CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Insight generation using language models in enterprise settings

From retrieval to recursion: a comparative enterprise case study

Master's thesis in Computer science and engineering

Filip Appelblad
Ludvig Nordberg

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Insight generation using language models in enterprise settings

From retrieval to recursion: a comparative enterprise case study

Filip Appelblad
Ludvig Nordberg

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Insight generation using language models in enterprise settings
From retrieval to recursion: a comparative enterprise case study
Filip Appelblad
Ludvig Nordberg

Supervisor: Shirin Tavara, CSE
Advisor: Ernan Bektic, Essity
Examiner: Peter Damaschke, CSE

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2026

Insight generation using language models in enterprise settings
From retrieval to recursion: a comparative enterprise case study
Filip Appelblad
Ludvig Nordberg
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Recent improvements in generative AI, in particular Large Language Models (LLM) have created a desire for organizations to utilize the technology to drive value creation. One promising application of the technology is insight generation from internal enterprise data. Despite rapid technological advancements, organizations face substantial technical, organizational, and environmental challenges when trying to implement such systems. The purpose of this thesis is to investigate insight generation in enterprise settings, and which key aspects and challenges that influence a successful implementation.

The study is conducted as a single-case design research project within a large enterprise environment. Based on exploratory findings and requirement analysis, two language model based pipelines are designed, implemented, and evaluated: a Retrieval-Augmented Generation (RAG) pipeline with Chain-of-Thought prompting, and a Recursive Language Model (RLM) inspired pipeline, using iterative reasoning and content retrieval. Both pipelines are integrated into a chat-based prototype that allows users to query a large internal corpus, retrieving results with clear provenance.

The pipelines are evaluated using a combination of automated metrics and qualitative expert assessments. The results indicate that while both approaches are capable of generating insights from internal enterprise data, the RLM pipeline consistently produces more relevant, and insightful responses, while also being more trustworthy. However, this increased performance comes at the cost of increased computational overhead in the form of latency and token usage. Beyond model accuracy, the findings suggest that successful enterprise adoption depends equally on non-technical factors, including trust, transparency, organizational constraints, and organizational readiness. These factors have to be addressed as requirements and implemented into the solutions artifact to ensure a successful insight generation system.

Keywords: Large Language Models (LLMs), Recursive Language Models (RLM), Retrieval-augmented generation (RAG), AI Assistant, Insight Generation, Generative AI, Enterprise AI.

Acknowledgements

We would like to thank our supervisor at Chalmers, Shirin Tavara, for her guidance and steady support throughout this project. We are also grateful to our examiner, Peter Damaschke, for his oversight and feedback.

At Essity, we want to thank our industry supervisor, Ernan Bektic, for his close collaboration and engagement with the project at every stage. We are equally grateful to our manager, Jan Wirén, for his input to the thesis and for helping shape its direction.

Finally, we thank everyone at Essity who took time out of their schedules to share their expertise, sit through interviews, and contribute to the evaluation. This thesis would not have been possible without their generosity.

Filip Appelblad & Ludvig Nordberg, Gothenburg, 2026-05-06

Contents

List of Figures

List of Tables

1

Introduction

1.1 Research Context and Motivation

Organizations across major industries are investing heavily in artificial intelligence (AI) to extract value from internal data and generate business insights. Despite these initiatives, many companies struggle to translate AI investments into measurable business value. A survey by Boston Consulting Group (BCG) from fall 2024 indicated that nearly three-quarters of organizations had yet to demonstrate substantial value from their AI projects [1]. A more recent BCG survey from early 2026 suggests that the picture is changing: corporations expect to double AI spending in 2026, and four out of five CEOs are more optimistic about the return on investment of AI initiatives than a year earlier [2]. Simultaneously, the pressure to convert investments into measurable value has increased, about half of CEOs in the survey stated that their own job security depends on AI initiative outcomes [2]. Therefore, the gap between ambition and realized value persists as a challenge. Machucho and Ortiz [3] attribute this gap to both technical and organizational factors, which, individually and in combination, create a gap between the potential of their data and the value that the organization is able to realize.

Even when organizations recognize the potential of their internal data, they face obstacles originating from data heterogeneity and quality issues. Enterprise environments usually contain diverse data sources, including structured databases, unstructured text documents, and internal repositories such as SharePoint. A large majority of this information is often unstructured, estimated to be 80-90% of enterprise data. These vast, unstructured repositories hold immense potential value if analyzed effectively [4]. Recent improvements in Large Language Models (LLMs) have demonstrated strong potential to address some of these challenges. LLMs are able to interpret and summarize large sets of structured and unstructured text data. This progress enables insight extraction that was previously difficult or even impossible to implement [5].

In this thesis, insights refer to actionable knowledge acquired from synthesizing information across internal documents. Examples include identifying recurring themes, surfacing lessons learned that are transferable to new situations, or revealing connections between work conducted in separate teams or time periods. Such insights often go beyond what a single source states explicitly. However, applying these models naively within enterprises in the hope of generating insights remains problematic.

A major limitation arises from context-window constraints, showing degrading performance with lengthy inputs, especially when information is placed in the middle of long contexts. The limitation prevents the LLMs from processing all organizational data at once [6]. This has led to the development of strategies for efficient information grounding, where the model is provided with relevant pieces of internal knowledge [7]. Another limitation arises from reasoning shortcomings. Even LLMs grounded with relevant information struggle with multi-step reasoning. This ability is essential for uncovering deeper insights and answering complex questions, which are common in real use cases [8]. To illustrate the shortcomings, an individual seeking to understand how reporting requirements have evolved over time would need a model to process a large set of reports simultaneously, exceeding typical context limits. Similarly, identifying recurring failure patterns across reporting would require a model to retrieve information from one document, connect it with another, and draw an intermediate conclusion, a form of multi-step reasoning that standard LLM-prompting does not reliably support.

A wide range of methods has been developed to mitigate these shortcomings. One of the most widely deployed approaches is Retrieval-Augmented Generation (RAG), which enables the model to retrieve relevant internal documents before producing an answer [9]. To enhance reasoning, different techniques that guide models through structured intermediate steps have been developed. One major category of methods is Chain-of-Thought (CoT) prompting. These methods work by encouraging the model to break down complex problems into smaller reasoning steps [10]. CoT-style approaches have been shown to improve accuracy on multi-step tasks and help models with multi step information retrieval [8]. Recently, different fundamental approaches have been developed as a complement and alternative to classic solutions. One of these frameworks is the Recursive Language Model (RLM) [11], which addresses information grounding and reasoning in a different way, allowing iterative retrieval and reading. These hybrid techniques represent strategies to unlock deeper insights from data. They remain relatively unexplored in organizational settings, but since their design suggests they might offer advantages for enterprise insight-generation tasks, they represent an emerging direction of high relevance for organizations seeking more capable AI systems.

1.2 Problem Statement

Enterprises need a way to generate insights from large collections of internal documents and data in a way that is performance-wise consistent, trusted, and operationally robust. The core challenge is enabling language models to utilize internal knowledge, despite context size limits, to reason around, and synthesize information for users. Another core challenge is ensuring the solution is viable in a real enterprise context, and is accepted by stakeholders.

1.3 Aim and Research Questions

The aim of this thesis is to explore how organizations can successfully extract valuable insights from internal data using language model-based methods, by addressing both the technical challenges and the organizational considerations involved. In particular, the project aims at developing different model pipelines that address the fundamental challenges of naive language model approaches, notably limited context and reasoning ability, and on assessing how these solutions can fit within an enterprise setting in an effective manner.

Concretized, the thesis project involves designing and implementing two different models, one incorporating RAG and CoT prompting, and the other a Recursive Language Model, and comparing their performance in insight generation from internal documents. Two forms of evaluation will be employed: an expert qualitative evaluation and an automated quantitative evaluation. The study also seeks to identify transferable insights about the enabling factors and conditions that influence the successful application of similar AI methods across different enterprise settings. These factors include how to address ethical and governance issues.

To clarify the scope and objectives, the following research questions (RQs) are posed:

- **Main RQ**: How can enterprise organizations leverage language model techniques to generate insights from internal data, and what key technical aspects and organizational challenges are essential for the successful implementation of such a system?
- **RQ1**: What model architecture yields the most accurate and insightful results on the internal unstructured document corpus? Specifically, how do Retrieval-Augmented Generation with Chain-of-Thought prompting and Recursive Language Model approaches compare in their ability to answer queries grounded in the organization's data?
- **RQ2**: What are the key factors and challenges for successfully implementing an insight-generation system in an enterprise environment?

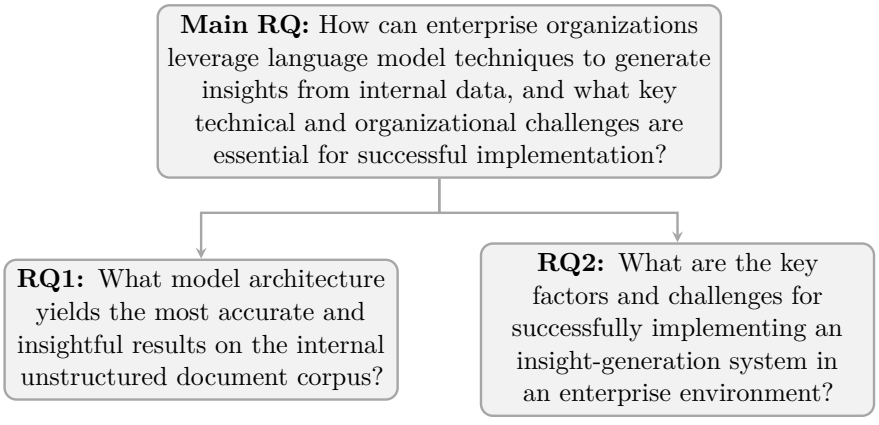

Figure 1.1: Visualization of the research questions and its relations.

For confidentiality, any specific insights or sensitive content from Essity's data will not appear in the final report, instead the thesis will document methods, structure, illustrative non-sensitive examples, and aggregated evaluation results. By focusing on methodology and lessons learned, the results are intended to be more generalizable.

1.4 Delimitations

No fine-tuning or training of LLMs will be performed on the company data. Fine-tuning is computationally expensive and introduces compliance concerns. The study instead relies on pre-existing LLMs and adapts them via prompting and retrieval. The model chosen is Claude Sonnet 4.5, a frontier model approved for use in this enterprise setting [12]. The model is a hybrid-reasoning model that enables built-in CoT reasoning. However, for increased transparency and control over reasoning, this feature will not be used in the project. The motivation for selecting Sonnet 4.5 over higher capacity alternatives such as Opus is due to practical deployment trade-offs, with respect to cost and latency. The choice is also motivated by the high improvement rate in frontier model performance, which suggests that any marginal performance gap between current top-tier models is likely to become obsolete within a relatively short time span [13]. The contribution is thereby connected to the overall system design and not the specific model used.

The second delimitation is that the study will focus on data from a single, specific domain. As the scope is limited to a single company, the insights are therefore limited to what exists within that domain. Findings and best practices may not be directly transferable to unrelated domains. Evaluation of insight quality is performed with Essity's domain experts, which introduces subjectivity and context-specific criteria.

These limitations clarify the boundaries of this thesis. The choices enables a focus on the core research questions within a feasible scope. Risk is also mitigated by not venturing into further, nearby areas, such as the fine-tuning of large language models. Understanding these limitations is important when interpreting the results.

2

Background and Analytical Framework

This chapter presents the theoretical foundations that support the thesis. The chapter starts with a connection to knowledge management literature to build an understanding of the concept of insight generation. Then, it reviews the technical components: LLMs and their context constraints, RAG, CoT prompting, and RLMs. The chapter then concludes with the Technology-Organization-Environment framework, which provides the analytical perspective for examining enterprise implementation factors.

2.1 Enterprise Insight Generation and Knowledge Management

The challenge of generating valuable insights from internal data stored in the organization is connected to knowledge management (KM) literature. Knowledge management is an important process in organizations that covers the identification, coordination, and distribution of information. Inadequate KM can lead to increased costs for an enterprise, as the knowledge seeking process is prolonged, resulting in opportunity costs when time could have been spent on more valuable tasks [14]. KM research has long been clear that organizational performance depends not only on the knowledge that exists within an organization at a given time but also on its ability to create and apply knowledge in an effective manner [15].

As data volumes have grown rapidly, the field has turned to technological solutions, and knowledge management may serve as a necessary step between data and decision making, turning insights into value. AI has been recognized as a central enabler of a more dynamic knowledge flow, automating information retrieval in addition to dynamic insight and content generation. Generative AI tools have begun to shift knowledge processes from static to dynamic capabilities [16].

