of 0.86. This is only a slight decrease from the full performance of the system (precision of 1.00, recall of 0.85, and F_1-score of 0.92), as can be seen in Figure 5.10. In particular, while there is a small decrease in precision from the perfect score of the entire system, the precision remains very high at 0.96.

Removing the `EvaluateContextAgent` and `ExpandRetrievalAgent` maintained a high precision of 0.94, the recall decreased significantly to 0.56, resulting in an F_1-score of 0.70. This represents a 24% reduction in F_1-score compared to the full system's validation performance.

Disabling the `ValidationAgent` and `RefinementAgent` did not substantially affect the performance on the validation set. Removing these two agents resulted in the same precision, recall and F_1-score as removing the `ExpandQueryAgent` and `SelectionAgent`. This indicates that for the validation set, the initial responses generated were often sufficient without requiring refinement or validation.

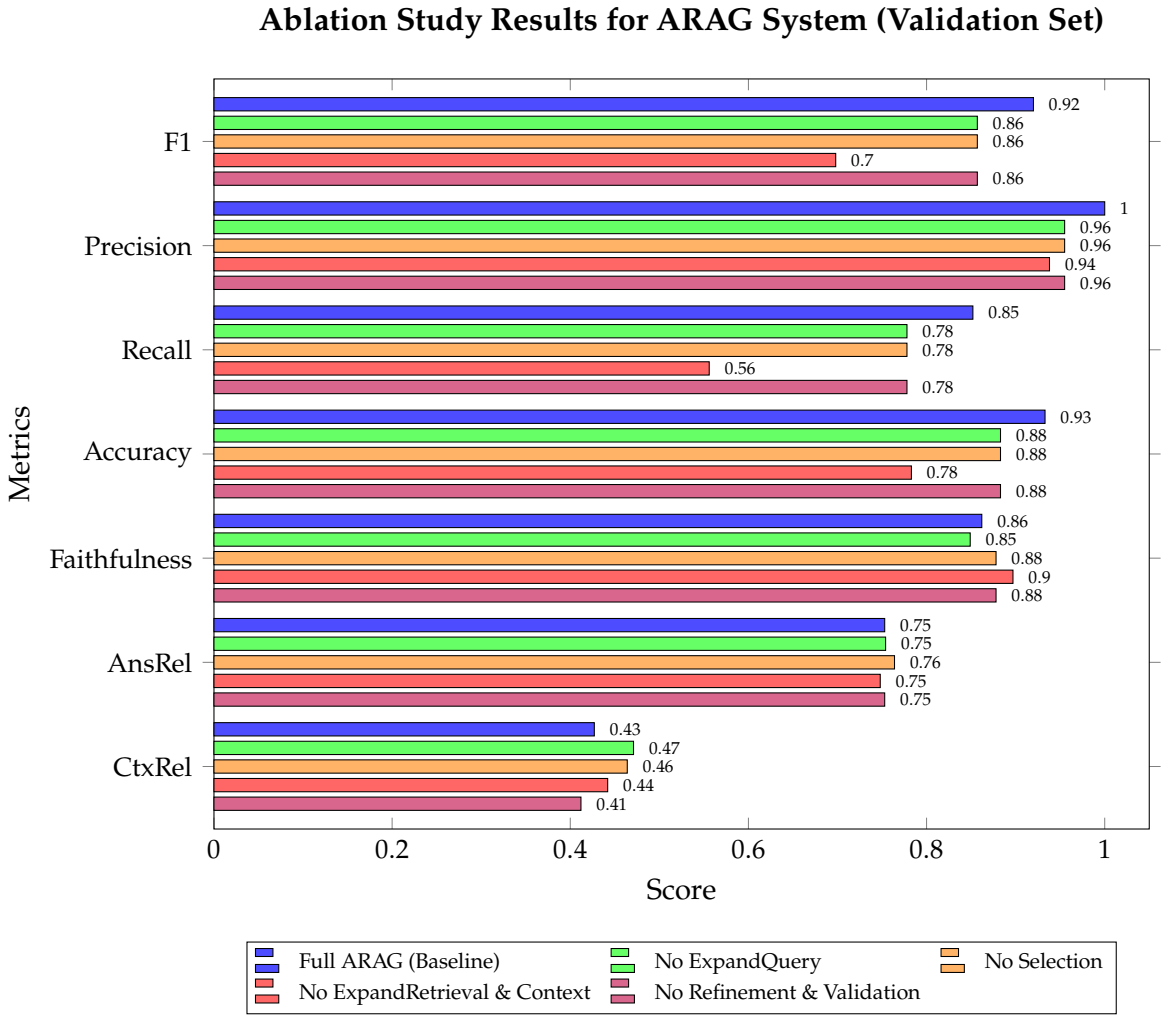


Figure 5.10: Ablation results of the ARAG system on the validation set.

Impact on Test Set
The results of the ablation experiments on the test set are shown in Figure 5.11. These findings offer insights into the impact of each component removal on the ARAG system's performance when applied to unseen data. In auditing, precision is particularly critical, as it indicates how reliable the system is when it identifies compliance. High precision means fewer false positives that could lead auditors to overlook actual compliance issues.

Removing the `ExpandQueryAgent` resulted in a precision of 0.88, with the recall dropping to 0.60 from 0.64, resulting in an F_1-score of 0.71. This represents a 4% decrease in the performance of the complete ARAG system, as can be seen in Figure 5.11.

Disabling the `SelectionAgent` led to a more notable decrease in precision, while maintaining a similar recall, as presented in Figure 5.11.

The removal of the `EvaluateContextAgent` and `ExpandRetrievalAgent` has the most significant impact on performance, as is also the case on the validation set. This ablation achieved the highest precision among all test configurations, shown in Figure 5.11, which is even higher than the performance of the full ARAG system. This higher precision comes with the cost of a lower recall, resulting in the lowest F_1-score.

Removing the `ValidationAgent` and `RefinementAgent` resulted in keeping a relatively high precision, but the recall decreased to 0.56, leading to an F_1-score of 0.68, as can be seen in Figure 5.11. This suggests that for more challenging or ambiguous cases, the refinement and validation processes are important to maintain high recall without sacrificing precision.

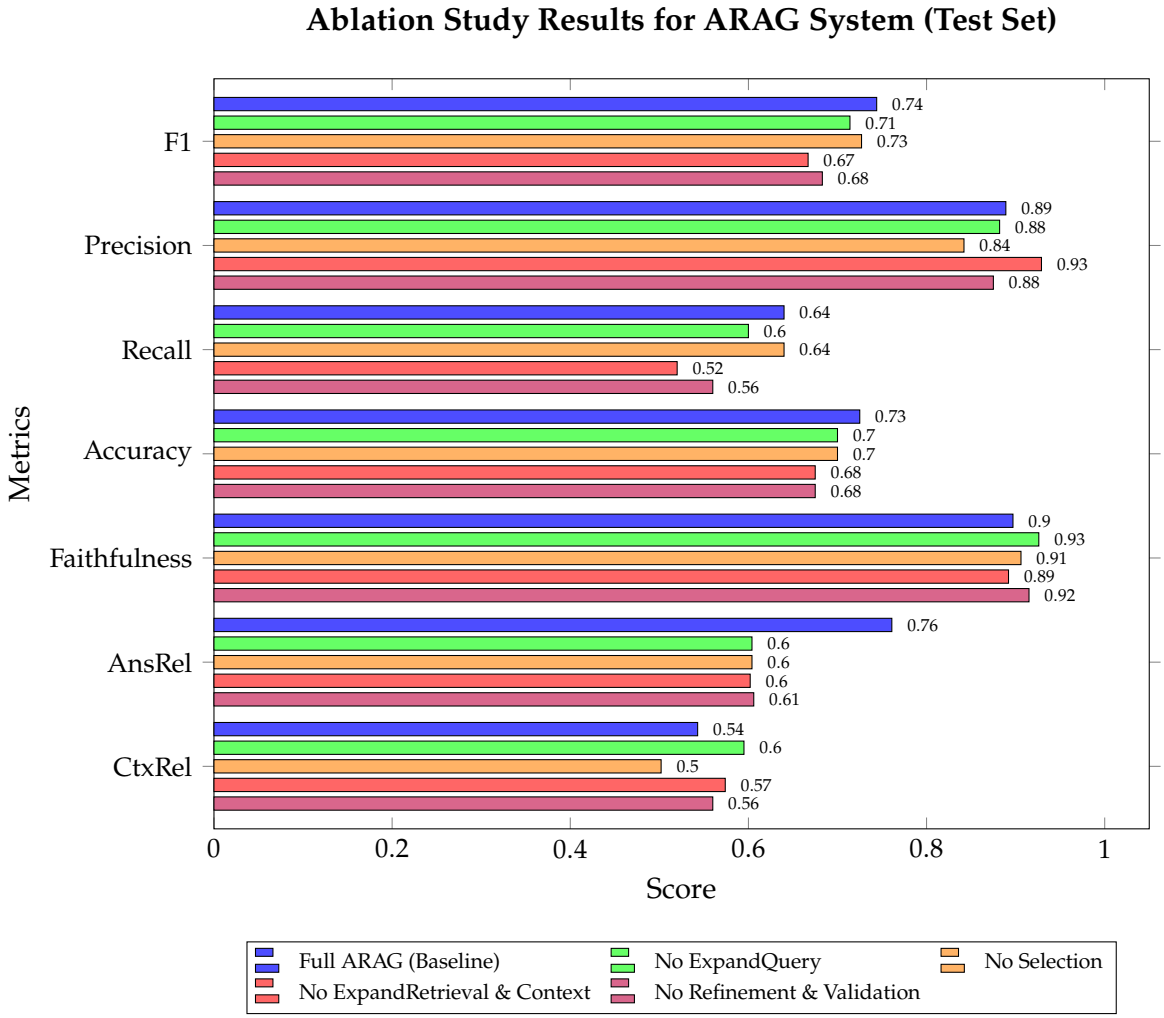


Figure 5.11: Ablation results of the ARAG system on the test set.

LLM-based Metrics in Ablation Studies

An interesting observation in the ablation experiments is the variation in LLM-based evaluation metrics. In some cases, the ablated agents exhibited higher scores on certain metrics compared to the full ARAG system. For example, removing the `ExpandQueryAgent` resulted in a higher faithfulness score on the test set compared to the full ARAG system, as presented in Figure 5.11. Similarly, when the `EvaluateContextAgent` and `ExpandRetrievalAgent` were removed, the context relevance score increased from 0.54 to 0.57 on the test set.

Another interesting result is when removing the `ValidationAgent` and `RefinementAgent`, the faithfulness also increases, as shown in Figure 5.11, on the test set. This counterintuitive result can be explained by the fact that, without these agents, the system generates simpler answers that reflect more directly the retrieved context. The refinement and validation process introduces additional reasoning that, while improving overall accuracy, can link the answer and the supporting context less explicitly.

These observations highlight that in this case, LLM-based metrics should be interpreted

alongside classification metrics, as they may not always align with the primary objective of compliance verification.

5.5. Cost Analysis

The preceding sections have focused on the precision and effectiveness of the ARAG system to check compliance with Dutch annual reports. A practical implementation also requires one to consider computational efficiency, resource requirements, and response times. The computational costs are crucial for audit firms, in this case Londen & Van Holland, that are seeking to implement AI-assisted compliance verification in a production environment. This section examines the computational costs associated with the deployment of the four different LLMs within the ARAG system.

In this analysis, two deployment scenarios are compared. First, an on-premises GPU server hosted by the company itself, and secondly, an Azure GPU virtual machine (VM) [51] (West-Europe region).

5.5.1. Hardware Requirements and Performance Metrics

Table 5.14 provides a comparison of GPU memory requirements, inference latency, and throughput capabilities for each model in both sequential and batch mode.

Model	Mem (BF16)	Peak Mem (batch of 8)	Seq. Time (min)	Batch Time (min)	Q/h (Batch)
Llama 3.1-8B-Instruct	16GB	50GB	5	5.6	86
Qwen3-8B	16GB	50GB	5	5.6	86
Qwen2.5-32B-Instruct	64GB	98GB	6	9	53
Llama 3.3-70B-Instruct	140GB	174GB	6	9	53

Table 5.14: GPU memory and performance metrics per model. Q/h = Queries per hour.

From Table 5.14 one can see that batch mode is more efficient, allowing the GPU to handle 8 queries at once and using its full capacity. The batch mode translates to approximately 7 times better throughput. This makes the batch mode essential for production environments.