Despite the clear trend, the transition to LLM-powered insight generation in enterprise settings is not straightforward. A systematic literature review by Karakurt & Akbulut [17], analyzing 63 studies on the integration of RAG and LLM combined systems for enterprise challenges, found that enterprise adoption remains mostly in an experimental phase. The review identified a real gap between academic settings

and real-world production-ready deployment. Important metrics such as business value, user satisfaction, process efficiency, and compliance are often left out. The finding ties back to the observation that many organizations that invest in AI fail to deliver measurable returns [1].

Furthermore, Anderson et al. [18] carry forward and argue that while LLMs show potential for unstructured document analytics, there is often a need for 'sweep and harvest' processes in such analyses. A system will both have to sweep through large corpora and perform operations. They believe that RAG-setups are fundamentally limited, since it relies on single retrieval passes, and instead propose multi-step query plans as the solution.

Finally, literature emphasizes that the deployment of AI for enterprise insight generation is not purely a technical task. Successful integration of AI systems into knowledge management requires technology choices tailored to the organizational context, committed leadership that actively supports initiatives, and flexible governance frameworks capable of adapting to change [16].

2.2 Large Language Models and Context Constraints

Large Language Models (LLMs) are pretrained on vast corpora and have demonstrated impressive abilities to interpret queries and generate coherent text. In their weights, they can both store and recall plenty of knowledge, and fine-tuned LLMs have achieved excellent results on downstream NLP tasks [9]. This makes them an attractive choice for insight generation.

However, LLMs face a fundamental limitation, namely, fixed context windows. Each model can only successfully process input up to a certain length before experiencing problems. Even though modern LLMs offer lengthy context windows comprising millions of tokens [19], enterprise data often consists of much more. As a result, vanilla LLMs cannot simply be fed 'all data' at once; they will inevitably miss information unless guided. Furthermore, a phenomenon called 'context rot', has been observed, meaning that model performance tends to degrade as input length grows, even within the maximum context windows [19]. In essence, important details and information in very long inputs may be forgotten by the LLMs.

The context limitation leads to an information grounding problem: how is it possible to ensure that the LLM has relevant facts from internal knowledge? It is known that problems related to relevant context, provenance provision, and updating world knowledge exist [9]. Furthermore, LLMs are prone to hallucinations, generating outputs that sound plausible but are not grounded in truth [20].

In summary, while LLMs are powerful, the problem faced is that a naive application of an LLM to an organization's data will be limited by context constraints, yielding untrustworthy answers.

2.3 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a framework that directly addresses context limitations [9] by fetching relevant domain documents from a database given the query and then supplying them as context to the model [21]. The model consists of the retriever and a generator that conditions on the query and retrieved evidence to provide an answer [9]. The hybrid architecture combines the model's parametric memory with non-parametric memory in the form of external documents. By conditioning generation on retrieved documents, RAG-models ground their outputs in factual context, improving the accuracy and transparency of responses [22]. Furthermore, the design is modular, allowing updates to the external knowledge source without retraining the language model, letting the system stay up-to-date [9]. The ability to use specialized corpora also enables domain-specific applications [22].

One challenge with RAG-models is choosing the right chunks to retrieve, ensuring relevant information is passed without overwhelming the model with too much irrelevant text. Another challenge is that the answer will only be as good as the retrieved context, meaning that if something crucial is not retrieved, the model lacks evidence to produce a grounded answer. Mitigation strategies exist, such as hybrid retrieval [22], which can enhance factual consistency, but the dependency on a single retrieval pass remains a structural limitation of the architecture. In conclusion, RAG provides a scalable way to apply LLMs to large corpora and has been implemented in many different applications [22].

2.4 Multi-Step Reasoning and Chain-of-Thought Prompting

Many insights of interest in an enterprise context require multi-step reasoning. Humans tackle such problems by breaking them down, gathering information from one source, then another, make intermediate conclusions, formulate sub-questions, and so on, before reaching a conclusion. Standard LLM-prompting, does not guarantee that models will perform sequential reasoning. In fact, LLMs often skip straight to an answer, which can lead to incorrect answers in complex settings [23].

There are a couple of prompting setups that affect multi-step reasoning quality, allowing LLMs to be guided through prompting without performing any training updates.

- Zero-shot prompting: instruction only, no examples. Provides the model with a natural language-type description [24].
- One-shot prompting: the instruction plus a single example that demonstrates the desired format [24].
- Few-shot prompting: the instruction and a few examples [24].

2.4.1 Chain-of-Thought Methods

Chain-of-Thought (CoT) prompting is a form of prompting in which the model is encouraged to generate explicit intermediate steps before concluding with a final answer. The key idea is to follow the human method and break down the problem into smaller subproblems. CoT nudges the model to decompose the problem at hand. Wei et al. [25] show that CoT-prompting can substantially improve performance by utilizing few-shot prompting, but with examples that include step-by-step reasoning.

The simplest form is called zero-shot CoT, where a simple instruction can trigger step-by-step reasoning without providing any examples of reasoning. Kojima et al. [26] report large benefits compared to standard zero-shot prompting with this approach.

Although zero-shot CoT shows promising results in comparison to simple zero-shot prompting, even better performance is reported from few-shot CoT, where examples of step-by-step reasoning are provided to the model [26]. Such demonstrative examples consist of a question paired with an associated reasoning chain, which is composed of a rationale detailing the intermediate reasoning steps followed by a final answer. However, a practical limitation with few-shot CoT is that such demonstrations must be manually crafted, called Manual-CoT [27]. The manual crafting is non-trivial in nature, labor-intensive, and task-specific.

Zhang et al. [27] address this problem with Auto-CoT, an automatic method that partitions a set of queries into clusters via k-means [28] over sentence embeddings. Then, for each cluster a small set of representative queries is selected guided by considering them in order of shortest distance to the cluster centroid. Reasoning chains are generated for these queries using zero-shot CoT until the desired number of demonstrations is obtained [27]. The clustering step is critical as it produces a diverse demonstration set without any manual authoring, reducing the risk that a single erroneous chain dominates the entire setup.

Shum et al. [29] extend the idea of Auto-CoT with Automate-CoT, which adds a pruning quality gate that removes chains with a final answer that disagree with ground-truth labels. Subsequently, surviving chains are optimized through a policy-gradient procedure. The method have achieved gains over both Manual-CoT and Auto-CoT baselines, but its reliance on a labeled gold data set limits the applicability in domains where ground-truth annotation is expensive, difficult, or even unattainable.

A related technique that addresses the quality of reasoning chains without requiring labels is self-consistency, introduced by Wang et al. [30]. Rather than relying on a single sampled reasoning chain, the method generates multiple independent reasoning chains for the same query and selects the modal answer. The intuition behind choosing the most frequent answer is that correct reasoning paths are more likely to converge on the same answer, making majority agreement a good proxy for chain quality even in absence of ground-truth labels.

2.5 Recursive Language Models

Recursive Language Models (RLM) is a framework that allows the model to engage in iterative retrieval and reasoning, effectively conducting a multi-step dialogue with the data itself. The central concept of RLM, introduced by Zhang et al. [11], is to separate the reasoning steps from the context length constraints by treating the document corpus as an external environment the model is able to query and navigate programmatically.

RLMs can produce outputs that include instructions for themselves. One example could be 'Search all documents for information about Project A', which the system then executes, retrieves some documents, and feeds those back into the model for the next round. If the system wants to explore interesting documents more deeply, it can 'spawn' sub-instances with a limited scope and specific sub-tasks. This means that the model recursively calls itself on smaller chunks of data and propagates the result of the sub-instance to the main iterative loop. The loop continues until the model determines that it has enough information to answer the query [11]. This can be seen as similar to how a human researcher would perform iterative research: asking a broad initial question, finding some results, and asking follow-up questions about interesting subtopics.

In theory, the RLM approach should offer several advantages for insight generation:

- A dramatic increase in effective context size. Instead of being limited to a fixed number of tokens at once, the model can use information far beyond that by utilizing successive steps. Zhang et al. demonstrated RLMs that handled inputs two orders of magnitude larger than the base model's context window with good performance [11].
- Enabling dynamic multi-step reasoning. The model is not forced to use only a single retrieval pass of some top-k chunks. After retrieving information, it can decide what additional information is needed [11]. The recursive setup should allow the model to combine pieces of information in a more sophisticated way than, for instance, a vanilla RAG-setup that only sees disjoint chunks.
- Provenance benefits that stem from a kind of algorithmic transparency. As the RLM breaks down problems into subproblems by recursively finding more information and asking sub-queries, a trace of what operations it performed is created [11]. The idea is similar to CoT, but at a higher level involving document retrieval actions as well. The trace can be logged and reviewed, which is useful both for the model to induce, track, and update its plans and also for users to understand how insight was generated [31].

However, RLM comes with challenges. It can be more computationally intensive, with multiple LLM calls and retrieval rounds. Ensuring that the recursion converges (for instance, no infinite loops) is also a problem, but one solution could be to limit the maximum depth. Challenges aside, RLMs are promising in the process of mitigating some of the LLM's limitations, allowing the model to read and reason over data of unbounded size in a controlled manner [11].

2.6 Comparison of Architectures

The two model architectures described in Sections 2.2 to 2.5 address the same underlying problem: limited context and insufficient reasoning. However they differ fundamentally in their approaches, as shown in Table 2.1.

Table 2.1: Theoretical comparison of RAG with CoT and RLM architectures

Dimension	**RAG with CoT**	**RLM**
Retrieval strategy	Single-pass retrieval of top-k chunks [9]	Iterative retrieval driven by the model itself [11]
Reasoning approach	Structured decomposition within a single generation call with CoT-prompting [25]	Recursive approach where each step can trigger new retrieval and reasoning [11]
Effective context	Bounded by chunk size and number of retrieved chunks [9]	Extended across recursive steps, theoretically unbounded [11]
Provenance	Traced to the fixed set of retrieved chunks	Traced through a log of retrieval actions and subqueries
Computational cost	Lower: one retrieval pass, one LLM call	Higher: multiple LLM calls and retrieval rounds per query

2.7 Enterprise Context and Implementation Challenges

Implementing an insight generation system in an enterprise is not only a technical deployment, it also covers a socio-technical change process that must align with the enterprise context. A framework used to examine innovation adoption has been widely used in recent AI adoption studies, namely Technology-Organization-Environment (TOE) [32]. The framework has been utilized to capture the multi-dimensional nature of enterprise readiness [33]. Following the usage, to break down the dimensions further, different constructs have been suggested, and two of these are highlighted below in Table 2.2. Hirtranusi et al. [33] suggests twelve key factors based on a systematic literature review, while Hughes et al. [34] conducted an empirical study to find a more focused set of five key factors.

Broadly speaking, the technological context refers to the internal technological capabilities and resources that support AI deployment, such as data quality, infrastructure, and the availability of AI tools. Organizational context encompasses internal factors in the organization that affect adoption, such as leadership support, culture, human competencies, and trust. Environmental context covers the external environment, including market and industry pressure, and the regulatory and ethical environment [33].

Table 2.2: Two suggested Technology-Organization-Environment constructs

Dimension	**Hirtranusi et al., 2026 [33]**	**Hughes et al., 2026 [34]**
Technology	System capability Data and analytics readiness Technology usability Innovation capability	Complexity Relative advantage
Organization	Strategic and managerial support Human and structural readiness Cultural and governance alignment Knowledge and learning	Staff skills Change capacity
Environment	Regulatory and policy context Market dynamics Collaborative ecosystem Social legitimacy and ethics	Regulatory environment (also labelled regulatory challenges)

Notably, Hirtranusi et al. [33] emphasize that the TOE framework's three contexts are not independent silos but interdependent factors, meaning that the individual contexts, the following constructs, and the interdependencies should be analyzed.

In this thesis, the TOE framework provides a theoretical lens for examining the implementation of the insight generation system. It offers a structured way to organize the organizational findings from the thesis, mapping the empirical observations to TOE constructs and thereby systematically identifying important factors when implementing an insight system. The usage also ensures that all relevant influences are considered rather than focusing narrowly on the technology.

3

Methods

This chapter describes the research design and the methods that are utilized to answer the research questions. It begins by discussing the case study context and the overarching Design Science Research approach. Thereafter, it explains the exploratory interview study, the use case prioritization process, and the system design principles. The chapter also addresses the evaluation design and concludes with ethical considerations regarding data handling and reporting throughout the project.

3.1 Case Study Context

This thesis uses a single-case study approach, situated within a specific enterprise. Yin [35] describes a case study as an inquiry that examines a current phenomenon in depth within its real-world context. The approach is especially fitting when 'how' or 'why' research questions are posed, when the researchers have little control over events, and when the study focuses on a real-world present-day phenomenon, meaning it is not fully historical. All three conditions apply in this thesis: asking how language model techniques can generate insights from internal data, the organizational setting and its constraints cannot be manipulated experimentally, and the relationship between the system and its enterprise context is itself central to the study.