5.5.2. Hardware Costs for On-premises Deployment

For on-premises deployment, initial hardware investments represent the most significant cost component. To host a server, one needs several components, for example, the smaller models already would need at least two RTX 4090 GPUs. To have a server, one needs cooling, memory, a motherboard, a rack tower, CPUs, and OS memory. The mean electricity prices in the Netherlands in April 2025 were around €0, 26 kWh [52]. In Table 5.15, one can see the energy usage of several GPUs and their market price to buy one GPU (including VAT).

GPUs	kWh/day	Market Price (€)
RTX 4090	10.80	approx 1.779
A100 (40GB)	7.20	approx 11.000
A100 (80GB)	9.60	approx 17.850
H100 (94GB)	16.80	approx 34.800

Table 5.15: Energy usage per GPU and approximate market price in euros.

The H100 GPUs are more expensive than the A100 GPUs, but offer better performance for batch processing. Based on current market prices as of May 2025, Table 5.16 shows the estimated hardware costs for each model based on batch mode.

Model	Required Hardware	Approx Initial Cost (€)	Monthly Power Cost (€)	Annual Maintenance (€)
Llama 3.1-8B-Instruct	Server with 2× RTX 4090	10.000	169	1.000
Llama 3.1-8B-Instruct	Server with A100 (80GB)	25.000	75	2.500
Llama 3.1-8B-Instruct	Server with H100 (94GB)	40.000	131	4.000
Qwen3-8B	Server with 2x RTX 4090	10.000	169	1.000
Qwen3-8B	Server with A100 (80GB)	25.000	75	2.500
Qwen3-8B	Server with H100 (94GB)	40.000	131	4.000
Qwen2.5-32B-Instruct	Server with 2× A100 (80GB)	40.000	150	4.000
Qwen2.5-32B-Instruct	Server with 2× H100 (94GB)	75.000	262	7.500
Llama 3.3-70B-Instruct	Server with 3× A100 (80GB)	58.000	225	5800
Llama 3.3-70B-Instruct	Server with 3× H100 (94GB)	110.000	393	11.000
Llama 3.3-70B-Instruct	Server with 4× H100 (94GB)	145.000	524	14.000

Table 5.16: On-premises hardware costs (Power costs based on €0.26/kWh and 24/7 operation and 30 days within a month, maintenance estimated at 10% of hardware cost annually).

During this study, the configurations used on Snellius are shown in Table 5.17.

Model	Used GPUs in this study
Llama 3.1-8B-Instruct	One H100 (94GB)
Qwen3-8B	One H100 (94GB)
Qwen2.5-32B-Instruct	Two H100 (94GB)
Llama 3.3-70B-Instruct	Four H100 (94GB)

Table 5.17: The used GPU configurations for each model on Snellius in this study.

For the 70B and 32B parameter models in batch mode, the memory requirements require multiple high-end GPUs. One can also see that in the used GPU configurations, if multiple

GPUs are used, the number of GPUs is always even. Even GPUs split the work into perfectly even chunks and form a power-of-two NCCL ring, so each device spends less time idling on compute.

5.5.3. Cloud-Based Deployment Costs on Azure

For Azure-based deployment, the current pricing of GPU-accelerated VMs is analysed. All prices listed in Table 5.18 are for Azure [51] VMs as of May 2025.

Model	Azure VM Type	Monthly Costs Pay as you go (€)	Monthly Costs One year savings plan (€)
Llama 3.1-8B-Instruct	NC64as T4 v3	3.489, 94	2.556, 04
Llama 3.1-8B-Instruct	NC40ads H100 v5	5.825, 12	4.845, 91
Qwen3-8B	NC64as T4 v3	3.489, 94	2.556, 04
Qwen3-8B	NC40ads H100 v5	5.825, 12	4.845, 91
Qwen2.5-32B-Instruct	NC48ads A100 v4	6.126, 64	5.099, 81
Qwen2.5-32B-Instruct	NC80adis H100 v5	11.650, 24	9.691, 83
Llama 3.3-70B-Instruct	NC96ads A100 v4	12.253, 27	10.199, 63
Llama 3.3-70B-Instruct	ND96isr H100 v5	81.998, 14	- -

Table 5.18: Azure GPU VM costs (assuming 24/7 operation).

The pricing of cloud-based GPU resources, shown in Table 5.18, shows a substantial cost increase when moving to instances that are capable of running larger models, particularly in batch mode. The ND96isr H100 v5 from Azure consists of eight H100 GPUs, which means that it could run multiple LLMs at once.

5.5.4. Analysis for Londen & Van Holland

Current Workload Distribution
Londen & Van Holland processes approximately 200 annual reports yearly. Currently, each annual report requires approximately 8 hours of a manager's time to verify compliance, with managers costing €250 per hour. This translates to a total of 1600 hours, resulting in a total cost of €400.000.

The workload is not even distributed throughout the year. During the peak season, approximately 150 annual reports must be processed over four months, which is an average of 37.5 reports per month. These 37.5 reports per month translate into a workload of 300 hours a month for managers during the peak period. In contrast, the off-peak period spans eight months with only 50 reports to process, averaging 6.25 reports per month. This corresponds to a reduced workload of 50 hours per month.

Potential System Integration Scenarios
The integration of the ARAG system into Londen & Van Holland's operations could be facilitated through several deployment options. Based on the hardware requirements

presented in Table 5.14, and considering the performance metrics presented in Section 5.2, the following scenarios are considered: high-performance on-premise deployment, cost-efficient on-premise deployment and cloud-based alternatives.

On-Premise Deployment with Llama 3.3-70B-Instruct
The Llama 3.3-70B-Instruct model, which demonstrated the highest evaluation performance, would require substantial computing resources. This setup involves a server infrastructure with four H100 GPUs, with an estimated hardware investment of €145.000. The associated electricity cost is approximately €524, and the annual maintenance is approximately €14.000. The total allocation of resources for the first year is estimated at €159.524.

On-Premise Deployment with Qwen2.5-32B-Instruct
As a more resource-efficient alternative, the Qwen2.5-32B-Instruct model offers a balance between performance and costs. This configuration requires two A100 GPUs or two H100 GPUs, where the two H100 GPUs offer better performance for batch processing. Using the two H100 GPUs in this analysis, the estimated hardware costs are €75.000, the electricity costs are approximately €262, and the annual maintenance costs are €7.500. The total allocation of resources for the first year is estimated at €82.762.

Cloud-Based Depolyment Options
Deployment through cloud infrastructure offers great flexibility, but at a higher recurring cost. Hosting Llama 3.3-70B-Instruct on an NC96ads A100 v4 instance results in a monthly expense of €12.253, 27, or €147.039, 24 annually. In contrast, running Qwen2.5-32B-Instruct on the Azure NC48ads A100 v4 instance results in a monthly cost of €6.126, 64, totalling €73.519.68 a year. The use of Qwen2.5-32B-Instruct on the Azure NC80adis H100 v5 instance results in a monthly cost of €11.650, 24, totalling €139.802, 88 a year. Opting for a one-year savings plan on Azure can reduce these costs by approximately 16.8%.

Batch Processing Capabilities and Workflow Integration
The batch processing capability of the ARAG system is particularly valuable during the peak period, which could allow the overnight processing of compliance checks. The performance characteristics demonstrated suggest that with batch enabled, a configuration using the Llama 3.3-70B-Instruct model with four H100 GPUs could process around 53 questions per hour. This processing capacity could alter workflow patterns, particularly for routine compliance verification.

Resource Allocation Considerations
It is important to note that the integration of the ARAG system would never eliminate the need for professional oversight. Auditors still need to review all items flagged as non-compliant by the system, as well as conduct random checks on items marked compliant to ensure the reliability of the system. The confusion matrices presented in Section 5.2 indicate that even the best-performing model still produced false negatives 9 out of 25 truly compliant cases in the test set.

This means that time savings would not be directly proportional to the system's accuracy rate. A more realistic assessment of the ARAG system's value must account not only for its classification performance but also for the time implications in a real-world auditing workflow.

Specifically, time savings are achieved primarily on items correctly identified as compliant, which may require minimal or no further review. However, items flagged as non-compliant, whether correctly or incorrectly, will still need a monthly inspection by professionals. In addition, for the deployment of the ARAG system, time is needed for quality assurance and system validation.

The ARAG system can handle around 53 questions an hour. For an average annual report, around 300 questions from the checklist must be answered. That is, the system is on average 5.6 hours busy with an annual report. This means that the system can handle 4 annual reports per day and thus 120 annual reports per month (considering that a month has 30 days). This processing capacity would be sufficient to manage the entire annual workload of 200 reports, including the peak period demand of 37.5 reports per month.

If we consider a scenario where auditors would still need one to two hours to review the non-compliant items and perform random checks on compliant times. This results in a time reduction of approximately 75% for managers. Applied to 1600 hours annually, this could lead to a reduction of 1200 hours annually. The corresponding financial resources would range from €70.000 to €160.000 annually.

However, one must consider several practical implementations:

1. The time needed for quality assurance and system validation.
2. The adoption period for integrating the system into existing workflows.
3. Potential adjustments to the review process based on system performance.
4. Additional time for managing edge cases and complex compliance scenarios.

Taking into account these factors, a more conservative estimate of 50% time reduction could be more realistic for each manager in the initial phase. Taking this into account, annual costs would range from €270.000 to €360.000, resulting in a potential annual resource savings of €40.000 - €130.000.

The batch processing capabilities of the ARAG system would provide particular value during the peak period from February to May, potentially allowing more flexible resource allocation during these critical time frames. This could mean that during off-peak hours, one can choose sequential processing instead of batch processing.

5.6. Production Application

To use the ARAG system researched in this study, a first version of the application is built. This section outlines the key features and functionalities of the production implementation.

5.6.1. Implementation & Functionality

The application is implemented as a web-based user interface that facilitates the interaction between auditors and the underlying ARAG system. The primary interaction mechanism is a conversational interface through which auditors can communicate with “Krissie”, the AI assistant that leverages the knowledge of BW2T9 and the RJ. This interface allows auditors

to obtain clarification on specific accounting regulations without manual research through regulatory documents.

The application also allows auditors to upload annual reports in PDF format for automated compliance verification. The system processes these documents using the ARAG pipeline described in Section 4.2 and generates a compliance report. To ensure seamless integration with existing audit workflows, the system maintains comprehensive audit trails of all compliance checks performed. This includes logs of which items were verified automatically and which require professional intervention. In Figure 5.12, the user interface is shown.

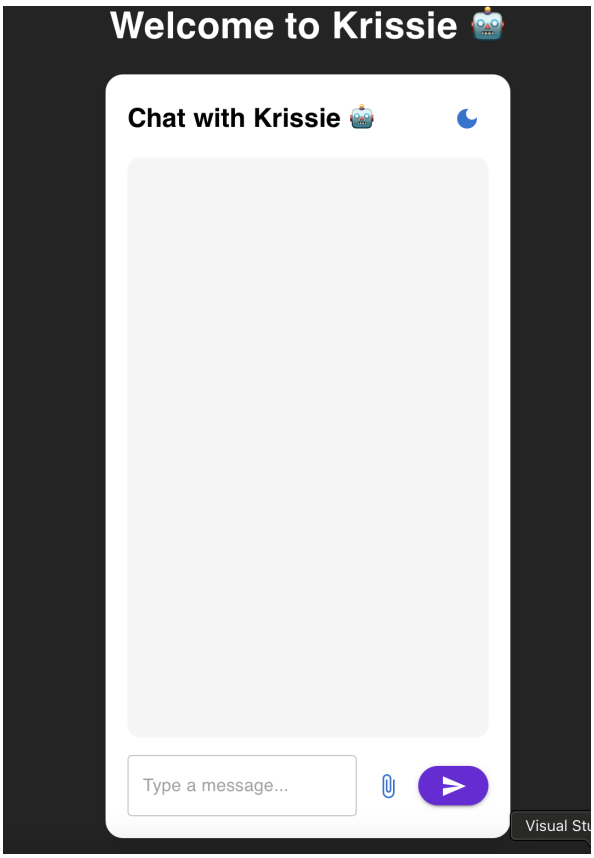


Figure 5.12: User interface Krissie.

5.6.2. Future Development Roadmap

To enhance the application, several additional optimisations can be considered. These include implementing session persistence for document retention throughout an audit engagement, developing customisable compliance checklists to adapt to different client scenarios, establishing integration with existing audit management software, and implementing performance analytics to support continuous system improvement. The production implementation represents the practical application of the research findings. Transforming the experimental ARAG system into an operational tool that improves audit efficiency while preserving the essential role of professional judgment in the compliance verification process.

6

Discussion

This chapter discusses the findings of the evaluation of the ARAG system for auditing Dutch annual reports. The discussion is organised into three main sections: main findings, limitations, and directions for future research.

6.1. Main Findings

6.1.1. Evaluation of the ARAG System

The evaluation of the ARAG system for auditing Dutch annual reports reveals several important insights.

Validation
The validation phase provides several insights into the different configurations of the ARAG system and the different performance characteristics of each model. The Llama 3.3-70B-Instruct model, which is the largest model evaluated in this study, shows outstanding performance, as shown in Table 5.7, across the validation set. It achieves a perfect precision score of 100% and maintains a high recall of 85%, resulting in the highest F_1-score of 0.92. The model performs the best according to Bayesian optimisation, on the validation set, with a relatively small chunk size of around 164 tokens and $k = 11$ or $k = 25$. What stands out with this model is that it maintains perfect precision across various configurations. This indicates that the model is robust in avoiding false positives. This is due to the prompts given to the agents, which explicitly tell the agents that whenever it is uncertain, they should be cautious, and thus label the question as non-compliant. This is a critical feature for compliance verification, where false positives can carry significant regulatory risks.

Despite the smaller size, the Qwen3-8B model performed surprisingly well, see Table 5.3. The model achieves an F_1-score of 0.81, a precision of 95%, a recall of 70%, and an accuracy of 85%. What stands out is that the model strongly prefers smaller chunk sizes, indicating that it processes information more effectively with a precise targeted context.

The Llama 3.1-8B-Instruct model shows the lowest performance among the models tested, see Table 5.1. Although it is the same size as Qwen3-8B, it achieves a lower F_1-score of 0.72. The model achieves an accuracy of 85%, a precision of 95%, and a recall of 70%. A

notable observation with this model is its sensitivity to hyperparameters, where even a small 5 token change in chunk size can dramatically alter the precision-recall balance. For example, the configuration with a chunk size of 768 tokens achieves the highest precision of 88% but a lower recall of 56%, while the best configuration found with a chunk size of 763 tokens better balances both metrics. This happens because the performance landscape contains steep-gradient regions where small parameter adjustments can substantially alter the system's behaviour. Because the validation set contains only 60 questions, each additional true positive answer significantly changes the precision and recall measured. In this case, the configuration with an F_1-score of 0.72 (chunk size = 763) has around 7 false positives, while the configuration with an F_1-score of 0.68 (chunk size = 768) has only 2 false positives, but has more false negatives. This sensitivity illustrates an important characteristic of retrieval parameters in RAG systems in general.

The Qwen2.5-32B-Instruct model also shows good performance on the validation set, see Table 5.5. Using this model results in an F_1-score of 0.88, accuracy of 90%, a precision of 96%, and a recall of 81%. The best parameters found for this model are quite stable, with the top five configurations, sorted by F_1 score. All chunk sizes among the top configurations are in the range 441-452, and all require a top-k of 14. This suggests that Qwen2.5-32B-Instruct has a relatively stable parameter space, based on the validation set, compared to the smaller models.

The model also achieved strong scores on LLM-based metrics within the validation phase, with a faithfulness score of 0.84, answer relevance of 0.75, and context relevance of 0.53. These validation metrics indicate that the model effectively uses the retrieved context and produces answers that align well with the original queries. This stability is also confirmed by Table 5.6, where the top two configurations by precision are also the top two configurations by F_1-score.

The validation results reveal a general correlation between model size and performance, with larger models demonstrating better overall metrics. In addition, the architectural differences between models of similar sizes, such as Qwen3-8B and Llama 3.1-8B-Instruct, lead to very different hyperparameter configurations and performance. This highlights that model architecture can be as important as model size in the performance of the ARAG system.

Testing and Comparison
The testing phase provides insight into the generalisation capabilities of the ARAG system using different language models.

The Llama 3.3-70B-Instruct model (k = 11, chunk size of 164 tokens and chunk overlap of 33 tokens), demonstrated the strongest overall performance, shown in Figure 5.5, achieving 73% accuracy, 89% precision, and 64% recall. This resulted in an F_1-score of 0.74 on the test set. This performance confirms that the largest model excels not only during validation but also when facing new unseen questions. The strong precision indicates that when this model identifies compliance, the model is highly certain, which is a crucial characteristic for audit applications where false positives carry significant risks.

The consistent strength of the LLM-based metrics for Llama 3.3-70B-Instruct indicates that the system's responses were not only accurate but also factually aligned with the retrieved context. This is essential for audit applications where explainability is as important as accuracy.

The smaller models showed a more varied performance. Qwen3-8B achieved the lowest overall performance on the test set with an F_1-score of 0.50, accuracy of 55%, precision of 82%, and recall of 36%. Despite this lower performance, the use of Qwen3-8B achieved the highest context relevance of 0.60 among all models. This suggests that while its classification decisions are often incorrect, the system still managed to retrieve the context that is relevant to the task.

Llama 3.1-8B-Instruct performed slightly better than Qwen3-8B, with an F_1-score of 0.58, accuracy of 60%, precision of 85%, and recall of 44%. The difference in performance between these models highlights the importance of model architecture and training methodology beyond just parameter count. Another reason could be the token difference. During testing, the best configurations, according to Bayesian optimisation, were used. This means that during the tests, Llama 3.1-8B-Instruct used a chunk size of 763 tokens, while Qwen3-8B used only 128 tokens. This significant difference could have potentially restricted Qwen3-8B's ability to access sufficient contextual information. Because Bayesian optimisation explores the search space with a limited budget, it can prematurely converge to a local optimum. If the surrogate model sees a slight drop in F_1-score when it tests chunk sizes just above the 128 tokens, the optimiser then assumes that the chunk sizes are bad and stops checking them. As a result, the 128 tokens are just the best configuration in that small area, not the best overall. Running the search with more evaluation steps, or a different seed, might uncover even better results.

Qwen2.5-32B-Instruct, with its medium-sized architecture, achieved an F_1-score of 0.62, accuracy of 63%, precision of 86%, and recall of 48%. This puts it between the smaller 8B models and the large 70B model in performance. This indicates a general correlation between model size and compliance verification capability. However, Qwen2.5-32B-Instruct received the lowest answer relevance score of 0.61 among all models. This suggests that while its classification accuracy is reasonable, its responses sometimes fail to directly address the specific compliance questions asked.

A particularly revealing pattern in the testing results is that all models correctly identified 13 out of 15 non-compliant items, regardless of size. Thus, all maintain relatively high precision, ranging from 82% to 89%, as can be seen in Figure 5.5. A closer examination of these results shows an interesting pattern in the error distribution. The two larger models, Llama 3.3-70B-Instruct and Qwen2.5-32B-Instruct, shared one common misclassified non-compliant item, while the two 8B models, Llama 3.1-8B-Instruct and Qwen3-8B, also shared a common error. Additionally, each model had one unique non-compliant item that it also misclassified. This pattern suggests that model size may influence the types of errors made, with models of similar scale exhibiting similar errors. The unique errors across all models further indicate that different model architectures have distinct blind spots, even when their overall performance metrics are comparable.

However, their ability to identify compliant cases varied significantly, with larger models demonstrating better recall, as can be seen in Figure 5.5. Using the Llama 3.3-70B-Instruct model, the ARAG system correctly identified 16 out of 25 compliant items, substantially outperforming medium and smaller models. Using Qwen2.5-32B-Instruct resulted in correctly identifying 12, while Llama 3.1-8B-Instruct identified 11, and Qwen3-8B identified only 9. The system's bias toward predicting non-compliant can probably be traced to the instructions

given in the prompts of the generation and refinement agents, as mentioned in the validation section in Section 6.1.1. Another potential factor is the class imbalance between test and validation splits. The validation set is almost evenly split (45% "yes" and 55% "no") while the test split has more yes answers (62.5% "yes" and 37.5% "no"). Because most of the test items are compliant, whenever the system says "yes", the prediction is very likely to be correct. This also explains the high precision scores. Since the models are tuned on a nearly balanced validation set, the models remain conservative and still give more "no" responses. This mismatch keeps precision high but reduces recall, so the F_1-score for compliance is lower.

The difference between detecting non-compliance and confirming compliance shows an important imbalance in verification tasks. Identifying non-compliance often requires finding a single contradiction, while confirming compliance demands full understanding of both the regulatory rules and the supporting evidence in the annual report. This explains why even the best model still missed 9 out of the 25 compliant cases.

When comparing performance between models, it is important to consider the impact of the relatively small size of the data set. With only 40 test questions, each question represents 2.5% of the test set, so even small variations in predictions can noticeably affect performance metrics. For example, getting two more questions correct could change the accuracy by 5 percentage points. This sensitivity suggests that small differences in performance metrics (such as the 4 percentage point difference in F_1-score between Qwen2.5-32B-Instruct and Llama 3.1-8B-Instruct) must be treated with caution, as they may reflect the test set's specific content more than actual model differences. However, larger performance gaps (such as the 24 percentage point difference in F_1-score between Llama 3.3-70B-Instruct and Qwen3-8B) are substantial enough to likely represent significant disparities in model performance. The consistent patterns observed on both the validation and the test sets, along with the human evaluation results, give more confidence in these findings despite the small size of the data set.

ARAG System vs. Non-RAG Baseline

The comparison, shown in Figure 5.7, reveals several important results for all models. First, except for Qwen3-8B, all models show substantial improvements in their F_1-score when using the ARAG system compared to the non-RAG baseline. The highest increase in performance is observed with Llama 3.3-70B-Instruct, which shows a 20% improvement in F_1-score, followed by Qwen2.5-32B-Instruct (14%) and Llama 3.1-8B-Instruct (13%).

A particularly notable observation is the consistent improvement in precision across all models when using the ARAG system. The greatest improvement appears with the Llama 3.1-8B-Instruct model, showing a 38% increase in precision. This suggests that the ARAG system is particularly effective in reducing false positives. For recall, the results vary more significantly across the models. Larger models show substantial improvements in recall, for example, Llama 3.3-70B-Instruct achieves a 23% improvement. In contrast, the smaller model Qwen3-8B experiences a decrease, with recall decreasing by 18%.

In particular, Qwen3-8B is the only model that shows a decrease in F_1-score with the ARAG system, resulting from the reduced recall. When examining the retrieval patterns, Qwen3-8B's best found configuration used a much smaller chunk size (128 tokens) compared to other models. This may have resulted in an overly fragmented context that lacked sufficient

information to identify compliant items. The model's improvement in precision also suggests that it adopted a more conservative decision-making approach when provided with retrieved context. This makes the model less willing to identify compliance unless it is very confident.

These results indicate that the ARAG system generally improves the model's performance. In addition, the ARAG system appears to lead to more reliable compliance verifications, particularly for models with stronger reasoning capabilities.

Human Evaluation
The human evaluation of the ARAG system with five professional auditors provides insights into its practical applicability. This evaluation is carried out on a complete annual report from a medium-sized Dutch company using the Llama 3.3-70B-Instruct model.

The ARAG system demonstrated strong performance across all five auditors, achieving an average F_1-score of 0.83, with individual scores ranging from 0.78 to 0.86, as shown in Figure 5.8.

The precision scores are particularly high. Two auditors (Auditor 1 and Auditor 3) evaluated the precision of the system as perfect, with a score of 1.00, with an average precision of 0.95 across all evaluators. This aligns with the design goal mentioned earlier that precision is the primary concern in financial compliance, since falsely approving non-compliant items carries greater regulatory risks than missing compliant ones. Human evaluation confirms that the ARAG system successfully prioritises precision in practice.

While precision is outstanding, recall shows more variability, ranging from 0.69 to 0.79, with an average of 0.74. This result is consistent with the observations in the automated evaluations, where the system shows stronger performance in identifying non-compliant items than in confirming compliance. This pattern extends across different evaluators, suggesting that it is an inherent characteristic of the system rather than an individual judgment.

The overall accuracy of 0.86 across all auditors demonstrates that the system can reliably handle a diverse range of compliance questions for Dutch annual reports. The Fleiss' kappa coefficient of 0.73 indicates substantial agreement, suggesting that the auditors were generally consistent in their evaluations of whether the annual report complied with the requirements specified in the questions. This statistic adds credibility to the performance metrics presented in Figure 5.8, as it shows that the auditors' judgments, which served as the ground truth for these metrics, were reliable.

The quality of the explanations and supporting evidence is rated on a scale of 1-5, with the results shown in Figure 5.9. The auditors' assessment of the justification quality showed that 70% of the ARAG system's responses were rated as "Excellent". This indicates that most of the explanations and supporting evidence provided by the system were of the highest quality according to professional audit standards. In total, 86% of the responses received ratings 3 or higher, indicating that the vast majority of the justifications of the ARAG system were considered at least adequate by the professional auditors.