A common concern with single-case research is the extent of generalizability of findings. Flyvbjerg [36] addresses the exact concern directly, arguing that the belief that one cannot generalize from such a study rests on several misunderstandings of the role case studies play in knowledge production. A single case can serve as a critical test of theory, generate transferable knowledge, or provide concrete, context-dependent insight that purely abstract studies cannot offer. In design science research, the goal is not statistical generalization from a studied sample to an entire population, but rather analytical generalization, where the case can be used to highlight theoretical propositions and design principles [35]. This is in line with the sought-after contributions, aiming to produce transferable insights about pipeline architectures and enterprise implementation factors derived from a focused study within one real setting.

The case organization is Essity, a Swedish global company specializing in hygiene and health products. Essity operates in around 150 countries, with approximately 36,000

employees, and develops products across four business areas: Health & Medical, Personal Care, Consumer Tissue, and Professional Hygiene [37].

The study was performed within the R&D division, responsible for product development, material science, packaging innovation, and engineering. The division operates through teams that span technology areas, laboratory testing, prototyping, and consumer insight. Over time, such teams generate a substantial amount of internal documentation, well-suited for a single-case study.

3.2 Design Science Research

This thesis follows a Design Science Research (DSR) approach [38], complemented by a qualitative organizational analysis. The combination mirrors the research question setup in this thesis. RQ1 calls for the construction and comparative assessment of technical artifacts, while RQ2 requires an understanding of the organizational factors found in the context where those artifacts will operate.

Design science research, as presented by Hevner et al. [38], seeks to extend both organizational and human capabilities through the creation of novel artifacts. Unlike natural sciences, which aim to explain or predict natural phenomena, DSR is instead prescriptive. This paradigm shifts the focus from observing and analyzing an existing reality to a creative exploration of what could be, thereby enabling the development of innovations that tackle problems traditional methods may fail to address [38]. In this paradigm, knowledge is not only generated by observing and theorizing, but also through the process of building and evaluating a designed solution to an identified problem. The approach is suited for research that sits in the intersection between technology, people, and organizations [38], which describes this thesis precisely.

Peffers et al. [39] employed the paradigm in a process model consisting of six activities: problem identification and motivation, definition of objectives, design and development, demonstration, evaluation, and communication. This thesis follows the first five phases. The sixth, communication, is excluded from the thesis as it concerns the distribution of findings rather than the production of such findings. In a thesis context, the report and its defense fulfill this function. The activities did not proceed strictly in sequence but iterated throughout the project. The activities are shown in Figure 3.1.

The first two activities, problem identification and objective definition, are addressed through an exploratory interview study (Section 3.3) and the subsequent use-case prioritization (Section 3.4). Interviews with stakeholders across the organizations surfaced concrete needs and important factors to consider.

The design and development activity corresponds to the construction of the two insight-generation pipelines. Both pipelines were developed iteratively within the enterprise's environment, with design choices informed by design methodology (Section 3.5).

The demonstration and evaluation activities were performed through the launch of the system within the enterprise. Furthermore, a comparative study is performed

where key users test both pipelines against the same query set. That evaluation combines both quantitative and qualitative measures (Section 3.6).

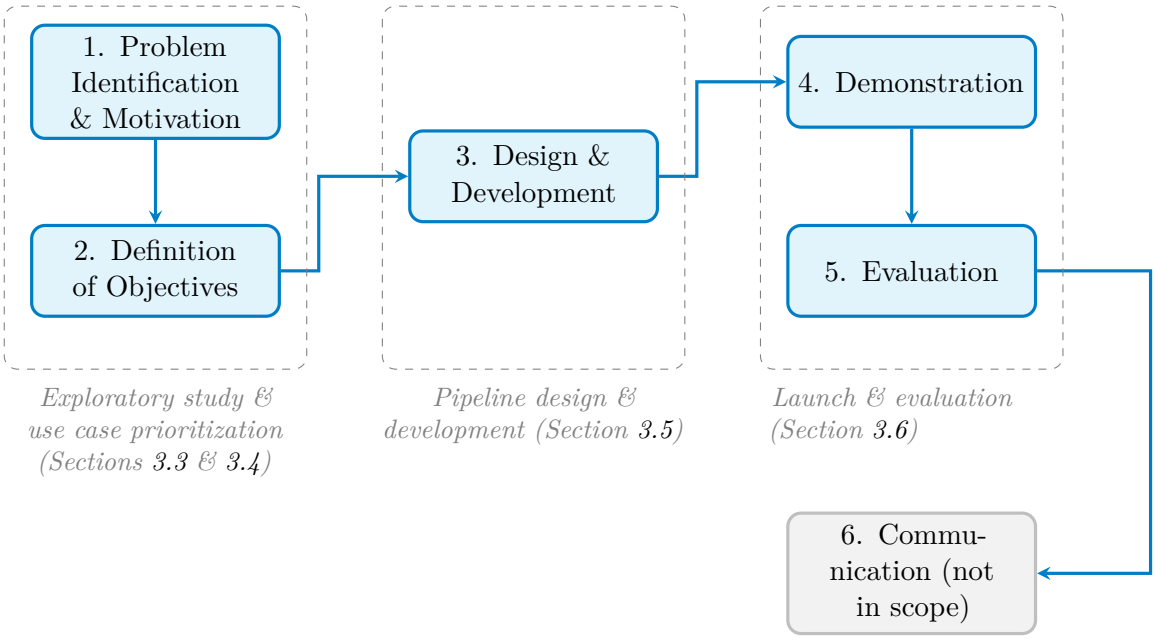

Figure 3.1: DSRM process activities and their mapping to thesis sections.

The organizational analysis that addresses RQ2 is based on the qualitative data gathered during the exploratory phase together with continuous observations. Findings are interpreted through the TOE [32] framework which provides a structure for identifying factors affecting implementation of insight-generation systems in enterprise settings. The analytical component does not follow the DSR-model, but contributes as a complementary behavioral perspective, much in line with Hevner et al.'s [38] observation that design science and behavioral science are both important to systems research.

3.3 Exploratory Interview Study

The thesis utilizes an exploratory, qualitative study aimed at capturing stakeholder perceptions about AI opportunities, data availability, and organizational factors. In terms of the DSR-process, this addresses the two initial activities: problem identification and definition of objectives. The inquiry also serves a second purpose of supplying qualitative data. The data acts as a basis for analyzing enterprise implementation factors through the TOE-framework.

Several authors reinforce the importance of this preparatory work. In a study by Hofmann et al. [40], the study relates to the 'Preparing', 'Discovering', and 'Understanding' steps for developing purposeful AI use cases. The steps require a systematic gathering of domain-specific information about processes, data availability, security concerns, and organizational challenges. Interviews supply these inputs by

exploring problems, opportunities, and dependencies for assessing feasibility and fit. Furthermore, Bodendorf [41] reinforces why this exploratory study is needed. Organizations typically face barriers such as weak overall AI strategies, unclear returns on investment, lack of data quality and availability, limited expertise, and lack of acceptance. A structured, extensive planning step that captures high-value tasks, data constraints, and stakeholder needs is therefore essential to mitigate risks.

3.3.1 Participant Selection

Twenty-one stakeholders were interviewed, selected to broadly represent many roles and functional areas within and adjacent to the R&D division. Participants included product developers, scientists, engineers, laboratory staff, technology area managers, coordinators, IT specialists, and consumer insight managers. The selection aimed for diversity rather than statistical representativeness, following the case study logic [35]. A full list of participants can be found in Appendix A.

3.3.2 Interview Design

The interviews followed a semi-structured format. Semi-structured interviews differ from standardized survey interviews in that they do not follow a fixed order for asking questions. Instead, they are guided conversations that follow some thematic prompts, allowing for a free dialog [42]. They aim to help respondents express their experiences and observations in their own terms. They are effective because the flexibility uncovers motivations, beliefs, and insights that rigid, standardized questions would miss [43]. Questions revolved around five areas: Background and Role, Data Sources, Current Pain Points, AI Opportunities and Use-cases, and Additional Input. The full interview guide is shown in Appendix B.

3.3.3 Data Analysis

The interview data are analyzed using thematic analysis following the six-step process described by Braun and Clarke [44]: familiarization with the data, generation of initial codes, search for themes, review of themes, definition and naming of themes, and production of the report. The analysis was driven by the participants' own opinions, allowing themes to surface from the material rather than being forced in advance. The resulting themes were subsequently investigated through the TOE-framework, connecting empirical findings to the theoretical structure.

3.4 AI Opportunities: Use Case Prioritization

Following the interview study, a structured transition from exploratory findings to actual design inputs was needed. The interviews surfaced important findings around TOE-factors, but also revealed a range of potential AI use cases. The potential cases were not all equally feasible or valuable, therefore a prioritization step is necessary to determine what to focus on and to derive requirements for system design.

Use case prioritization has been addressed by both academic frameworks and leading companies, with most approaches converging on two-axis scoring models that balance business value against implementation constraints. Industry examples include Google Cloud's value-actionability-feasibility matrix [45], OpenAI's impact-effort method [46], and BCG's finding that leading companies achieve better outcomes by prioritizing fewer, higher-value use cases [47]. Weber et al. [48], formalize the same logic academically along two core dimensions: business orientation and implementation complexity. In this thesis, their framework serves as the primary scoring model. However, the composite dimensions were broken down differently and adapted to the specific enterprise constraints for this thesis, as showcased in Table 3.1.

Table 3.1: Prioritization Dimensions and Weightings

Dimension	**Sub-criterion**	**Weight**
Business Orientation (BO)	Strategic Alignment (0–5)	0.40
	Value Potential (0–5)	0.40
	Reach (0–5)	0.20
Implementation Complexity (IC)	Implementation Feasibility (0–5)	0.40
	Data Accessibility (0–5)	0.20
	Data Sensitivity (0–5)	0.40
Final Prioritization Score	BO vs. IC	0.50/0.50

The scoring model is then based on grading each composite dimension between 0 and 5 and aggregating the scores along the core dimensions, according to predefined weights. The final prioritization score combined business orientation and implementation complexity with equal importance, reflecting that neither dimension should dominate the selection as a use case must be both valuable and feasible.

3.5 System Design

In this thesis, two different insight generation systems are designed and constructed, which relate to the third activity in the DSR-process: design and development. Hevner et al. [38] characterizes this activity as inherently iterative, following a generate-test cycle in which designs are produced, evaluated, and later refined. This principle guided the development throughout. Rather than producing a fully specified system and then implementing it, each pipeline was developed incrementally and revised based on stakeholder feedback.

A related concern is that design science methods often focus excessively on technical factors at the expense of organizational context. Sein et al. [49] argue that IT artifacts are formed by organizational settings both during development and during use. They claim that building, organizational intervention, and evaluation should be interwoven rather than sequential activities. This perspective informed the approach in this thesis: the pipelines were not designed in isolation and then shipped off for testing. Instead, development was performed in close collaboration with domain

stakeholders and was based on ongoing insight on what constitutes a valuable on design, output formats, and constraints to respect.

The design drew on the established theoretical foundations. The first pipeline builds on architecture proposed by Lewis et al. [9], with a reasoning layer built on techniques by Wei et al. [25], Kojima et al. [26], and Zhang et al. [27]. The second pipeline builds on the Recursive Language Model framework introduced by Zhang et al. [11].

Since the system was designed for use by real stakeholders, the user interface should follow some user-centric approach. Sharp et al. [50] argue that a user-centered approach within interaction design should be grounded in understanding who the users are, what they are trying to achieve, and where the interaction takes place, while using iterative cycles of prototyping and evaluation involving actual users throughout the process. Norman [51] emphasizes the importance of designing around the users conceptual models or mental models rather than internal system logic. The resulting system interface should communicate its functionality in a way that aligns with how people naturally think and act. These principles guided the design through collaboration with a user-centric design coach within the organization, whose role is to assist in developing intuitive and accessible interfaces prior to application launch.

3.6 Evaluation Design

To ensure rigor in the evaluation strategy that aligns technical performance with business utility, while adhering to the DSR-paradigm, this project will employ the Framework for Evaluation in Design Science (FEDS) [52].

The FEDS framework is designed to guide researchers in planning and implementing the evaluation of solution artifacts within DSR. The framework proposes a four step method to create the evaluation design. These steps consist of analyzing the requirements for the evaluation, mapping requirements to evaluation strategies, selecting an appropriate evaluation method, and designing the evaluation in detail [52].