The fact that almost 70% received the highest possible rating for justification quality is particularly noteworthy. This indicates that the system can connect its assessments to specific passages in the annual report and that its explanation meets certain standards. Such

explanations enhance transparency and allow auditors to verify the system's reasoning, which is essential for maintaining oversight and to provide a judgment.

However, the evaluation also revealed limitations, with 9% of the responses rated as "Very Poor" in terms of justification quality. These cases represent instances in which the system failed to identify the relevant passage in the annual report or misinterpreted the regulatory content.

An interesting observation from the human evaluation is the consistency with which different auditors evaluated the performance of the ARAG system. Although BW2T9 and the RJ are designed to be used uniformly, the Fleiss' kappa coefficient of 0.73 indicates "substantial agreement" and not "almost perfect". This suggests that even written regulations can be interpreted differently in practice.

6.1.2. Contribution of Individual Agents

The ablation experiments reveal important information about the importance and contributions of each agent within the ARAG system for compliance verification.

The combination of the `EvaluateContextAgent` and `ExpandRetrievalAgent` has the most significant influence on overall performance, as can be seen in Figure 5.10 and Figure 5.11. Removing these agents increased the precision from 89%to 93%, but dramatically reduced the recall from 64% to 52% on the test set. These agents also had the greatest impact on the validation set, leading to a decrease in performance. This illustrates a trade-off in compliance verification: when the system cannot evaluate the context and expand retrieval when necessary, the system becomes more conservative. Although this reduces the number of false positives, it also leads to many non-compliant cases being missed. These two agents form the information expansion of the ARAG system, which is particularly crucial for confirming compliance, since confirming compliance is a task that requires more complete and thorough evidence than identifying non-compliance.

The `SelectionAgent` contributes significantly to maintaining high precision. Its removal caused the largest decrease in precision on the test set, from 0.89 to 0.84, as can be observed in Figure 5.11. This highlights its importance in filtering irrelevant information that could lead to false positives. The different impact of this agent between the validation and test sets suggests that, as compliance questions become more diverse and challenging, the ability to filter out misleading information becomes more important.

The `ExpandQueryAgent` primarily enhanced recall with minimal effect on precision, demonstrating its role in broadening the query representation to capture various aspects of compliance requirements. Its modest impact suggests that this agent provides minor improvements rather than core functionality.

The `ValidationAgent` and `RefinementAgent` showed minimal impact on the validation set, as can be seen in Figure 5.10, but substantial impact on performance on the test set, reducing the F_1-score to 0.68, see Figure 5.11. This shows that self-correction becomes more valuable when compliance questions are more complex or unfamiliar. This is similar to how human auditors work, since more complex cases usually get extra attention and should be reviewed more carefully.

The results show that with all agents enabled, the complete ARAG system achieves the highest F_1-score on both the validation and the test sets. Although some individual ablations resulted in slightly higher precision, they consistently led to a notable decrease in recall and overall performance. This suggests that the combination of all agents, provided in this study, provides the most balanced configuration for the compliance verification of Dutch annual reports. The fact that removing multiple agents often has a greater impact than removing individual agents suggests that the agents work together and rely on each other to improve the system's performance. The agents talk to each other, creating an integrated system that is more effective than individual agents.

6.1.3. Cost-Benefit Considerations

The cost analysis given in Section 5.5 reveals important practical implications for implementing the ARAG system in a business environment.

On-premises deployment requires a substantial initial investment (€10.000 - €145.000 depending on the model choice), requires space in the company itself for servers and requires annual maintenance. Cloud-based alternatives through Azure provide flexibility without the need for a physical infrastructure, but there are significant recurring costs that can exceed €140.000 annually for high-performance configurations. For Londen & Van Holland, requiring 1600 manager hours annually at €250 per hour, even conservative estimates suggest a potential annual savings of €40.000 - €130.000 by using the ARAG system.

The ARAG system can process approximately 120 annual reports, in batch mode, per month. This provides sufficient throughput even during peak periods for Londen & Van Holland, who are processing on average 37 - 38 reports per month in this peak period (a total of 200 reports a year, of which 150 during the peak period of February-May). This has the potential to enable more efficient resource allocation throughout the year. However, actual returns will depend on several factors such as the quality for assurance that is needed and the system validation processes.

This theoretical cost analysis indicates that despite substantial upfront costs, the ARAG system can be a viable choice to help auditors check compliance with annual reports, with a reasonable payback period of 1-3 years depending on the choice of the model. Furthermore, the performance cost analysis suggests that medium-sized models like Qwen2.5-32B-Instruct may offer the most balanced value for many firms. These models provide high precision and are still very conservative without requiring the most expensive hardware configurations.

It is important to note that additional development costs should be anticipated for future system improvement. New and better LLMs continue to appear, so companies will need to budget for periodic updates. These updates will help maintain competitive performance and comply with the latest regulatory requirements. These ongoing development investments, which are not taken into account in this study, represent an important consideration in the long-term total costs of running an ARAG system.

6.2. Limitations

Despite the promising results demonstrated by the ARAG system, several limitations should be acknowledged.

First, the scope of this study was restricted by the manual effort required to convert the MKB checklist into a suitable format for the ARAG system. Of the complete checklist, only 100 questions were selected, reformulated and divided into validation (60) and test (40) sets. This reduction was necessary due to the labour-intensive process of manual reformulation with the help of compliance officers and accountants to create clear yes/no questions suitable for binary classification. This limitation also means that the ARAG system was evaluated on only a subset of the full compliance requirements that auditors must verify in practice, potentially missing edge cases, or more complex compliance scenarios. Furthermore, the class distribution in these sets was not perfectly balanced, with the validation set containing 45% "yes" answers and 55% "no" answers, while the test set had 62.5% "yes" answers and 37.5% "no" answers. This imbalance could potentially influence the model's learning during parameter optimisation and affect performance metrics.

Second, this study focused mainly on medium-sized Dutch companies, using a specific subset of related questions regarding one subject of compliance questions from the MKB checklist. Although this allowed for a controlled evaluation environment, the performance of the ARAG system can vary when applied to companies of different sizes with their respective regulatory requirements. Furthermore, only two real Dutch annual reports were used, one for the validation and testing phase and one for the human evaluation. This small sample limits the diversity of different report scenarios and can reduce confidence in how well the findings generalise. It was not possible to add more annual reports in this study due to the manual labour of annotating.

Third, while the LLMs used in this study are all multilingual, they were not specifically optimised for the Dutch language and even less so for Dutch law, which has a harder structure to read. These models may have varying proficiency levels across different languages, with a potential preference for English (Llama models) or Chinese (Qwen models) over Dutch. This could impact their performance when processing Dutch annual reports and regulatory texts like BW2T9 and the RJ. Although the prompts were structured appropriately for the Dutch language, some linguistic nuances or domain-specific terminology could have been interpreted differently than a native Dutch financial expert would. Furthermore, LLMs are rapidly evolving, and the models evaluated in this research represent a snapshot of the available technology as of early 2025. The performance difference between models may change as new versions are released. This could also potentially alter the cost-benefit calculations presented in Section 5.5.

Fourth, the human evaluation conducted in this study, while providing valuable insights, included only five auditors from a single firm. This means it could have some company bias in the results, a more diverse panel of experts from different firms could have led to different assessments.

Fifth, the chunking and retrieval methodologies used in this study only represent one approach among many possible configurations. Although Bayesian optimisation was used

to determine the optimal parameters, focusing on the chunk size and top-k values for retrieval, this approach can have limitations. The optimisation process may converge to a local optimum rather than a global optimum, and despite the 35 trials per model, a more effective configuration may lie outside the explored parameter space or have not been found due to the trials being limited to 35, as discussed in Section 6.1.1. In addition, other potentially influential parameters, such as the mixing weight in the hybrid retrieval ($\lambda = 0.5$) or the chunk overlap (20%), were fixed and not included in the optimisation process. This approach streamlined the experimentation process, but potentially overlooked more effective parameter combinations. Additionally, this research only focused on sentence-level chunking and hybrid retrieval, while other chunking methods or retrieval strategies might lead to improved results.

Sixth, to get the results, the study relied on the Snellius supercomputer, where System Billing Units (SBUs) were limited. This restricted the number of experiments that could be conducted, particularly for the larger models like Llama 3.3-70B-Instruct, which required substantial computational resources. Despite these limitations, the available resources were optimised by implementing batch processing. However, more extensive experimentation, such as exploring a wider range of hyperparameters or testing additional LLMs, would have been beneficial but was not feasible within the allocated resources.

Finally, this research focused primarily on the technical capabilities of the ARAG system rather than its integration into existing audit workflows. The actual savings in time and efficiency improvements in practical audit scenarios can differ from the theoretical projections, as factors such as auditor trust, the learning curve, and organisational change management were not fully evaluated.

6.3. Future Research

The limitations identified in Section 6.2 point to several promising directions for future research that could extend and maybe enhance the ARAG system for auditing Dutch annual reports.

First, expanding the scope of compliance questions would be valuable. Future work should aim to incorporate the complete MKB checklist into the ARAG system, including edge cases. This would provide a more comprehensive evaluation of the ARAG system. Additionally, developing automated methods for reformulating checklist items into clear yes/no questions could reduce the manual effort required and enable a more efficient scaling of the system. This would include making the questions clearer so that there is no ambiguity in what the questions attempt to ask.

Second, evaluating the ARAG system across a larger and more diverse set of annual reports from companies of different sizes (micro, small, and large) would provide insight into how the system's performance generalises across different regulatory contexts.

Third, investigating language-specific fine-tuning approaches could potentially improve performance for Dutch financial texts. Future research could explore the adaptation of multilingual models specifically for the Dutch financial and regulatory language, incorporating domain-specific terminology and linguistic patterns. A promising model is the GPT-NL model, which is likely to be released at the end of 2025 and is developed by the Netherlands

Organisation for Applied Scientific Research (TNO), the Netherlands Forensic Institute (NFI), and the ICT company SURF [53].

Fourth, a more extensive exploration of the hyperparameter space could provide improved configurations. Future work could investigate additional parameters beyond chunk size and top-k, such as various values for the hybrid retrieval mixing parameter λ and chunk overlap values. Optimised approaches or a larger number of trials might help identify global optimums. Furthermore, comparing alternative chunking strategies and retrieval methodologies could be interesting for the ARAG system.

Fifth, even more computational efficiency optimisations could make the system more accessible to smaller audit firms. Research into model quantisation, distillation, or pruning techniques could reduce the resource requirements while maintaining performance. This could enable a larger adoption of the ARAG system.

Sixth, a comprehensive study on the integration of the ARAG system into existing audit workflows would be valuable. This could include studies over time that track efficiency, how often mistakes occur, and user satisfaction. Learning how auditors use and trust the system in a practical setting can help improve how the system is designed and used.

Finally, using the ARAG framework in other languages and regulatory environments beyond the Netherlands could show how flexible and useful it really is. To do so, the system would need to be adjusted to work with different accounting rules (such as IFRS or GAAP) and legal systems. This could also help reveal common principles that apply to automated compliance checks everywhere.

7

Conclusion

This research investigated the effectiveness of an Agentic Retrieval-Augmented Generation (ARAG) system in assessing the compliance of Dutch annual reports with regulatory requirements.

The key findings demonstrate that the ARAG system is effective for compliance verification tasks. The Llama 3.3-70B-Instruct model achieved the best performance with an F_1-score of 0.74 on the test set, precision of 89%, and recall of 64%. When evaluated by human auditors on a different annual report, the system performed even better, with an average F_1-score of 0.83 and an accuracy of 86%.

In the context of financial compliance, precision is particularly crucial, as false positives (incorrectly marking a non-compliant item as compliant) carry greater regulatory risks than false negatives. This is because auditors will check the cases where the model says it is non-compliant. All models achieved relatively high precision, 82% to 89% on the test set and 73% to 100% on the validation set, indicating that the ARAG system is especially strong in avoiding false positives regardless of the model used. However, the size of the model has the most influence on recall, with the largest model of 70B parameters showing a substantially better ability to identify truly compliant items (64%) compared to smaller models (36% to 48%) in both the test and validation sets. This suggests that while even using smaller models can be conservative and precise in the ARAG system, larger models within the system understand a wider range of compliance situations.

The number of retrieved documents (top-k) and chunk size (S) have a big impact on compliance evaluation within the ARAG system and vary by model. For example, using Llama 3.3-70B-Instruct in the ARAG system works best with a moderate setting of $k = 11$ and a chunk size of 164 tokens, reaching a precision of 100% on the validation set. Retrieving more $k = 25$ did not improve the results as it slightly reduced the faithfulness, showing that more content can introduce noise. Smaller models need different settings, for example, Qwen3-8B performs best with fewer documents $k = 6$ and a very small chunk size of 128 tokens, while Qwen2.5-32B-Instruct performed best with $k = 14$ and a chunk size of 441 tokens. The importance of these parameters is supported by the fact that even a small change can have big effects, for example, by increasing the size of the chunks from 763 to 768 tokens for the Llama 3.1-8B-Instruct increased the precision from 73% to 88% but decreased the recall from

70% to 56% on the validation set.

Ablation studies revealed the critical role of multiple agents in the ARAG system. Removing agent groups, where agents talk with each other, has the greatest influence on the performance of the ARAG system. This study's findings demonstrate that all multi-agent system components contribute to the overall performance, with different agents addressing specific aspects of the task within the pipeline.

In conclusion, an ARAG system can effectively assess compliance in Dutch annual reports with high precision and interpretability. This approach enables auditors to focus on complex judgment tasks while automating routine compliance checks, allowing them to focus on more complex judgment tasks. Although using larger models provides better overall performance, even using smaller models offers valuable precision for compliance tasks. This provides options for organisations with different computational resources and accuracy requirements. Future research should focus on expanding the system to encompass the complete compliance checklist, adapting it for different company sizes, and further exploring fine-tuned Dutch LLMs to improve performance in compliance verification in Dutch annual reports.

References

[1] KVK, *Waaruit bestaat de jaarrekening?* https://www.kvk.nl/deponeren/waaruit-bestaat-de-jaarrekening/, Accessed: 25-03-2025.

[2] P. Lewis *et al.*, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021. arXiv: 2005.11401 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.11401.

[3] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, *Realm: Retrieval-augmented language model pre-training*, 2020. arXiv: 2002.08909 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2002.08909.

[4] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, *Agentic retrieval-augmented generation: A survey on agentic rag*, 2025. arXiv: 2501.09136 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2501.09136.

[5] X. Yao *et al.*, *Smart audit system empowered by llm*, 2024. arXiv: 2410.07677 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2410.07677.

[6] T. B. Brown *et al.*, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.

[7] G. Team *et al.*, *Gemini: A family of highly capable multimodal models*, 2024. arXiv: 2312.11805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2312.11805.

[8] H. Touvron *et al.*, *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2307.09288.

[9] H. He, H. Zhang, and D. Roth, *Rethinking with retrieval: Faithful large language model inference*, 2022. arXiv: 2301.00303 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2301.00303.

[10] X. Shen, Z. Chen, M. Backes, and Y. Zhang, *In chatgpt we trust? measuring and characterizing the reliability of chatgpt*, 2023. arXiv: 2304.08979 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2304.08979.

[11] Y. Zhang *et al.*, *Siren's song in the ai ocean: A survey on hallucination in large language models*, 2023. arXiv: 2309.01219 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2309.01219.

[12] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'24/IAAI'24/EAAI'24, AAAI Press, 2024, ISBN: 978-1-57735-887-9. DOI: 10.1609/aaai.v38i16.29728. [Online]. Available: https://doi.org/10.1609/aaai.v38i16.29728.

[13] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, *Retrieval augmentation reduces hallucination in conversation*, 2021. arXiv: 2104.07567 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2104.07567.

[14] I. Iaroshev, R. Pillai, L. Vaglietti, and T. Hanne, "Evaluating retrieval-augmented generation models for financial report question and answering," *Applied Sciences*, vol. 14, no. 20, p. 9318, 2024. DOI: 10.3390/app14209318.

[15] Y. Zhao *et al.*, "Optimizing LLM based retrieval augmented generation pipelines in the financial domain," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, Y. Yang, A. Davani, A. Sil, and A. Kumar, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 279–294. DOI: 10.18653/v1/2024.naacl-industry.23. [Online]. Available: https://aclanthology.org/2024.naacl-industry.23/.

[16] A. Q. Jiang *et al.*, *Mistral 7b*, 2023. arXiv: 2310.06825 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.06825.

[17] Qwen *et al.*, *Qwen2.5 technical report*, 2025. arXiv: 2412.15115 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2412.15115.

[18] S. Fatemi, Y. Hu, and M. Mousavi, "A comparative analysis of instruction fine-tuning large language models for financial text classification," *ACM Trans. Manage. Inf. Syst.*, vol. 16, no. 1, Feb. 2025, ISSN: 2158-656X. DOI: 10.1145/3706119. [Online]. Available: https://doi.org/10.1145/3706119.

[19] Y. Gao *et al.*, *Retrieval-augmented generation for large language models: A survey*, 2024. arXiv: 2312.10997 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2312.10997.

[20] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018, ISSN: 1935-8237. DOI: 10.1561/2200000071. [Online]. Available: http://dx.doi.org/10.1561/2200000071.

[21] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, *Ares: An automated evaluation framework for retrieval-augmented generation systems*, 2024. arXiv: 2311.09476 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2311.09476.

[22] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, *Ragas: Automated evaluation of retrieval augmented generation*, 2023. arXiv: 2309.15217 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2309.15217.

[23] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, *Evaluation of retrieval-augmented generation: A survey*, 2024. DOI: https://doi.org/10.1007/978-981-96-1024-2_8. arXiv: 2405.07437 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.07437.

[24] S. Kim, H. Song, H. Seo, and H. Kim, *Optimizing retrieval strategies for financial question answering documents in retrieval-augmented generation systems*, 2025. arXiv: 2503.15191 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2503.15191.

[25] B. Zhang, H. Yang, T. Zhou, M. Ali Babar, and X.-Y. Liu, "Enhancing financial sentiment analysis via retrieval augmented large language models," in *Proceedings of the Fourth ACM International Conference on AI in Finance*, ser. ICAIF '23, Brooklyn, NY, USA: Association for Computing Machinery, 2023, pp. 349–356, ISBN: 9798400702402. DOI: 10.1145/3604237.3626866. [Online]. Available: https://doi.org/10.1145/3604237.3626866.

[26] Logius Ministerie van Binnenlandse Zaken en Koninkrijkrelaties, *Burgerlijk Wetboek Boek 2, Titeldeel 9: De jaarrekening en het bestuursverslag*, PDF https://wetten.overheid.nl/BWBR0003045/2025-01-01, Accessed: 10-01-2025, Rijksoverheid.

[27] Artifex, *PyMuPDF: Python binding for mupdf*, version 1.25.4, Installed 01-04-2025, installed via PyPI. [Online]. Available: https://pymupdf.readthedocs.io/.

[28] Samuel Hoffstaetter, *pytesseract: Python–tesseract ocr wrapper*, version 0.3.13, Installed 01-03-2025, installed via PyPI. [Online]. Available: https://github.com/madmaze/pytesseract.

[29] Raad voor de Jaarverslaggeving, Ed., *Richtlijnen voor de jaarverslaggeving voor middelgrote en grote rechtspersonen*, Jaareditie 2024. Deventer: Wolters Kluwer, 2024, PDF file (generated 09-11-2023); applicable to reporting periods beginning on or after 1 January 2024, ISBN: 9789013170849.

[30] D. S. Team, "Docling technical report," Tech. Rep., version 1.0.0, Aug. 2024. DOI: 10.48550/arXiv.2408.09869. eprint: 2408.09869. [Online]. Available: https://arxiv.org/abs/2408.09869.

[31] SURF, *Snellius: de Nationale Supercomputer*, https://www.surf.nl/diensten/rekenen/snellius-de-nationale-supercomputer, Accessed 01-04-2025, 2023.

[32] LangGraph Agentic RAG Tutorial, *Langgraph agentic rag tutorial: Nodes and edges*, https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_agentic_rag/#nodes-and-edges, Accessed: 01-02-2025.

[33] L. Wang, N. Yang, and F. Wei, *Query2doc: Query expansion with large language models*, 2023. arXiv: 2303.07678 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2303.07678.

[34] A. Madaan *et al.*, "Self-refine: Iterative refinement with self-feedback," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23, New Orleans, LA, USA: Curran Associates Inc., 2023.

[35] X. Wang *et al.*, *Searching for best practices in retrieval-augmented generation*, 2024. arXiv: 2407.01219 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2407.01219.

[36] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, 2012. arXiv: 1206.2944 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1206.2944.

[37] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, *Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation*, 2024. arXiv: 2402.03216 [cs.CL].

[38] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," *arXiv preprint arXiv:2210.07316*, 2022. DOI: 10.48550/ARXIV.2210.07316. [Online]. Available: https://arxiv.org/abs/2210.07316.

[39] K. Enevoldsen *et al.*, "Mmteb: Massive multilingual text embedding benchmark," *arXiv preprint arXiv:2502.13595*, 2025. DOI: 10.48550/arXiv.2502.13595. [Online]. Available: https://arxiv.org/abs/2502.13595.

[40] M. Douze *et al.*, *The faiss library*, 2024. arXiv: 2401.08281 [cs.LG].

[41] PostgreSQL Global Development Group, *Postgresql: The world's most advanced open source relational database*, https://www.postgresql.org, version 16.6, Accessed: 01-01-2025 and installed 01-01-2025.

[42] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, Apr. 2009, ISSN: 1554-0669. DOI: 10.1561/1500000019. [Online]. Available: https://doi.org/10.1561/1500000019.

[43] Dorian Brown, *Rank-bm25: A two line search engine*, https://github.com/dorianbrown/rank_bm25, version 0.2.2, Accessed: 01-01-2025 and installed 01-01-2025.

[44] C. Li, Z. Liu, S. Xiao, and Y. Shao, *Making large language models a better foundation for dense retrieval*, 2023. arXiv: 2312.15503 [cs.CL].

[45] AI@Meta, *Llama 3 model card*, 2024. [Online]. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[46] Q. Team, *Qwen2.5: A party of foundation models*, Sep. 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/.

[47] Q. Team, *Qwen3*, Apr. 2025. [Online]. Available: https://qwenlm.github.io/blog/qwen3/.

[48] Scikit-Optimize Contributors, *Scikit-optimize: Sequential model-based optimization in python*, https://github.com/scikit-optimize/scikit-optimize, version 0.10.2, Accessed: 01-02-2025, 2021. [Online]. Available: https://scikit-optimize.github.io/.

[49] S. T. Gross, "The kappa coefficient of agreement for multiple observers when the number of subjects is small," *Biometrics*, vol. 42, no. 4, pp. 883–893, 1986, ISSN: 0006341X, 15410420. [Online]. Available: http://www.jstor.org/stable/2530702 (visited on 05/14/2025).

[50] OpenAI, *Gpt-4o-mini*, Used May 2025, 2024. [Online]. Available: https://platform.openai.com/docs/api-reference/introduction.

[51] Microsoft Corporation, *Linux Virtual Machines Pricing*, https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/#pricing, Accessed: 09-05-2025, 2025.

[52] ANWB. "Wat kost 1 kwh? stroomprijs per kwh." Accessed: 10-05-2025. (Apr. 2025), [Online]. Available: https://www.anwb.nl/energie/wat-kost-1-kwh.

[53] Digitale Overheid. "Nederland bouwt eigen open taalmodel gpt-nl." Accessed on 16-05-2025. (2023), [Online]. Available: https://www.digitaleoverheid.nl/nieuws/nederland-bouwt-eigen-open-taalmodel-gpt-nl/.

A

Appendix

A.1. Structure of the Data

```
{
    "title": article number,
    "text": content of the article,
    "metadata": {"source": BW2, "article": article number}
}
```

Figure A.1: Example of the structure of BW2T9.

```
{
    ''metadata": {
        ''title": ''RJ 2024",
    },
    ''chapters": [
        {
            ''title": "chapter here",
            ''sections": [
                {
                    ''title": ''Title here",
                    ''subsections": [
                    {
                            ''title": ''Title of subsection here.",
                            ''content": ''Content of subection here.",
                        },
                    ],
                    ''content": ''Content of section here",
                }
            ],
            ''content": ''content of chapter",
        }
    ]
}
```

Figure A.2: Example of the structured dictionary of the RJ.

```
[
  {
    "Checklist_text": "Text in first column",
    "Groot": 1 or 0, (based on if the specific row is for Big companies)
    "Midden": 1 or 0, (based on if the specific row is for Medium companies)
    "Klein": 1 or 0, (based on if the specific row is for small companies)
    "Bron": RJ, BW2 or null,
    "Bron_text": text of source or null,
    "row_type": "MAIN, algemeen or check",
    "combined_context": text of all the previous algemeen or null
    "subheader": subheader,
    "referenced_subheader": referenced subheader or null,
    "is_section_header": true or false
  },
]
```

Figure A.3: Example of the structured dictionary of the checklist.