3.6.1 Analyzing Requirements for Evaluation

The first step of FEDS involves defining the overarching objectives of the evaluation. This is to ensure they align with the research's constraints and purpose. According to Venable et al. [52], this requires identifying, analyzing, and prioritizing all the goals and requirements for the evaluation. This includes determining evaluands, the nature of the evaluands, constraints, and the required rigor of the evaluation [52].

In this project, the RAG and RLM are identified as the primary evaluands. The requirements are derived from the overarching goal of the project and were decided to be categorized into three core areas:

- **Technical Rigor**: Given the stakes of the data, the system has a low tolerance for hallucinations and factual inaccuracies. The evaluation should thereby prove that insights are derived strictly from the provided context.

- **Socio-Technical Fit**: As the project aims to explore how these systems fit within an enterprise setting, the evaluation should measure how effectively the models bridge the gap between technical ability and user utility.
- **Contextual constraints**: Due to data sensitivity, the evaluation has to be performed in a controlled, company approved environment.

How the three requirements are met is explained in detail in the subsequent sections.

3.6.2 Mapping Requirements to Evaluation Strategies

In this step, the required contextual factors from step 1 are mapped to a specific strategy in the DSR evaluation strategy framework. This process involves matching the projects specific goals, artifact properties, and resource constraints against two criteria dimensions defined by Venable et al. [52].

Venable et al. explain that the selection of an appropriate strategy is made with the help of a two-dimensional framework that relates strategies to the dimensions of ex ante vs ex post and artificial vs naturalistic.

- **Ex Ante vs. Ex Post:** This dimension distinguishes between evaluations performed prior to or during the artifact's construction (Ex Ante) and those performed after the artifact has been created (Ex Post).
- **Artificial vs. Naturalistic:** This dimension refers to the evaluation environment, distinguishing between controlled settings (Artificial) and real-world contexts (Naturalistic).

By relating the contextual factors to criteria for these dimensions, the framework provides a map to four quadrants, where each quadrant contains a set of suitable strategies. Venable et al. also note that many different strategies can be chosen, and a project does not need to be limited to only one quadrant [52].

This study adopts a hybrid evaluation strategy, including both the Artificial Ex Post and Naturalistic Ex Post quadrants. This is to enable controlled technical benchmarking with real-world organizational feedback.

The Artificial Ex Post strategy will act as a technical verification gate. This is to baseline the technical abilities of the systems before testing on organizational data.

The Naturalistic Ex Post strategy will work as a comprehensive organizational-centric evaluation metric. This is done by evaluating the models in its intended setting with real user considerations. The evaluation will thereby reflect true organizational value.

3.6.3 Selecting Appropriate Evaluation Method

The third step is to choose an appropriate evaluation method based on the chosen strategies. This is done using a DSR evaluation method selection framework, which relates the quadrants in the evaluation strategy framework to quadrants with appropriate evaluation methods.

For the artificial strategy, an automated testing method was chosen and implemented, drawing inspiration from the RAGAS framework [53], which employs an LLM-as-a-Judge approach, where a separate language model is responsible for scoring [54]. The paradigm combines the scalability of automatic evaluation with contextual awareness closer to human judgment, although it may introduce known biases, such as positional effects, that must be accounted for in the interpretation of results [54]. Even so, the method was chosen as it provides a cost-effective and scalable evaluation that provides a baseline technical benchmark before including human evaluation. The method is further explained in Section 3.6.4.

For the naturalistic strategy, a quantitative survey method was chosen. This method will measure dimensions of perceived value, trust, and accuracy of the systems. These dimensions are critical aspects that automated metrics cannot objectively measure, as they are subjectively perceived. Thereby, the survey evaluates the alignment between the system output and the user's mental models [51]. The strategy is expanded upon in Section 3.6.5.

3.6.4 Designing Automated Evaluation in Detail

The last step is planning the evaluation in detail. Details like how different evaluation will fit together and in which order to execute the evaluation will be decided [52]. For this project the focus will be on how to fairly compare the linear RAG model with the multi-step RLM model.

To ensure that the RAGAS evaluation is grounded in representative real-world data, the evaluation will utilize a modified version of the RAGAS testset generator. Modifications was made to ensure compatibility with company approved LLM servicepoint. The module uses the LLM to generates an evaluation set consisting of 80 questions and ground truth pairs [53]. To test the reasoning ability of the systems, the dataset generator will use a question-generating distribution with a high percentage of multi-step, specific, and abstract queries. Specific queries are questions that ask for direct facts and don't require complex reasoning to answer. Abstract queries are, on the other hand, intentionally phrased to have an increased complexity that requires reasoning to be answered. The numbers for each strategy is the following:

- **Single-Hop Specific Query:** 20
- **Multi-Hop Specific Query:** 30
- **Multi-Hop Abstract Query:** 30

The Multi-Hop Queries are constructed so that the model has to synthesize information across different parts of the corpus [53]. This explicitly addresses the technical aspects of limited context and reasoning.

Comparing the RAG system with the RLM using the RAGAS framework is not trivial. This is because the internal processes of the two models differ, which will affect metrics that are not model-agnostic. To address this issue, this thesis proposes a two-step approach.

1. **Model agnostic metrics:** These metrics are agnostic to the inner workings of the model and are just conditioned on the provided output of the model. The evaluation of the two models works identically. Using Ragas' recommended metrics as a guide, the following relevant metrics were selected.

    - **Answer Correctness:** Evaluates the semantic similarity between the generated answer and the test set 'ground truth'

    - **Answer Relevancy:** Evaluates how directly connected the answer is to user intent

2. **Model specific metrics:** To evaluate the technical rigor, regarding how well the model adheres to grounding the answers in the provided context, model-specific metrics are also required.

    - **Faithfulness:** Evaluates the proportion of generated claims that can be supported by retrieved documents. For RAG, it will be based on the provided context documents. For the RLM, the evaluation will be based on context summaries retrieved from subsystems [53]. Context retrieved in the root system is not used for grounding an answer and is thereby omitted in the evaluation.

All three metrics are implemented based on the RAGAS definitions using LLM-as-a-judge, instructed to score the results from 0 (low) to 1 (high). The specific prompts used are shown in Appendix C. In addition to the three RAGAS based metrics, token usage and latency were recorded to allow for a cost-effectiveness comparison.

3.6.5 Designing Expert Evaluation in Detail

While the Ragas framework method provides an objective evaluation of the baseline technical performance of the models, it does not directly measure the user value of the system as this is realized through the organization's interaction and interpretation of outputs [49]. To evaluate this value, a quantitative survey method is used. The survey is grounded in the principles of User-Centered Design [50] and Mental Model Alignment [51]. Using a 5-point Likert scale [55], the survey is designed to measure three perceived quantities, where the values 1, 3, and 5 are anchored by explicit descriptions, as shown in Table 3.2, and 2 and 4 are interpreted as intermediate values. The quantities scored are the following:

- **Utility:** The insights provided are useful and non-trivial.

- **Alignment:** The answer and reasoning logic aligns with the question.

- **Trust:** The answer is correctly grounded and is considered trustworthy.

Table 3.2: Scoring rubric for expert evaluation dimensions

Dimension	**1**	**3**	**5**
Utility	Trivial, irrelevant or non-factual	Somewhat useful	Highly useful and non-trivial
Alignment	Does not address the question	Addresses the question with some gaps	Fully and directly addresses the question
Trust	Would not use without thorough verification	Would use with some caution	Would confidently use at face value

To test these qualities, queries are collected from system stakeholders with instructions to represent real-world questions. A total of 40 sample queries are executed across both systems, yielding query-answer pairs for the two models. Each pair is distributed to three separate domain experts, all of whom had general knowledge of the domain as a whole and specific sub-domain knowledge.

Full coverage by all three experts on every query is not feasible. This is because domain experts has limited availability, and some queries will fall outside individual experts' areas of knowledge. For these scenarios, the experts are instructed to skip rather than guess. The unbalanced coverage is accepted as to retain the full 40-query breadth of queries, which spans multiple sub-domains instead of restricting the evaluation to a smaller and more general subset that all three experts could complete fully. To minimize bias, the question and answer order is shuffled so experts are unaware of which pipeline produced what answer.

Since every expert rating on a query is rated for both pipeline outputs, the design is paired for each expert-query. This means each query rated by an expert yields a direct comparison of pipeline scores. Pipeline preference is therefore primarily assessed on paired differences, calculated by subtracting the RLM score from the RAG score. Regarding the paired differences, positive values signify a higher score for the RAG output. Statistical significance is derived using the Wilcoxon signed-rank test [56], pooled across all comparisons. For each metric, results are summarized by two complementary statistics. The median delta captures the typical magnitude of the score gap between pipelines, and the share of expert-query comparisons favoring each pipeline captures how consistently one was rated higher. The two summaries are complementary, where the median captures the effect size, while the share captures the prevalence.

As a final visual complement, the aggregated score distributions for each metric will also be included. Given that the distributions are highly dependent on the preferences and evaluation frequency of individual evaluators, which is a very small set ($n = 3$), the distributions are not used to draw any statistically significant conclusions. Instead, they are reported only as a visual reference for where each pipeline's ratings concentrate on the five-point scale, complementing the paired analysis.

3.7 Ethics

The project handles internal company documentation and must comply with data protection standards. The following subsections outline the main risks identified and the measures taken to mitigate them.

3.7.1 Information Leakage

The documentation used in the project may contain sensitive operational or business-critical information. To prevent unintended disclosure or information leaks, all data processing takes place exclusively within Essity's secure and access-controlled environment. Documents are not exported, transferred, or stored outside approved systems. Sensitive personally identifiable information is masked during the document parsing process, ensuring that such information is not sent to the LLM or provided to the system users.

3.7.2 Model Hallucination and Provenance

LLMs may generate fabricated or speculative information. To address this, all model outputs used in the study will be grounded in verifiable documents. Provenance tracking is enforced so that responses can be traced back to original sources. In the application interface, users are instructed not to act on any output that lacks document support, as it should be treated as unverifiable and invalid.

3.7.3 Disclosure in Reporting

Examples and visualizations presented in the thesis report carry a risk of unintentionally revealing internal data. To mitigate this risk, no real document fragments, project identifiers, personal data, or sensitive terminology are quoted directly. Examples are rewritten or paraphrased to show patterns without exposing the true organizational content. Information about the dataset is only presented in aggregated form, never showcasing any real content.

3.7.4 Ethical Use of AI

The system is intended for internal insight-generation. It is not designed to act autonomously in any way, but to assist employees in understanding and navigation of complex, and sometimes messy documentation. Model outputs should therefore be interpreted as supportive guidance rather than authoritative answers. The model interface will display a warning of this kind.

From an ethical standpoint, this approach ensures that the model does not mislead users into relying on AI-synthesized information. Using the information without domain oversight could result in misunderstandings or incorrect conclusions. With human interpretation, the project avoids these risks and ensures that expert judgment remains a key element in the process.

4

Exploratory Findings and Design Requirements

This chapter reports the empirical findings from the exploratory study of the project. It presents the results of the thematic analysis of stakeholder interviews and maps the findings to the TOE framework. Then, the outcome of the use case prioritization is demonstrated, showcasing the selection of project documentation as the target data domain. The chapter concludes with a translation of exploratory findings into a set of functional and non-functional design requirements for the technical solution.

4.1 Thematic Analysis of Interviews

As detailed in Section 3.3, the interview data were analyzed using thematic analysis. The analysis revealed four themes that together depict a coherent picture: locating relevant insights often required manual effort and depended on individual memory rather than systematic retrieval.

Theme 1: Fragmented knowledge ecosystems limit efficiency

Across roles and teams, participants described working within complex environments composed of multiple independent systems. The systems were, for instance, used to manage knowledge, laboratory information, project data, material data, or supplier documents. The platforms contain large volumes of both structured and unstructured data, but operate with little to none interoperability. The result is that individuals may know that relevant information or data should exist within the organization, but cannot locate or combine it efficiently.

Three subthemes were observed within the theme. First, interviewees pointed to scattered data systems without cross-system searchability. Second, there was an observed difficulty in reusing historical work, noting that outcomes and ideas were sometimes hard to discover. Third, inconsistencies in nomenclature and archiving were described as obstacles to navigating documentation across teams.

> "Finding information about product specifications takes time, it is unnecessary, it becomes very manual. The data exists internally, but you mainly end up emailing people" - Interviewee 6

Theme 2: Time-consuming manual processes

A direct consequence of the fragmented knowledge landscape is that individuals within the organization spend time on low-value information tasks. Interviewees reported long searches through systems to locate correct files, repetitive administrative documentation and management, and inefficient cycles. The manual processes reduce the capability available for higher-value work.