A.2. Code for the Data

```
def extract_bw2(self) -> List[dict]:
    """Extracts articles from the PDF using regex."""
    doc = fitz.open(self.pdf_path)
    full_text = ""

    for page in doc:
        text = page.get_text("text")

        if not text.strip() and self.use_ocr:
            images = convert_from_path(
                self.pdf_path, first_page=page.number + 1, last_page=page.number + 1,
                    dpi=300
            )
            ocr_text = pytesseract.image_to_string(images[0], lang="nld")
            text = ocr_text.strip()

        full_text += text + "\n"

    #  Use regex to find "Artikel 360" etc.
    article_pattern = r"(Artikel \d+)"
    split_text = re.split(article_pattern, full_text)

    structured_docs = []
    for i in range(1, len(split_text), 2):
        article_number = split_text[i].strip()
        article_content = split_text[i + 1].strip() if i + 1 < len(split_text) else ""

        if article_number and article_content:
            structured_docs.append(
                {
                    "title": article_number,
                    "text": article_content,
                    "metadata": {"source": "BW2", "article": article_number},
                }
            )

    return structured_docs
```

Figure A.4: Code for processing BW2T9.

```
    def extract_rj(self):
        """
        Extract structured data from a PDF while preserving font sizes and bold styling.
        """
        structured_data = {"metadata": {"title": "RJ 2024"}, "chapters": []}
        current_chapter = None
        current_section = None
        current_subsection = None

        for page_number, page_layout in enumerate(
            tqdm(extract_pages(self.pdf_path), desc="Processing PDF Pages", unit="page")
        ):
            for element in page_layout:
                if isinstance(element, LTTextBox):
                    for text_line in element:
                        if isinstance(text_line, LTTextLine):
                            text = text_line.get_text().strip()
                            font_size = None
                            is_bold = False

                            for char in text_line:
                                if isinstance(char, LTChar):
                                    font_size = char.size  # Get font size
                                    is_bold = "Bold" in char.fontname or "Black" in char.
                                        fontname  # Check if bold

                            if not font_size:  # Skip if no font size found
                                continue

                            text_level = self.get_text_level(font_size, is_bold, text)

                            if text_level == "chapter":
                                if current_chapter:
                                    if current_section:
                                        if current_subsection:
                                            current_section["subsections"].append(
                                                current_subsection)
                                        current_chapter["sections"].append(current_section)
                                    structured_data["chapters"].append(current_chapter)
                                current_chapter = {"title": text, "sections": [], "content
                                    ": ""}
                                current_section = None
                                current_subsection = None

                            elif text_level == "section":
                                if current_section and current_chapter:
                                    if current_subsection:
                                        current_section["subsections"].append(
                                            current_subsection)
                                    current_chapter["sections"].append(current_section)
                                current_section = {"title": text, "subsections": [], "
                                    content": ""}
                                current_subsection = None

                            elif text_level == "subsection":
                                if current_subsection and current_section:
                                    current_section["subsections"].append(
                                        current_subsection)
                                current_subsection = {"title": text, "content": ""}

                            elif current_subsection:
                                current_subsection["content"] += f" {text}" if
                                    current_subsection["content"] else text
                            elif current_section:
                                current_section["content"] += f" {text}" if current_section
```

```
                                    ["content"] else text
                            elif current_chapter:
                                current_chapter["content"] += f" {text}" if current_chapter
                                    ["content"] else text

        # Append any remaining content
        if current_subsection and current_section:
            current_section["subsections"].append(current_subsection)
        if current_section and current_chapter:
            current_chapter["sections"].append(current_section)
        if current_chapter:
            structured_data["chapters"].append(current_chapter)

        return structured_data
```

Figure A.5: Code for processing RJ.

B
Appendix

B.1. Agent Prompts

B.1.1. Expand Query Agent

The `ExpandQueryAgent` uses an LLM to generate a passage p based on the query. The prompt used is as follows:

> Je bent een Nederlandse taalassistent. Schrijf altijd in het Nederlands. Beantwoord de vraag met een korte passage. Geef alleen de passage terug, zonder enige uitleg of context.
>
> `Vraag: query`

The LLM response is the passage p, using the methods of Wang et al. [33] an expanded query q^+ is created.

B.1.2. Selection Agent

The `SelectionAgent` selects the context the agent finds the most useful to answer the query. The agent has two separate tasks. It must identify the relevant sections of the annual report and the relevant regulatory content. To identify the relevant sections of the annual report, the following prompt is used:

> Je bent een expert in het controleren van jaarrekeningen.
> Je krijgt een set van documenten:
> JAARREKENING SECTIES: Deze documenten vormen de feitelijke jaarrekening, met financiële cijfers en toelichtingen.
> Jouw taak is als volgt:
> Analyseer de JAARREKENING SECTIES en selecteer uitsluitend de documenten waarvan de inhoud essentieel is om de vraag te beantwoorden. Geef hiervoor een lijst met indices (genummerd van 0 tot `len(annual_reports) - 1`), bijvoorbeeld (niet overnemen): [0,2,5].
> **Let op:** Het onderstaande voorbeeld is uitsluitend bedoeld als illustratie en mag niet letterlijk worden overgenomen in je antwoord!

Voorbeeld:
Query: Wat zijn de nieuwe waarderingsgrondslagen voor vaste activa?
JAARREKENING SECTIES:
Document #0: De vaste activa worden gewaardeerd op historische kostprijs...
Document #1: Er is een toelichting op de afschrijvingsmethoden...
...
Document #9: ...
Resultaat: [0,2,5]
Geef als resultaat uitsluitend de indices van de documenten die relevant zijn voor de vraag.

```
Vraag:  query
JAARREKENING SECTIES: annual report sections
Geef als resultaat uitsluitend een lijst met indices
```

This prompt gives back the indices of the annual report sections that are essential to answer the query. For regulatory content, a prompt of the same type is used.

B.1.3. Evaluate Context Agent

Evaluating the context is done by the `EvaluateContextAgent`. This agent is asked to validate the context that will be used to generate a response. If the agent finds it insufficient, it indicates the need for expanded retrieval. The agent uses the following prompt:

Je bent een expert in het beoordelen van documentkwaliteit voor Nederlandse accountants.
Bepaal of de onderstaande documenten en jaarrekening secties voldoende informatie bevatten om de vraag correct te beantwoorden.
Let op, het doel is om de jaarrekening te controleren aan de hand van de gestelde vraag, met de extra context (documenten).
Gebruik alleen de woorden 'VOLDOENDE' of 'ONVOLDOENDE'.

```
VRAAG: query
DOCUMENTEN: doc texts
JAARREKENING SECTIES: report texts
```
—ANTWOORD—
Antwoord uitsluitend met 'VOLDOENDE' of 'ONVOLDOENDE'.

The agent will return sufficient (VOLDOENDE) or insufficient (ONVOLDOENDE).

B.1.4. Expand Retrieval Agent

The `ExpandRetrievalAgent` is an LLM-based agent that interacts with the rule-based `RetrievalAgent` during its operation. It first alters the original query by generating three alternative query formulations using the following prompt:

> Je bent een expert in informatieretrieval binnen de context van Nederlandse accountancy en wet- en regelgeving. Je taak is om alternatieve zoektermen te genereren die kunnen helpen bij het vinden van relevantere documenten. Genereer 3 alternatieve zoektermen of formuleringen die kunnen helpen om relevantere documenten te vinden.
>
> `query: query`,
> Genereer alternatieve zoektermen, geef alleen 3 alternatieven zoektermen terug. Niks anders.

After the generation of the alternative queries, the Retrieval agent is called again to retrieve the relevant documents related to the alternative queries.

B.1.5. Generate Agent

The `GenerateAgent` answers the compliance question, based on the sections of the annual report. The regulatory documents are used to help understand the requirements underlying the question. The answer is generated using the following prompt:

> Je bent een expert in het assisteren van Nederlandse accountants en je heet Krissie.
> Je helpt accountants bij het controleren van jaarrekeningen aan de hand van Boek 2 Titel 9 (Burgelijk Wetboek 2) en de Richtlijnen voor jaarverslaggeving (RJ).
> Geef duidelijke antwoorden en wees volledig in het geven van je antwoord. Wees precies in je verwijzingen naar wet- en regelgeving, als deze relevant zijn.
> Hier onder vind je de INSTRUCTIES:
> 1. De JAARREKENING SECTIES bevatten delen uit de jaarrekening zelf. Alleen deze secties zijn de daadwerkelijke jaarrekening die je moet controleren op de vraag.
> 2. De EXTRA INFORMATIE UIT BRONNEN bevat wet- en regelgeving (BW2) en Richtlijnen voor de Jaarverslaggeving (RJ). Gebruik de EXTRA INFORMATIE alleen als beoordelingskader (regels/criteria), of om begrippen te verduidelijken als deze onduidelijk zijn.
> - Voor definities of uitleg van termen die in de JAARREKENING SECTIES niet zijn toegelicht, kun je de EXTRA INFORMATIE raadplegen.
> - Neem geen voorbeelden uit de EXTRA INFORMATIE over als onderbouwing van de inhoud uit de jaarrekening. De voorbeelden zijn uitsluitend illustratief.
> 3. Geef als antwoord 'Conclusie: Ja' als het antwoord op de vraag 'ja' is en antwoord 'Conclusie: Nee' als het antwoord op de vraag 'nee' is.
> 4. Als je het antwoord niet zeker weet geef dan als conclusie 'Conclusie: Nee'. Dit om False positieve te voorkomen.
> Hier onder vind je het FORMAT waar je antwoord in moet geven:
> Antwoord exact in het volgende format:
> 'Conclusie: Ja' of 'Conclusie: Nee'
> Toelichting: [maximaal 4 zinnen waarom je dit concludeert]
> Onderbouwing uit jaarrekening: [exacte passages uit JAARREKENING SECTIES, als het niet in de jaarrekening secties staat zeg dan 'Het staat niet in de

jaarrekening']
==========
VOORBEELD (NIET ONDERDEEL VAN DE JAARREKENING OF DE EXTRA INFORMATIE)
==========
Vraag (voorbeeld): Is er sprake van een juiste toelichting op de waarderingsgrondslagen voor vaste activa?
JAARREKENING SECTIES (voorbeeld):
'Paragraaf 4: Vaste activa'
De vaste activa worden gewaardeerd op historische kostprijs. Een jaarlijkse impairment test wordt uitgevoerd.
'Er zijn geen verdere toelichtingen opgenomen.'
EXTRA INFORMATIE (voorbeeld, niet officieel!):
BW2 Titel 9, Artikel 362, lid 1: 'De waardering van activa en passiva geschiedt op basis van betrouwbare en verifieerbare methodes.'
RJ 212.1: 'Voor materiële vaste activa dient een toelichting te worden gegeven op de toegepaste waarderingsgrondslagen, inclusief afschrijvingsmethode.'
Voorbeeldantwoord (in het gewenste format):
Conclusie: Nee
Toelichting: Er ontbreekt een toelichting op de afschrijvingsmethoden en de termijn. Hiermee is de informatie niet volledig.
Op grond van de regelgeving had dit vermeld moeten worden. Hierdoor schiet de toelichting tekort.
'Onderbouwing uit jaarrekening: Paragraaf 4: Vaste activa — Er zijn geen verdere toelichtingen opgenomen.'
LET OP: Dit is een fictief voorbeeld. Het maakt geen deel uit van de daadwerkelijke jaarrekening of de echte wet- en regelgeving.
Gebruik voor je uiteindelijke antwoord uitsluitend de echte JAARREKENING SECTIES en de informatie uit EXTRA INFORMATIE UIT BRONNEN die BW2 of RJ bevatten (als beoordelingskader).

Controleer of de onderstaande jaarrekening voldoet aan de gestelde vraag:
—-VRAAG: query
—-JAARREKENING SECTIES (Enkel deze secties vormen de feitelijke jaarrekening): retrieved reports
—-EXTRA INFORMATIE UIT BRONNEN (Bevat wet- en regelgeving uit BW2 en RJ. Gebruik dit uitsluitend ter interpretatie van de vraag; niet als bron voor de jaarrekening zelf): retrieved text

To ensure that the agent responds in the correct format, a set of examples is provided. This guidance helps the agent to follow the desired structure consistently.

B.1.6. Validation Agent

The `ValidationAgent` is responsible for verifying the quality of the generated response. The agent checks whether the answer is factually correct, complete, and directly addresses

the query, using only the selected sections of the annual report and supporting regulatory documents. The agent uses the following prompt:

> Je bent een expert in het controleren van Nederlandse jaarverslagen.
> Je beoordeelt of een gegeven antwoord klopt op basis van de bronnen en of het de originele vraag volledig beantwoordt.
> Wees kritisch en precies in je beoordeling, maar focus alleen op feitelijke juistheid en volledigheid.
>
> `Originele vraag: query`
> `Gegenereerd antwoord: response`
> `Relevante secties uit het jaarverslag: annual report text`
> Controleer het antwoord op basis van de volgende criteria:
> 1. Feitelijke juistheid: Komt het antwoord overeen met de inhoud van het jaarverslag?
> 2. Volledigheid: Beantwoordt het alle aspecten van de vraag?
> 3. Directe beantwoording: Geeft het een duidelijke conclusie (Ja/Nee)?
> 4. Als het antwoord 'Het staat niet in de jaarrekening' is, controleer dan of wat de vraag vraagt inderdaad niet in de jaarrekening staat.
> Geef je beoordeling in het volgende format:
> VALIDE: [true/false]
> REDEN: [beknopte uitleg waarom het wel/niet valide is]
> VERBETERPUNTEN: [specifieke punten die aangepast moeten worden indien niet valide]

If the answer is not valid, the `ValidationAgent` provides specific feedback for correction by the `RefinementAgent`.

B.1.7. Refinement Agent

The `RefinementAgent` improves responses that have been flagged as invalid by the `ValidationAgent`. Its task is to correct factual inaccuracies, complete missing information, and ensure that the answer follows the required structure. The following prompt is used:

> Je bent een expert in Nederlandse jaarverslagen.
> Je taak is om een eerder gegeven antwoord te verbeteren op basis van feedback.
> Gebruik de relevante secties uit het jaarverslag (JAARREKENING SECTIES) als primaire bron voor je verbeteringen.
> Zorg dat je antwoord volledig, feitelijk correct is en direct begint met een conclusie (Ja/Nee).
> Als je het antwoord niet zeker weet geef dan als conclusie 'Conclusie: Nee'.
> Het format van het antwoord moet zijn:
> 'Conclusie: Ja' of 'Conclusie: Nee'
> Toelichting: [maximaal 4 zinnen waarom je dit concludeert]
> Onderbouwing uit jaarrekening: [exacte passages uit JAARREKENING SECTIES, als het niet in de jaarrekening secties staat zeg dan 'Het staat niet in de

jaarrekening']

```
Originele vraag: query
Eerder gegenereerd antwoord met problemen: response
Feedback op het antwoord: feedback
Relevante secties uit het jaarverslag (JAARREKENING SECTIES):
annual report text
Extra context, dit is bedoeld om je te helpen bij het begrijpen van
de vraag en het jaarverslag (dit is de wetgeving): extra context
history context
```

Verbeter het antwoord zodat het:
1. Feitelijk correct is en overeenkomt met de bronnen (JAARREKENING SECTIES)
2. De vraag volledig beantwoordt is
3. Begint met een duidelijke conclusie (Ja/Nee)
4. Onderbouwd wordt met relevante passages uit het jaarverslag
5. Als je het antwoord niet zeker weet, geef dan als conclusie 'Conclusie: Nee'.
Dit om False positieve te voorkomen.
Geef alleen het verbeterde antwoord zonder uitleg over het verbeterproces.

B.2. LLM-based Metrics

B.2.1. Faithfulness

An answer a is faithful to the context c if the claims made in the answer can be inferred from the context [22]. This is done using a two-step process. In the first step, the LLM is asked to extract sentences (statements L) using the answer a, using the following prompt:

Gegeven een vraag en een antwoord, maak één of meer stellingen van elke zin in het gegeven antwoord.

```
vraag: question
antwoord: answer
```

In the next step, for each statement $l \in L$, the LLM is asked to determine if the statement is supported by the context (Yes or No), making it a binary verdict. This is done using the following prompt:

Bekijk de gegeven jaarrekening, context en de stellingen. Bepaal of de stellingen worden ondersteund door de informatie in de jaarrekening en context (gebruik de context alleen voor wetgeving). Geef een oordeel (Ja/Nee) of de stelling ondersteund wordt door de jaarrekening en de context (gebruik de context alleen voor wetgeving). Wijk niet af van het gespecificeerde formaat.

```
jaarrekening: annual_report_sections

context: context
```

> `Stellingen: chr(10).join([f"stelling: s" for s in statements])`
>
> Het formaat waarin je moet aantwoorden is:
> stelling 1: Ja/Nee
> stelling 2: Ja/Nee
> ...

Once all verdicts V have been obtained, the faithfulness is calculated by:

$$\text{Faithfulness} = \frac{|V|}{|L|}.$$

B.2.2. Answer Relevance

An answer a is relevant if it directly addresses the question in an appropriate way [22]. To determine the relevance of the answer, the LLM is prompted to generate questions $\hat{q}$ from the answer a using the prompt:

> Genereer N vragen voor het gegeven antwoord. Geef alleen de vragen terug, zonder uitleg of context. Elk antwoord moet op een nieuwe regel beginnen met "Vraag: ".
>
> `antwoord: answer`
> Voorbeeld (let op dit is een voorbeeld, geen echte vragen), de vragen beginnen elk op een nieuwe regel:
> Vraag: ....
> Vraag: .....

Using the prompt, the LLM generates N questions. The relevance is calculated using the cosine similarity function $\text{sim}_{\cos}$ which takes the embeddings E of the original question q and the generated questions $\hat{q}$:

$$\text{Answer Relevance} = \frac{1}{N} \sum_{i=1}^{N} \text{sim}_{\cos} \left(E(q), E(\hat{q}) \right).$$

B.2.3. Context Relevance

The context c is relevant to the extent that it contains the information necessary to answer the question [22]. This can be achieved by prompting the LLM to extract the relevant sentences from the question q and context c. This can be done using the prompt:

> Beoordeel of elk van deze zinnen relevante informatie bevat voor het beantwoorden van de vraag.
> Antwoord voor elke zin alleen met "Ja" of "Nee".
> `Vraag: question`
> `Zinnen: sentences`

> Antwoord als JSON-array met n elementen van "Ja" of "Nee". Geef geen uitleg of extra context. Geef alleen de array terug. Bijvoorbeeld:
> `["Ja","Nee","Ja", ...]`
> ...

The relevance of the context is then calculated with:

$$\text{Context Relecance} = \frac{\text{number of relevant sentences in context}}{\text{number of total sentences in the context}}.$$

C Appendix

C.1. Results Hyperparameter Tuning

Config	Acc	Prec	Rec	F_1	Faithful	AnsRel	CtxRel
k=21, cs=292, co=58	0.6500	0.6364	0.5185	0.5714	0.8335	0.7260	0.4403
k=21, cs=663, co=133	0.6500	0.6154	0.5926	0.6038	0.8035	0.7209	0.5282
k=14, cs=218, co=44	0.6500	0.6250	0.5556	0.5882	0.8276	0.7269	0.4375
k=14, cs=427, co=85	0.6833	0.6538	0.6296	0.6415	0.8098	0.7328	0.4910
k=8, cs=711, co=142	0.6500	0.6875	0.4074	0.5116	0.7993	0.7158	0.4705
k=6, cs=775, co=155	0.7500	0.7727	0.6296	0.6939	0.8000	0.7239	0.5285
k=24, cs=129, co=26	0.5500	0.5000	0.4815	0.4906	0.7729	0.7258	0.4139
k=25, cs=681, co=136	0.6333	0.5926	0.5926	0.5926	0.7945	0.7270	0.5762
k=6, cs=782, co=156	0.7167	0.7273	0.5926	0.6531	0.8110	0.7256	0.5327
k=9, cs=768, co=154	0.7667	0.8824	0.5556	0.6818	0.8347	0.7388	0.5305
k=25, cs=758, co=152	0.6333	0.6000	0.5556	0.5769	0.7868	0.7286	0.6025
k=5, cs=770, co=154	0.7333	0.7895	0.5556	0.6522	0.8256	0.7271	0.5485
k=14, cs=785, co=157	0.6833	0.6667	0.5926	0.6275	0.8112	0.7161	0.5512
k=9, cs=460, co=92	0.6833	0.6818	0.5556	0.6122	0.8291	0.7214	0.5461
k=5, cs=395, co=79	0.6667	0.6842	0.4815	0.5652	0.7711	0.7171	0.4414
k=25, cs=437, co=87	0.7000	0.8235	0.5185	0.6364	0.8353	0.7269	0.5255
k=5, cs=835, co=167	0.6500	0.6364	0.5185	0.5714	0.7978	0.7189	0.5407
k=5, cs=759, co=152	0.6833	0.7000	0.5185	0.5957	0.8606	0.7145	0.4841
k=24, cs=495, co=99	0.6500	0.6364	0.5185	0.5714	0.8265	0.7230	0.5382
k=19, cs=775, co=155	0.6500	0.6500	0.4815	0.5532	0.7952	0.7247	0.5874
k=8, cs=775, co=155	0.6333	0.6316	0.4444	0.5217	0.8033	0.7211	0.5014
k=9, cs=970, co=194	0.6667	0.6522	0.5556	0.6000	0.7715	0.7141	0.5141
k=6, cs=598, co=120	0.6167	0.5909	0.4815	0.5306	0.7768	0.7200	0.4677
k=6, cs=772, co=154	0.7667	0.8095	0.6296	0.7083	0.8358	0.7234	0.5019
k=6, cs=765, co=153	0.7167	0.7778	0.5185	0.6222	0.8112	0.7194	0.4796
k=9, cs=763, co=153	0.7500	0.7308	0.7037	0.7170	0.8355	0.7234	0.5017
k=9, cs=760, co=152	0.7500	0.7500	0.6667	0.7059	0.8182	0.7352	0.5288
k=17, cs=366, co=73	0.6333	0.6190	0.4815	0.5417	0.8136	0.7282	0.4823
k=10, cs=810, co=162	0.7000	0.7143	0.5556	0.6250	0.8086	0.7100	0.5360
k=8, cs=630, co=126	0.6833	0.6333	0.7037	0.6667	0.7727	0.7254	0.4959
k=8, cs=619, co=124	0.6500	0.6071	0.6296	0.6182	0.7806	0.7213	0.4553
k=9, cs=756, co=151	0.6500	0.6364	0.5185	0.5714	0.8210	0.7220	0.5016
k=9, cs=766, co=153	0.7333	0.8235	0.5185	0.6364	0.8462	0.7344	0.4775
k=19, cs=1014, co=203	0.6500	0.6250	0.5556	0.5882	0.8016	0.7271	0.5332
k=17, cs=414, co=83	0.6833	0.7143	0.5556	0.6250	0.8106	0.7222	0.5175

Table C.1: Validation Results for Different Parameter Combinations for model: Llama 3.1-8B-Instruct.