The specific realizations varied across interviews, but the underlying pattern was the same. Time is spent gathering information rather than acting on it. The regulatory domain emerged as a particularly interesting example, where volumes and complexity of EU legislation pose a significant challenge. However, other participants described similar patterns in their own domains, not having trouble with coming up with areas with a desire for improved workflows.

> "It is a large document that is updated periodically, simply trying to understand the requirements demands substantial time and effort" - Interviewee 10

Theme 3: Organizational desire for AI-based acceleration

Despite the diversity of operational contexts represented in all interviews, many participants mentioned a shared ambition: AI systems could strongly accelerate interpretation, search, and concept exploration. Four categories of envisioned applications were discussed during interviews: search and summarization agents, analytical and predictive tools, regulatory interpretation tools, and concept generation tools.

The breadth of the four categories is notable. Although, across teams, AI was framed not as a replacement for expertise but as a means of reducing time spent on low-value tasks. Search and summarization agents were the most frequently mentioned application. Participants across nearly all functions expressed interest or experience with Copilot-style tools that can retrieve and synthesize information across systems. Worth noting was that every interviewee mentioned the need for referencing of sources in discussions around such agents.

> "AI is a tool that can break a barrier for us, especially when it comes to ..."
> - Interviewee 7

Theme 4: Need for connected and trustworthy data-driven workflows

The last theme captures conditions that were expressed as prerequisites for successful AI adoption. Rather than describing what AI should do, it addresses what the organization and its ecosystem must provide for AI to deliver value.

During the interviews, verification and traceability surfaced as the most common concerns. It was emphasized that a central feature is that AI must refer to source documents, to avoid creating misleading content. It was noted that a system producing answers without references would be worse than no system at all. Unified data structures were seen as necessary groundwork. Interviewees shared a desire for harmonized nomenclature, shared archiving setups, and consolidated repositories. Future-proofing requires that systems are not static, instead it should automatically incorporate new data, as long-term maintainability is important.

> "An answer would need to give a pointer to the system and a brief reference for why it was suggested. That alone would save an incredible amount of time"
> - Interviewee 20

Table 4.1 summarizes the four themes, their definitions, and associated subthemes. Then, Table 4.2 provides an overview of how the four themes connect to each TOE-dimension and its constructs.

Table 4.1: Summary of interview themes

Theme	**Definition**	**Subthemes**
Fragmented knowledge ecosystems	Extensive, but scattered knowledge resources with low interoperability across systems	Scattered data systems, Difficulty reusing historical work, Nomenclature and documentation inconsistencies
Time-consuming manual processes	Significant time spent on low-value information tasks due to fragmentation	Manual search for data, Administrative overhead, Inefficient iteration cycles
Desire for AI-based acceleration	Aspiration for AI tools that accelerate search, interpretation, prediction, and ideation	Search and summarisation agents, Predictive tools, Regulatory interpretation, Concept generation
Trustworthy data-driven workflows	Prerequisites for AI adoption including traceability, data quality, and governance	Verification and traceability, Unified data structures, Future-proofing

Table 4.2: Mapping of interview themes to TOE dimensions and constructs

Dimension	**Themes**	**Constructs touched**	**Key observations**
Technology	1, 2, 3, 4	System capability, Data and analytics readiness, Technology usability, Complexity, Relative advantage, Innovation capability	Limited integration and usability across platforms. High workflow complexity currently. Strong recognition of AI potential.
Organization	1, 2, 4	Human and structural readiness, Cultural and governance alignment, Knowledge and learning, Staff skills, Change capacity	Fragmented documentation and data systems hinder knowledge management and retention. Governance alignment viewed as essential for sustainable adoption.
Environment	2, 3, 4	Regulatory and policy context, Collaborative ecosystem, Social legitimacy and ethics	Obligations and supplier collaboration drive both manual workload and demand for AI-assistance.

The technology dimension was most broadly represented, showcased in all four themes. This reflects that the interviews were conducted in a technology-oriented R&D setting. The organizational dimension is shown primarily through the barriers to knowledge reuse and governance requirements for AI adoption. The environmental factors are mainly anchored in regulatory pressure and external collaboration.

4.2 AI Opportunities

Along with the thematic analysis, the interviews surfaced a wide range of potential AI applications. To transition from broad opportunities within insight generation to a focused implementation scope, these were prioritized.

Stage 1: Initial Scoring

A total of 41 AI use cases were identified across eleven areas. Each use case was scored in consultation with domain experts based on the criteria defined in Table 3.1. Several patterns emerged at the first stage. Many use cases were business relevant but were held back by implementation complexities, often due to lacking data accessibility. Use cases tied to well-structured datasets often performed better. Scores ranged between 2.2 and 4.3 with a median of 3.2.

Stage 2: Shortlisting

The five best performing use cases with the highest aggregated scores were selected for a deeper shortlist review. The five shortlisted use cases spanned several functions and are described generically in Table 4.3 to avoid violating confidentiality constraints.

Table 4.3: Shortlisted use cases with aggregate scores

AI use case	Weighted Score
Project learning retrieval	4.3
Regulatory interpretation assistant	4.0
Method retrieval assistant	4.0
Synthetic preference generation	3.6
Idea clustering	3.5

Stage 3 & 4: Feasibility Deep-dive and Final Selection

The shortlisted use cases were examined further with regard to data extraction, repository structures, potential integration, and additional stakeholder input. This expanded review brought forward constraints that the initial scoring could not capture. The methods retrieval assistant was removed at the deep-dive.

For the final selection, after confirming technical feasibility and data availability, the use case with highest business orientation emerged as the strongest overall candidate. A project-knowledge retrieval system enabling improved access to historical learnings and documentation across projects. Other use cases were saved as back up, and for

future development within the organization. A further explanation of the selection of project documentation as data domain is motivated in Section 4.3.

A summary of the prioritization and selection is showcased in Figure 4.1.

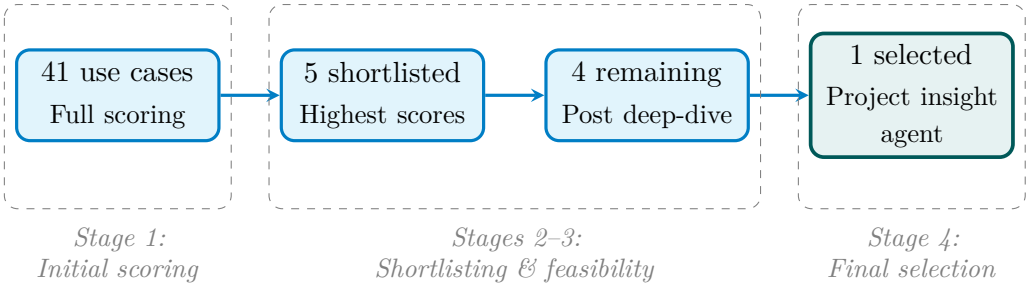

Figure 4.1: Use case prioritization funnel across three evaluation stages

4.3 Selected Domain: Project Documentation

The prioritization identified project documentation as the most suitable data domain for the insight-generation system.

The project documentation collection at the R&D division has accumulated over decades. In this thesis project, four types of documents were used: knowledge briefs, project final reports, product development reports, and knowledge area documents. At the time of the initial data collection, 600 documents were available and accessible. Knowledge briefs are crafted summaries of specific findings, targeted to transfer insights to a broader audience. Project reports and product development reports are more extensive documents, covering additional activities such as problem statements, designs, results, and conclusions. The knowledge area documents summarize outcomes and learnings from many projects within a topic area.

All four types of documents include project-specific details and learning outcomes, which means the corpus is well-suited for the types of insights defined in Section 1.1. Project documentation with its mix of decisions, outcomes, and learnings, provides a well-suited and natural foundation for cross-referential tasks such as finding recurring themes, transferable insights, and connections between separate projects.

The documents were suitable as they are predominantly text-based and follow structured or semi-structured formats. They were stored in accessible repositories without requiring tedious data exports from external services. Data sensitivity was at a manageable level. The data domain also contained few infrastructure dependencies that could cause problems.

The interview study provided direct evidence for the value of the project documentation corpus. stakeholders across nearly every function described the value of efficiently retrieving and reusing knowledge from prior projects. Several participants noted that sometimes documented project files and outcomes could be lost, for instance, if a project team disbands or the project leader leaves the organization. The

corpus therefore represents organizational knowledge that is rich in content, but underutilized in practice. The content is also often cross-referential, meaning that real enterprise questions about the data often require synthesizing information across multiple documents and drawing intermediate conclusions.

4.4 Translated Design Requirements

The exploratory findings combined with the characteristics of the enterprise and the selected use case were translated into a set of design requirements for the insight generation system. They are organized into functional requirements (FR), specifying what the system should be able to do and non-functional requirements (NFR), specifying how it should operate.

In Table 4.4, the list of requirements bounding the solution space for system design are shown. In addition, the source of the requirement is stated.

Table 4.4: Design requirements derived from exploratory findings

ID	Requirement	Type	Source
FR1	Natural language querying of project data	Functional	Themes 1, 3
FR2	Support for single-document, cross-project, and multi-step queries	Functional	Theme 3
FR3	Provenance tracing to source documents in every answer	Functional	Theme 4
FR4	Explicit indication when evidence is insufficient	Functional	Theme 4
NFR1	All processing within approved computing environment, no data export	Non-functional	Enterprise policy, Theme 4
NFR2	No fine-tuning on company data, prompting and retrieval only	Non-functional	Delimitation (Section 1.4)
NFR3	Outputs framed as guidance, not authoritative answers	Non-functional	Theme 4
NFR4	Interface usable without technical training	Non-functional	Themes 2, 3
NFR5	Accommodate new documents without manual reconfiguration	Non-functional	Theme 4

5

System Design and Implementation

This chapter describes the design and construction of the two insight-generation pipelines. The design follows the iterative, generate-test approach established in Section 3.5, where each component was developed incrementally alongside input of the stakeholders. The following sections first outline the shared architecture and preprocessing pipeline, before detailing the specific configurations for the RAG and RLM pipelines.

5.1 Architecture Overview

The system is designed modularly, with independent components. The main architecture can be divided into the following components:

- **Preprocessing pipeline:** Handles the process of extracting raw data, cleaning data, and storing the data in a structured database.
- **Model pipeline:** The core of the system, which handles the execution logic. Designed to support either the RAG or the RLM pipeline.
- **Frontend:** The user interface to the system. Handles the logic of passing queries and retrieving results to the selected model. The interface accepts plain natural language queries and presents responses in plain text with cited sources, requiring no configuration and low technical knowledge from the user, which is in accordance to the NFR4 requirement. The specific implementation details are not the focus of this report.

The structure ensures separation of data processing, model reasoning and retrieval, and user interaction. The infrastructure of the system is hosted on a cloud platform which handles data storage, computing and hosting internally. All processing and model inference takes place within this approved company approved environment, satisfying the constraints established in NFR1. Figure 5.1 visualizes the architecture with the flow of data and requests explicitly stated.

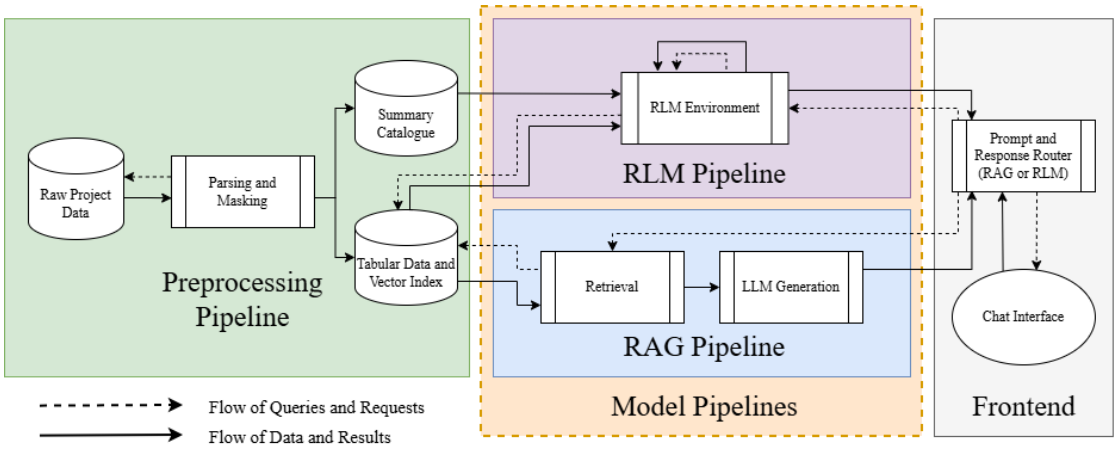

Figure 5.1: System Architecture

5.2 Data Pipeline

The data pipeline provides the foundational data used in the system. The pipeline is designed to be scalable and flexible. The pipeline should thereby be able to parse and structure the data from a custom number of raw documents. This structure ensures that the pipeline can run and ingest new documents efficiently and without additional manual work, which is in accordance with requirement NFR5.

The pipeline operates as an offline, batch process that is triggered on demand or by a configured scheduler. The pipeline produces three distinct artifacts.

- **Tabular Data**, containing plain-text and vectorised text chunks, and metadata
- **Corpus**, containing all plain-text data contained in a structured JSON file
- **Summary catalogue**, a hierarchical JSON catalogue describing the document collection at multiple levels of granularity.

The final preprocessing pipeline is visualized in Figure 5.2.

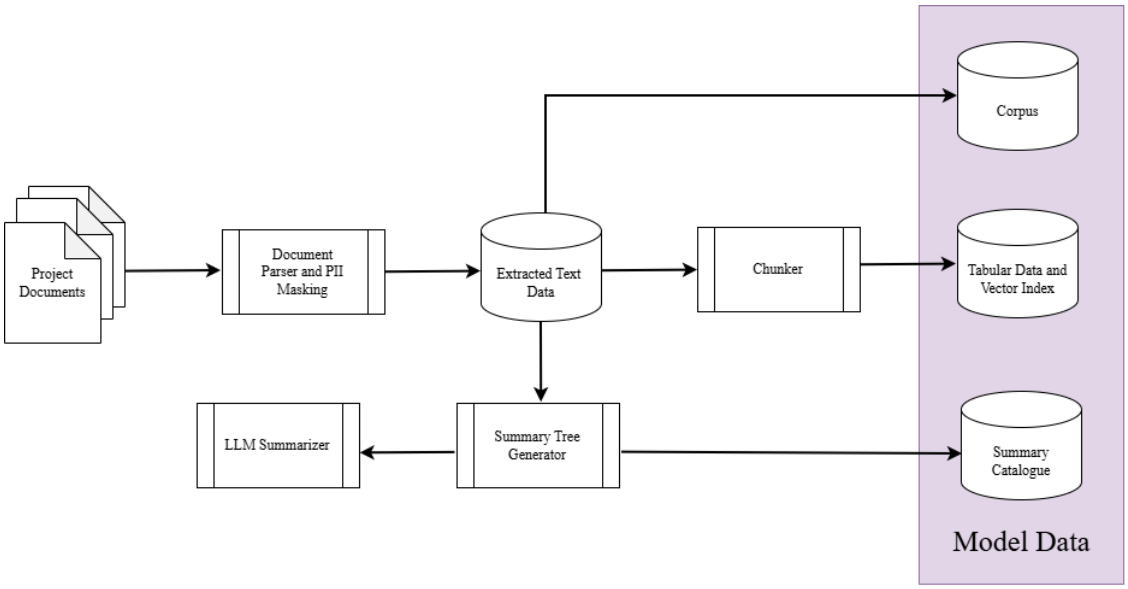

Figure 5.2: Preprocessing Pipeline

5.2.1 Document Ingestion and Parsing

The pipeline is initialised with a pointer to a designated source folder on the cloud platform, containing all raw data. The pipeline logic operates incrementally. The first step is to by enumerate all files in the folder comparing the set of files against the documents already processed. Then, it proceeds to process only the new documents, avoiding recomputation if documents has already been parsed. The next step is to route each one to a specific parser module based on file extension type. The routing process is configured to support PDF, Microsoft Word and Microsoft PowerPoint files. All the individual modules output a common plain-text representation which is passed downstream. The routing is designed to be extendable, which would allow it to handle additional file types to be supported in the future.

5.2.2 Personal Identifiable Information Masking

Before the text is stored or indexed it passes through a Personal Identifiable Information (PII) masker. A rule-based analyzer scans each document for personally identifiable information using pattern recognition. Current configurations include email, telephone numbers, and physical addresses. The information identified is replaced with anonymous placeholders that signal masked content. This step is applied to all documents to reduce the risk of sensitive information being surfaced.

5.2.3 Chunking and Vector Storage

Following PII masking, the parsed text is passed through a chunking module. This module chunks the text deterministically in 800 characters chunks, using an overlap of 100 characters. This size was chosen to be large enough to capture a coherent unit of knowledge, while remaining small enough to preserve semantic consistency

within each chunk. Each chunk is linked with its source document identifier and its position within the document, expressed as an integer page index. Each chunk is then embedded using a cloud-hosted embedding model and stored, together with its text and linked metadata, as a row in a SQL table. A cloud hosted vector search index is maintained over this table, thus enabling semantic retrieval of the chunks and corresponding metadata.

In parallel with the table write, the pipeline saves the text from each document to a structured JSON file, referred to as the corpus. This file provides a full view of all the data the model has access to in parsed standard text format.

5.2.4 Summary Catalogue Construction

The final artifact produced by the pipeline is a JSON structured catalogue that describes the collection of documents at multiple levels. The pipeline takes a bottom-up approach when building the catalogue. Starting at the bottom level for each document in a particular sub-folder, the pipeline calls the language model, providing it with a sample of the document's chunks and asking it to generate a concise summary. The summary, together with file name, path, and chunk count is written to the catalogue. When all documents in a particular folder have been processed, a summary of the folder is generated by providing the summaries of the folder's files to a language model with the explicit instruction to give a concise summary of the folder content. The process is iterated for all files and sub-folders, creating a tree catalogue of summaries, representative of the underlying raw file structure. Both the corpus and the catalogue files are then stored in a persistent volume in the cloud platform, enabling direct use for inference.

5.3 RAG Pipeline

The RAG pipeline combines retrieval-augmented generation with CoT prompting to produce structured, traceable answers from the processed documents. The implemented RAG pipeline consists of two primary modules:

- **Retriever:** The module takes the raw user query and performs a semantic search across the chunked data to rank and retrieve the chunks most similar to the query.

- **Generator:** The original query and the retrieved chunks are combined into an orchestrated prompt, which is then fed into an LLM to generate a response.

The design is sequential, using no loops or external tools beyond the initial context retrieval. The general RAG pipeline structure is shown in Figure 5.3.

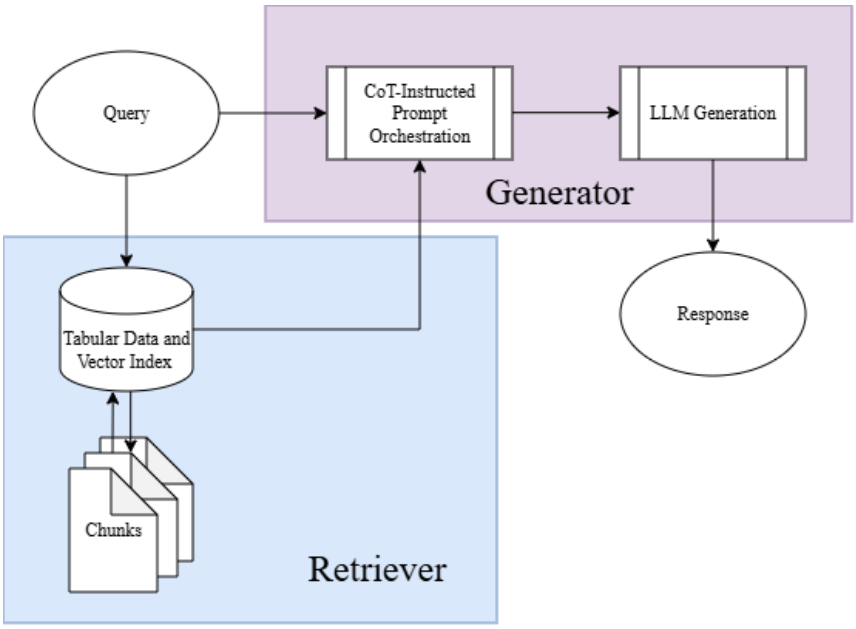

Figure 5.3: RAG Pipeline

5.3.1 Retriever

The user query is passed to the retriever, which executes a semantic search against the vector search index and returns the top-k candidate chunks based on the query, in this case, $k = 5$. The results are then passed through a ranking step, converting the embedding distances returned by the vector search into normalized similarity scores ranging from 0 to 1. The resulting chunks are ranked in descending order. Each ranked chunk also carries metadata, including source file name and chunk index, which is propagated to the generator for provenance tracking and citation purposes.

5.3.2 Generator

The generator receives the ranked chunks and assembles them into a numbered source block, with each passage prefixed by a bracketed citation anchor. This source block is injected into a few-shot CoT template, which is specified in Appendix D.1. The constructed prompt is submitted to the cloud-hosted language model endpoint. The pipeline returns a structured response containing the generated answer, the retrieved chunk texts, source provenance records, and token usage.

5.3.3 CoT-design

The CoT methods reviewed in Section 2.4.1 present a variety of possibilities and trade-offs. Zero-shot CoT is the lightest to deploy but is not able to handle as complex queries, Manual-CoT requires hand-crafted demonstrations, and Automate-CoT depends on a labeled dataset, unavailable in this domain where ground-truth insight answers do not exist. Auto-CoT requires neither manual effort nor labeling requirements, but lacks a quality gate.

This motivates the need for a design that retains Auto-CoT's diversity sampling but adds a label-free quality filter. Here, the filter is built upon self-consistency, which uses agreement across independently sampled chains as a representation of correctness. The resulting method is diversity-sampled Auto-CoT with self-consistency.

The procedure is based on an offline, three-stage pipeline applied to a set of 15 unlabeled queries collected from enterprise stakeholders, but held out from the expert evaluation. Initially, all 15 queries are embedded using the all-MiniLM-L6-v2 sentence-transformers model and clustered into $k = 3$ groups via k-means clustering [28]. As one representative query is selected per cluster, k sets the maximum number of few-shot exemplars. Three should high enough to cover distinct query types, but low enough to keep the prompt sufficiently compact.

For each cluster, the medoid query, defined as closest to the cluster centroid is used as an example for that cluster, with the mission of representing the semantic diversity of that cluster. In the second stage, a zero-shot CoT pipeline is called five times per example query at varying temperatures from 0.5 to 0.9, producing five independent reasoning chains per query. Varying temperature is used to encourage heterogeneity among the generated reasoning chains.

In the third stage, chain agreement is examined to filter out unreliable demonstrations. Because the pipeline produces free-form natural language answers rather than discrete output, majority voting is not directly applicable. Instead, the five reasoning chains for each representative query are embedded in the same way as in stage one, and the full pairwise cosine similarity matrix is computed. Then, a consistency score is computed as the mean of all pairwise similarities, capturing how tightly the reasoning chains converge in semantic space. A threshold τ is calibrated empirically to inspect consistency scores of the representative queries and excluding any query whose scores are substantially below other queries. In this case, all three representative queries for the three clusters produced consistency scores in a high, tight range (0.828, 0.885, 0.924), indicating stable semantic convergence across all sampled chains. No queries were discarded at this stage. For each stored representative query, the medoid reasoning chain is identified as the one with the highest similarity to the other four chains. In the unlikely event of a tie, priority is given in favor of the shortest chains as concise demonstrations are preferred.

Lastly, the surviving exemplars are subject to manual review before being incorporated into the few-shot template. This review step allows a human to confirm that the reasoning is factually grounded and representative of the expected answer style before the demonstrations are frozen for use in production. The embedding and clustering stages run entirely locally, ensuring no query data leaves the organization's environment during exemplar construction.

5.4 RLM-inspired Pipeline

The RLM-inspired pipeline is based on the RLM framework described in Section 2.5 by Zhang et al.'s [11]. While the design is based on RLM architecture, several adaptations were made to adhere to the constraints of the enterprise environment established in Chapter 4.4.

5.4.1 Overall Structure

The pipeline is structured as a two level system, similar to a RLM implementation with a recursion depth of one ($d = 1$). A root-level system receives the input query, navigates the context, and delegates specific tasks to subsystems. These subsystems operate within a limited context chosen by the root system, and are tasked with finding explicit information using a set of navigation tools. Each subsystem returns an answer to the root system which iterates exploration, reasoning and delegation until the user query is answered or computational budget is depleted. This iterative delegation structure attempts to addresses FR2, where single-document queries are handled by a direct subsystem delegation, while cross-project and multi-step queries are in theory solved through successive root-level exploration and parallel delegations across different documents. When the iterations are completed, a final LLM call is issued to orchestrate the final answer summary in a specified format.