Config	Acc	Prec	Rec	F_1	Faithful	AnsRel	CtxRel
k=21, cs=292, co=58	0.7333	0.8667	0.4815	0.6190	0.8298	0.7391	0.4725
k=21, cs=663, co=133	0.6333	0.7273	0.2963	0.4211	0.8458	0.7396	0.6717
k=14, cs=218, co=44	0.6667	0.7059	0.4444	0.5455	0.8332	0.7293	0.4141
k=14, cs=427, co=85	0.7000	0.7647	0.4815	0.5909	0.8588	0.7387	0.5480
k=8, cs=711, co=142	0.6500	0.6875	0.4074	0.5116	0.8335	0.7363	0.4873
k=6, cs=775, co=155	0.7167	0.9167	0.4074	0.5641	0.8610	0.7441	0.5256
k=24, cs=129, co=26	0.6667	0.7692	0.3704	0.5000	0.8328	0.7382	0.4517
k=25, cs=681, co=136	0.7500	0.9286	0.4815	0.6341	0.8758	0.7340	0.6687
k=25, cs=690, co=138	0.6667	0.7333	0.4074	0.5238	0.8421	0.7426	0.6670
k=25, cs=672, co=134	0.6833	0.7000	0.5185	0.5957	0.8396	0.7420	0.6633
k=25, cs=677, co=135	0.7167	0.8125	0.4815	0.6047	0.8449	0.7288	0.6761
k=21, cs=284, co=57	0.6833	0.7500	0.4444	0.5581	0.8383	0.7338	0.5173
k=21, cs=298, co=60	0.7333	0.7895	0.5556	0.6522	0.8101	0.7450	0.5523
k=21, cs=306, co=61	0.7667	0.8095	0.6296	0.7083	0.8502	0.7277	0.5433
k=21, cs=313, co=63	0.7500	0.8750	0.5185	0.6512	0.8532	0.7426	0.4955
k=21, cs=305, co=61	0.7500	0.8000	0.5926	0.6809	0.8538	0.7315	0.5264
k=22, cs=306, co=61	0.7000	0.8000	0.4444	0.5714	0.8133	0.7396	0.5392
k=24, cs=676, co=135	0.6833	0.7222	0.4815	0.5778	0.8506	0.7349	0.6866
k=14, cs=641, co=128	0.6833	0.7500	0.4444	0.5581	0.8188	0.7473	0.5384
k=21, cs=325, co=65	0.7500	0.8750	0.5185	0.6512	0.8274	0.7485	0.4922
k=21, cs=340, co=68	0.7667	0.8421	0.5926	0.6957	0.8401	0.7449	0.5561
k=21, cs=339, co=68	0.7667	0.8095	0.6296	0.7083	0.8455	0.7390	0.5490
k=21, cs=347, co=69	0.7667	0.9333	0.5185	0.6667	0.8473	0.7423	0.5592
k=21, cs=337, co=67	0.7167	0.8571	0.4444	0.5854	0.8374	0.7466	0.5708
k=19, cs=365, co=73	0.6833	0.7857	0.4074	0.5366	0.8608	0.7426	0.5289
k=21, cs=316, co=63	0.7333	0.8235	0.5185	0.6364	0.8509	0.7396	0.5295
k=21, cs=349, co=70	0.7000	0.7368	0.5185	0.6087	0.8313	0.7368	0.6071
k=21, cs=327, co=65	0.6667	0.6842	0.4815	0.5652	0.8420	0.7489	0.5674
k=25, cs=183, co=37	0.7000	0.8000	0.4444	0.5714	0.8524	0.7511	0.5249
k=21, cs=375, co=75	0.7667	0.9333	0.5185	0.6667	0.8577	0.7403	0.5353
k=5, cs=131, co=26	0.7500	0.8750	0.5185	0.6512	0.8779	0.7457	0.4113
k=6, cs=128, co=26	0.8500	0.9500	0.7037	0.8085	0.8926	0.7470	0.3638
k=7, cs=128, co=26	0.7333	0.8667	0.4815	0.6190	0.8882	0.7561	0.3995
k=6, cs=192, co=38	0.7833	0.8500	0.6296	0.7234	0.8917	0.7362	0.4021
k=17, cs=414, co=83	0.7500	0.9286	0.4815	0.6341	0.8689	0.7464	0.5203

Table C.2: Validation Results for Different Parameter Combinations for model: Qwen3-8B.

Config	Acc	Prec	Rec	F_1	Faithful	AnsRel	CtxRel
k=21, cs=292, co=58	0.8000	0.9412	0.5926	0.7273	0.8259	0.7636	0.4311
k=21, cs=663, co=133	0.7833	0.9375	0.5556	0.6977	0.7994	0.7497	0.4889
k=14, cs=218, co=44	0.7833	0.8889	0.5926	0.7111	0.8481	0.7482	0.4367
k=14, cs=427, co=85	0.8500	0.9500	0.7037	0.8085	0.8138	0.7437	0.4833
k=8, cs=711, co=142	0.8167	0.9000	0.6667	0.7660	0.8456	0.7548	0.5582
k=6, cs=775, co=155	0.8167	0.9444	0.6296	0.7556	0.8660	0.7506	0.5414
k=24, cs=129, co=26	0.8167	0.9000	0.6667	0.7660	0.8680	0.7466	0.4468
k=25, cs=681, co=136	0.7833	0.8889	0.5926	0.7111	0.8098	0.7499	0.4445
k=12, cs=445, co=89	0.7833	0.8889	0.5926	0.7111	0.8051	0.7503	0.5133
k=14, cs=441, co=88	0.9000	0.9565	0.8148	0.8800	0.8408	0.7484	0.5276
k=14, cs=449, co=90	0.8833	0.9545	0.7778	0.8571	0.8462	0.7512	0.4850
k=14, cs=471, co=94	0.8000	0.9412	0.5926	0.7273	0.8805	0.7520	0.4969
k=14, cs=442, co=88	0.8333	0.9474	0.6667	0.7826	0.8286	0.7499	0.5007
k=14, cs=408, co=82	0.8000	0.8947	0.6296	0.7391	0.8900	0.7497	0.4829
k=14, cs=452, co=90	0.8667	0.9130	0.7778	0.8400	0.8558	0.7426	0.5470
k=14, cs=444, co=89	0.8500	0.8750	0.7778	0.8235	0.8243	0.7397	0.4922
k=15, cs=443, co=89	0.8000	0.8571	0.6667	0.7500	0.8542	0.7470	0.5010
k=14, cs=445, co=89	0.8500	0.9091	0.7407	0.8163	0.8400	0.7461	0.5175
k=13, cs=439, co=88	0.8333	0.9474	0.6667	0.7826	0.8328	0.7520	0.5227
k=14, cs=446, co=89	0.8667	0.9130	0.7778	0.8400	0.8276	0.7388	0.4990
k=13, cs=406, co=81	0.8000	0.8947	0.6296	0.7391	0.8297	0.7495	0.4967
k=13, cs=470, co=94	0.8167	0.9000	0.6667	0.7660	0.8342	0.7513	0.5074
k=12, cs=911, co=182	0.8000	0.9412	0.5926	0.7273	0.8652	0.7443	0.4810
k=18, cs=879, co=176	0.8333	0.9474	0.6667	0.7826	0.8289	0.7443	0.4564
k=18, cs=913, co=183	0.8000	0.8947	0.6296	0.7391	0.8319	0.7560	0.4944
k=14, cs=447, co=89	0.8667	0.9130	0.7778	0.8400	0.8600	0.7412	0.5177
k=18, cs=850, co=170	0.8000	0.9412	0.5926	0.7273	0.8394	0.7506	0.5026
k=18, cs=131, co=26	0.8333	0.9474	0.6667	0.7826	0.8273	0.7456	0.4504
k=18, cs=160, co=32	0.8167	0.9000	0.6667	0.7660	0.8199	0.7469	0.4200
k=17, cs=128, co=26	0.8333	0.9474	0.6667	0.7826	0.8756	0.7512	0.4502
k=17, cs=156, co=31	0.8000	0.8947	0.6296	0.7391	0.8160	0.7493	0.4400
k=19, cs=129, co=26	0.8167	0.8636	0.7037	0.7755	0.8147	0.7474	0.4601
k=19, cs=163, co=33	0.8333	0.9048	0.7037	0.7917	0.8326	0.7472	0.4278
k=19, cs=186, co=37	0.8167	0.8636	0.7037	0.7755	0.8133	0.7514	0.4249
k=20, cs=163, co=33	0.8000	0.8947	0.6296	0.7391	0.8142	0.7500	0.4206

Table C.3: Validation Results for Different Parameter Combinations for model: Qwen2.5-32B-Instruct.

Config	Acc	Prec	Rec	F_1	Faithful	AnsRel	CtxRel
k=21, cs=292, co=58	0.8667	0.9130	0.7778	0.8400	0.8206	0.7415	0.4736
k=21, cs=663, co=133	0.8167	0.9444	0.6296	0.7556	0.8769	0.7475	0.4959
k=14, cs=218, co=44	0.8833	0.9545	0.7778	0.8571	0.8270	0.7487	0.4302
k=14, cs=427, co=85	0.8167	1.0000	0.5926	0.7442	0.8371	0.7460	0.4721
k=8, cs=711, co=142	0.8500	0.9500	0.7037	0.8085	0.9058	0.7479	0.4621
k=6, cs=775, co=155	0.8500	0.9500	0.7037	0.8085	0.8901	0.7527	0.5251
k=24, cs=129, co=26	0.9167	0.9583	0.8519	0.9020	0.9085	0.7499	0.4264
k=25, cs=681, co=136	0.8833	0.9545	0.7778	0.8571	0.8122	0.7494	0.4316
k=11, cs=128, co=26	0.9167	1.0000	0.8148	0.8980	0.8629	0.7525	0.3958
k=5, cs=139, co=28	0.8333	0.9048	0.7037	0.7917	0.9138	0.7466	0.4197
k=25, cs=691, co=138	0.8667	0.9130	0.7778	0.8400	0.8284	0.7517	0.4637
k=14, cs=231, co=46	0.8833	0.9545	0.7778	0.8571	0.8558	0.7509	0.4490
k=25, cs=199, co=40	0.9167	1.0000	0.8148	0.8980	0.8619	0.7451	0.4162
k=25, cs=128, co=26	0.9167	0.9583	0.8519	0.9020	0.8576	0.7529	0.4352
k=17, cs=128, co=26	0.9167	1.0000	0.8148	0.8980	0.8543	0.7541	0.4460
k=8, cs=189, co=38	0.8833	1.0000	0.7407	0.8511	0.8382	0.7512	0.4087
k=6, cs=206, co=41	0.8500	0.9091	0.7407	0.8163	0.8804	0.7480	0.3954
k=25, cs=1024, co=205	0.8500	0.9500	0.7037	0.8085	0.7532	0.7508	0.5188
k=14, cs=128, co=26	0.9000	1.0000	0.7778	0.8750	0.8753	0.7543	0.4448
k=20, cs=128, co=26	0.9167	1.0000	0.8148	0.8980	0.8765	0.7458	0.3971
k=11, cs=1024, co=205	0.8667	0.9130	0.7778	0.8400	0.8480	0.7474	0.4994
k=25, cs=408, co=82	0.8000	0.8571	0.6667	0.7500	0.8783	0.7473	0.5059
k=22, cs=171, co=34	0.8667	0.9524	0.7407	0.8333	0.8723	0.7464	0.4268
k=25, cs=195, co=39	0.9167	0.9583	0.8519	0.9020	0.8470	0.7428	0.4422
k=22, cs=250, co=50	0.8667	1.0000	0.7037	0.8261	0.8815	0.7536	0.4420
k=25, cs=163, co=33	0.9333	1.0000	0.8519	0.9200	0.8313	0.7471	0.4441
k=25, cs=610, co=122	0.8167	0.9444	0.6296	0.7556	0.8517	0.7423	0.4930
k=11, cs=164, co=33	0.9333	1.0000	0.8519	0.9200	0.8624	0.7530	0.4267
k=16, cs=895, co=179	0.9000	1.0000	0.7778	0.8750	0.8437	0.7417	0.5023
k=11, cs=901, co=180	0.9000	0.9565	0.8148	0.8800	0.8329	0.7371	0.5362
k=12, cs=161, co=32	0.9167	1.0000	0.8148	0.8980	0.8512	0.7471	0.4305
k=10, cs=157, co=31	0.9167	1.0000	0.8148	0.8980	0.8569	0.7441	0.4409
k=11, cs=187, co=37	0.9167	0.9583	0.8519	0.9020	0.8548	0.7460	0.4106
k=13, cs=848, co=170	0.8667	0.9524	0.7407	0.8333	0.8837	0.7423	0.4980
k=7, cs=929, co=186	0.9167	1.0000	0.8148	0.8980	0.8809	0.7407	0.5013

Table C.4: Validation Results for Different Parameter Combinations for model: Llama-3.3-70B-Instruct.

C.2. Results Comparision with Baseline

Model	Metric	ARAG	Non-RAG	Improvement (%)
Llama 3.1-8B-Instruct	F_1-score	0.579	0.512	13.2
	Precision	0.846	0.611	38.5
	Recall	0.440	0.440	0.0
	Accuracy	0.600	0.475	26.3
Qwen3-8B	F_1-score	0.500	0.537	-6.8
	Precision	0.818	0.688	19.0
	Recall	0.360	0.440	-18.2
	Accuracy	0.550	0.525	4.8
Qwen2.5-32B-Instruct	F_1-score	0.615	0.541	13.9
	Precision	0.857	0.833	2.9
	Recall	0.480	0.400	20.0
	Accuracy	0.625	0.575	8.7
Llama 3.3-70B-Instruct	F_1-score	0.744	0.619	20.2
	Precision	0.889	0.765	16.2
	Recall	0.640	0.520	23.1
	Accuracy	0.725	0.600	20.8

Table C.5: Comparison of ARAG vs Non-RAG performance across all models. Improvement is calculated as the percentage increase from Non-RAG to ARAG performance.