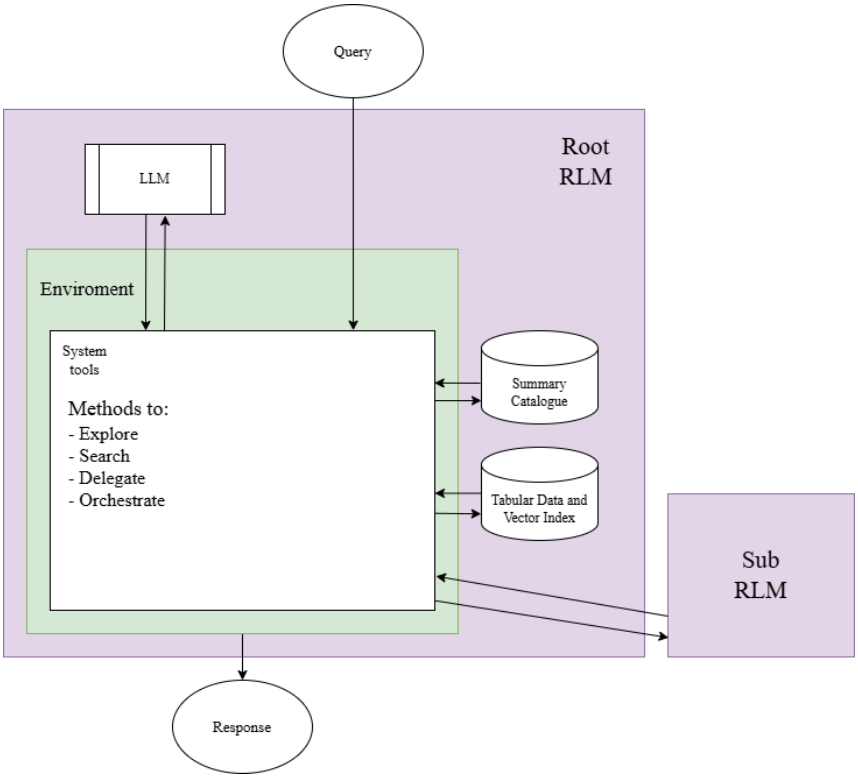

Figure 5.4: RLM Pipeline

5.4.2 Deviations from Standard RLM

The original RLM framework by Zhang et al. [11] operates through a Read-Eval-Print loop where the LLM generates Python code which is executed in a dedicated environment. Executing arbitrarily generated Python code with access to sensitive information could conflict with requirement NFR1 if the environment is not heavily restricted to make sure that no unapproved operations are performed. Instead, this implementation addresses the conflict by using a structured action protocol in which the model outputs a JSON object enclosed in a recognized XML tag at each iteration. The system then parses this output and routes the intended action to a predefined set of tools. This approach minimizes the risk of unintended actions, while preserving the iterative and adaptable workflow of the RLM method. To ensure that the LLM adheres to the specified format, specific system prompt templates was used for the root system, subsystem and orchestrator. The system prompts can be found in Appendix D.2.

5.4.3 Root System Tools

The root has access to five custom tools, which allows it to navigate, explore and delegate tasks, in the quest of answering the user query. The LLM is responsible for choosing which tools to use and which arguments to use in the call, the implemented tools are shown below.

- **Get top level tree:** Gives the top level of the summary catalogue tree, containing folder names and summaries.
- **Browse group:** Explores specific subfolders in the summary catalog returning summaries and names.
- **Broad search:** Uses vector search to find relevant documents and chunks based on semantic similarity. Used to guide further exploration of relevant documents.
- **Regex search:** Uses regular expression search to find documents with text that has exact pattern match.
- **Delegate:** Delegates a specific task to a subsystem with a given limited context, from one to a couple of documents. Root is also able to delegate to many subsystems in parallel when needed.

5.4.4 Subsystem Tools

Each subsystem has direct access to the limited context given by the root. The tools described below are restricted to only be able to access these specific documents.

- **Get document outline:** Returns sample text from intermitted intervals in the document specified. Used to inspect document structure before deeper exploration.
- **Vector search:** Used to locate relevant passages using semantic search.

- **Regex search:** Used to locate passages with exact pattern matches.
- **Read chunks:** Used to read text from the documents using from and to indexes.

5.4.5 Specific Design Choices

To ensure that the computational load remains within acceptable bounds for production use in the enterprise environment, which is consistent with NFR1, the system is limited by a hard iteration and token budget. The token budget is cost centered. Furthermore, generated tokens are weighed higher than ingested tokens. Added together, they give a bounded measure based on maximum allowed cost. The computational budget was set according to the needs of stakeholders, ensuring good cost to reward tradeoff. A hard iteration limit was also set to avoid infinite loops and wasteful exploration. Each system has its own iteration budget which it is reminded of during its iteration loops, encouraging it to use its tools efficiently, and gracefully complete its generation before being abruptly cut off.

FR3 requires that all answers are traceable to the original source document, and FR4 requires the system to indicate when the retrieved evidence is insufficient to answer the question. Both requirements are adressed at the subsystem layer. The subsystem LLM is instructed to answer with a confidence attribute (low, medium or high) and a found attribute (true or false) that flags whether relevant information was found in the assigned context. If a subsystem returns a 'not found' attribute, this is signaled to the root as a negative result, allowing the root to attempt an alternative search strategy. The final orchestrated answer utilizes the propagated information to ensure correct provenance and indication of answer confidence. Furthermore, the confidence level is surfaced in the final response by prompt instruction. This is to satisfy NFR3, as a low or medium confidence rating signals to the user that the output should be treated as a starting point for further investigation rather than a full answer.

Finally, in accordance with NFR2, no fine-tuning or training was performed. The custom system behavior is defined by the model's system prompts and the tools' functionality.

6

Launch and Evaluation

The developed prototype is deployed on the company-approved cloud platform, with direct access to the processed data and LLM service endpoints. Users interact with the system through a web interface, querying the models and receiving answers in a structured format with cited sources. The deployed version reflects the final design iteration, incorporating the outcomes of the iterative development process described in previous chapters. This chapter presents an overview of the final prototype's functionality, followed by the results of the system evaluation.

6.1 Overview of the Prototype

The final prototype is a web-based interface that allows users to ask questions and query the company's project and insight document corpus using plain natural language and receive grounded, cited answers. The prototype is designed based on the architecture in Section 5.1, and is meant to seamlessly integrate with the underlying models while adhering to the enterprise-based design requirements specified in Section 4.4.

The prototype interface exposes several features, each explained in the list showcased below.

- **Pipeline toggle:** A toggle that allows the user to switch between the RAG and RLM pipelines before submitting a query.
- **Query input:** A single text field that accepts natural language queries, requiring no specific syntax from the user.
- **Progress feedback:** RAG queries display a loading spinner for the duration of the pipeline run. RLM queries display a live status panel that streams intermediate steps, including retrieval actions, reasoning steps, and tool invocations.
- **Answer panel:** The generated answer is provided as formatted text, with numbered inline citation markers that link directly to the corresponding entry in the source panel below. All answers are accompanied with one or two sentences on how well the answer is grounded, whereas the specific answer format depends on the type of question asked.

- **Source panel:** Retrieved sources are listed as entries in a source summary. Each entry displays the document name, file path, source group, a document-level summary, and the extracted finding from the document. Together, this data enables provenance tracking, allowing claims in the answer to be traced back to a precise origin in the document corpus.

- **Token usage metrics:** Token counts are displayed below the answer, enabling usage monitoring and internal evaluation.

The interface uses a single-page layout, focusing the users attention on the query, sources and the results. Each query is processed independently, and the system maintains no chat history between queries. The limited scope of the prototype is intentional, keeping the system focused on the core retrieval and answer generation task evaluated in this chapter. Additions can be made in the future to align the system better with the evolving needs of the organization, such as conversational memory, broader corpus coverage, or role-based access control.

Figure 6.1 shows the front-end with a sample query and a masked RAG generated answer, where citations link each claim back to correct source document.

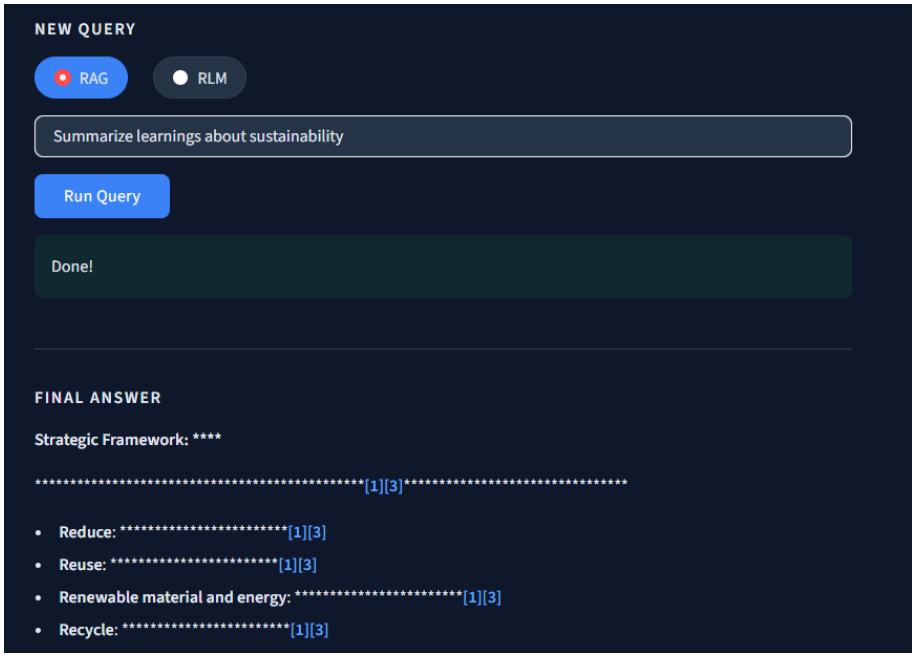

Figure 6.1: RAG Example Query

Figure 6.2 shows the same query answered by the RLM pipeline, where the interface includes a trajectory of intermediate steps.

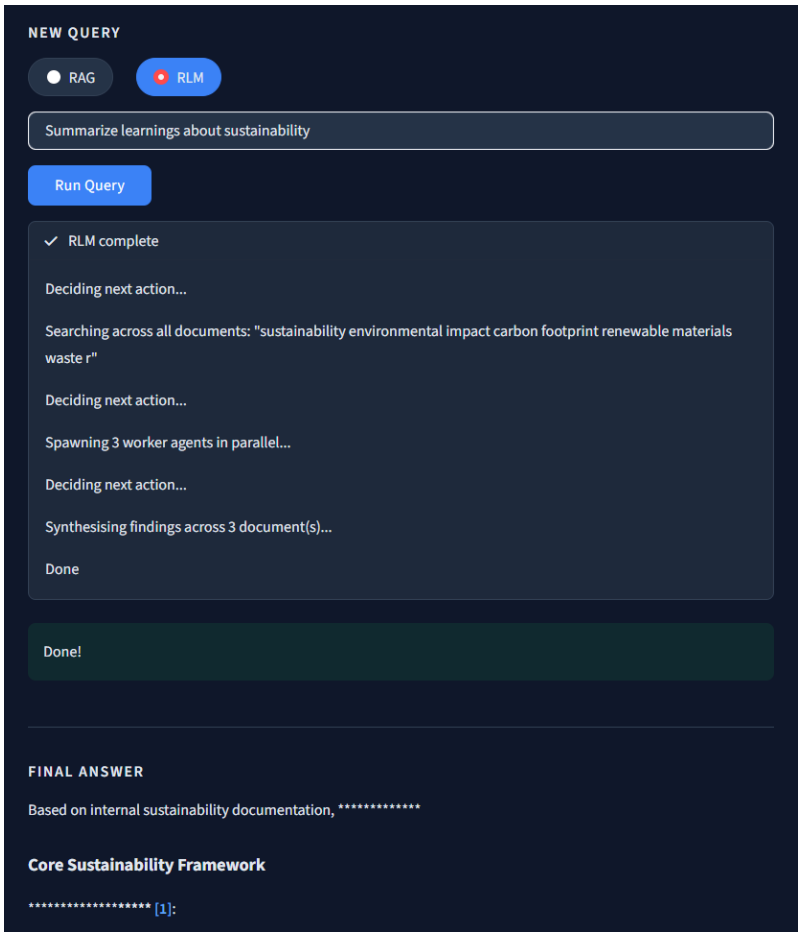

Figure 6.2: RLM Example Query

Figure 6.3 represent a example of the source panel, which allows users to track information back to the original document.

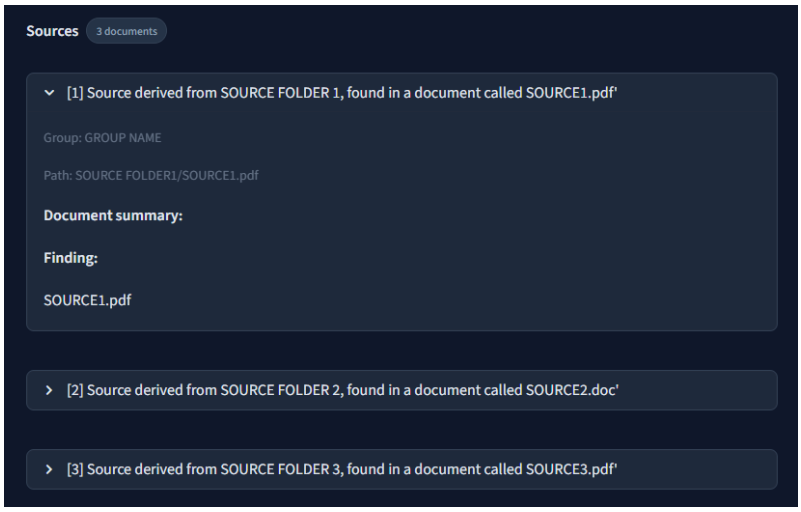

Figure 6.3: Source panel

6.2 Automated Evaluation Metrics

This section presents the results of the automated evaluation pipeline. The pipeline uses the custom testset and LLM-as-a-Judge metrics. The section covers model-agnostic quality metrics, model-specific metrics, as well as efficiency metrics. The results provide a baseline for the comparative technical analysis.

6.2.1 Evaluation Setup

The evaluation was setup in accordance to the design choices described in Section 3.6.4. The testset was generated from a subsection of the corpus using the custom generation pipeline, resulting in 80 question-and-answer pairs distributed across the three query categories. The same model used for the pipeline served as the judge for all three scoring metrics. The RAG and RLM pipelines were evaluated on the same testset. For each query and pipeline the automated metrics were recorded alongside the efficiency metrics.

6.2.2 Model-Agnostic Metrics

Table 6.1 represents the evaluated mean Answer Correctness and Answer Relevancy metrics for RAG and RLM pipelines respectively.

Table 6.1: Mean model-agnostic scores by pipeline

Pipeline	**RAG**	**RLM**
Answer Correctness	0.506	0.624
Answer Relevancy	0.806	0.907

The result from Table 6.1 show significant higher overall scores for the RLM pipeline compared to the RAG pipeline in both metrics. Table 6.2 breaks down these metrics by query strategy, showing how each pipeline performs when question complexity increases.

Table 6.2: Mean scores by pipeline and query strategy

Strategy	**Metric**	**RAG**	**RLM**
Single-hop specific	Answer Correctness	0.542	0.768
	Answer Relevancy	0.924	0.864
Multi-hop specific	Answer Correctness	0.473	0.628
	Answer Relevancy	0.802	0.913
Multi-hop abstract	Answer Correctness	0.515	0.531
	Answer Relevancy	0.732	0.928

Results in Table 6.2 indicate that the RLM pipeline maintains consistently high Answer Relevancy across all query strategies, while the RAG pipelines performance degrades when complexity increases. The Answer Correctness score gap between the pipelines is largest on single-hop queries and smallest on multi-hop abstract queries, suggesting that both pipelines struggle similarly on more complex queries.

Figures 6.4 and 6.5 visualize the score distributions for each metric and pipeline as box plots, illustrating variance and median across query types in addition to the mean scores. The central boxes spans the interquartile range (IQR), from the 25th to the 75th percentile, with the horizontal line inside the box marking the median. The whiskers reach the most extreme values that are at most 1.5 times the IQR from the box edges. Individual circles represent outliers, defined by scores that fall unusually far from the central distribution.

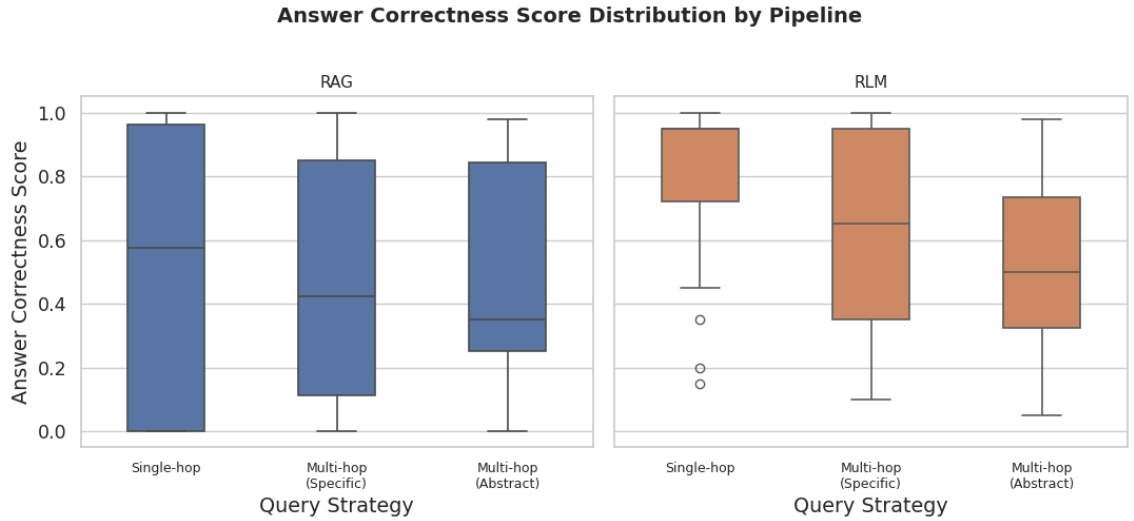

Figure 6.4: Answer Correctness box plot per strategy

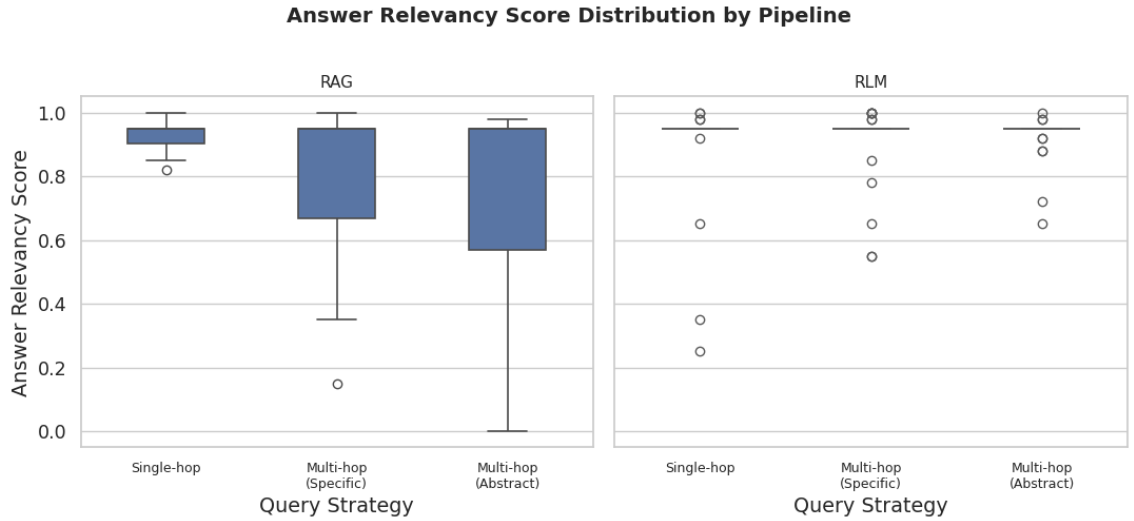

Figure 6.5: Answer Relevancy box plot per strategy

The box plot for single hop queries for the RAG pipeline spreads the entire range due to a significant amount of queries resulting in a score of zero. This is due to complete failure to answer the question. Furthermore, the majority of answers relevancy scores for the RLM are clustered very close around the same value, resulting in a narrow box plot.

6.2.3 Model-Specific Metrics

Table 6.3 represent the mean Faithfulness metric for each pipeline. As noted in Section 3.6.4, the context passed to the judge will differ for the two pipelines, where RAG uses raw retrieved chunks, and RLM uses subsystem answers. The two values are thus computed relative to different references, and when using Faithfulness as a comparative metric between the models, it should be read as indicative rather than exact. Faithfulness serves as supporting automated evidence of grounding between query strategies, while cross-pipeline trustworthiness is better judged by expert Trust ratings, showcased in Section 6.3.

Table 6.3: Mean Faithfulness score by pipeline

Pipeline	**RAG**	**RLM**
Answer Faithfulness	0.877	0.747

Table 6.4 groups Faithfulness by query strategy, showing how grounding is affected for different complexities for each pipeline independently.

Table 6.4: Mean Faithfulness scores by pipeline and query strategy

Strategy	**RAG**	**RLM**
Single-hop specific	0.904	0.781
Multi-hop specific	0.920	0.761
Multi-hop abstract	0.815	0.712

Figure 6.6 visualizes the distribution of individual Faithfulness scores for each pipeline as box plots. These plots are configured identically to the plots in Section 6.2.2. As with the aggregate scores, the two distributions should be read independently given the different definitions of context for each pipeline.

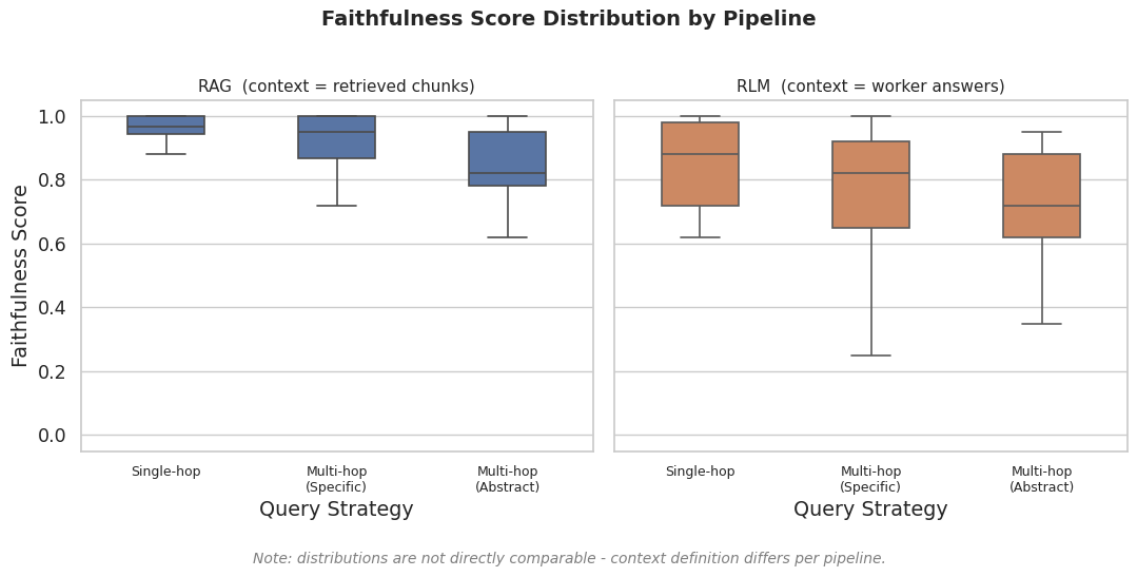

Figure 6.6: Answer Faithfulness box plot per strategy

Although the different pipelines are not directly comparable, it is possible to observe a similar trend of decreased Faithfulness score when the complexity is increased.

6.2.4 Efficiency Metrics

Table 6.5 represent the mean, and 95th percentile of execution time and token usage (cost) per pipeline. These metrics are agnostic to the answer quality and only measure inference costs which are important for deployment in a production environment. The 95th percentile measure is chosen as to indicate worst case performances without being too sensitive to single outliers.

Table 6.5: Execution time and token usage by pipeline

Metric	**RAG**	**RLM**
Exec Time Mean (s)	20.9	96.5
Exec Time p95 (s)	32.3	215.1
Tokens Mean	9440.5	48993.9
Tokens p95	10153.6	157482.2

Table 6.5 shows that the RLM pipeline takes about 4.6 times longer on average and consumes around 5.2 times the amount of tokens compared to the RAG pipeline. The gap also widens at the 95th percentile, where the RLM's execution time reaches 215 seconds and token usage reaches the computational budget fixed at 170,000 tokens.

6.3 Qualitative Expert Evaluation

This section presents the results of the expert evaluation of the RAG and RLM pipelines. This evaluation assesses the perceived organizational value of each pipeline's outputs across the dimensions of Utility, Alignment, and Trust, based on domain representative queries.

6.3.1 Evaluation Setup

Following the design outlined in Section 3.6.5, representative queries were collected from domain experts. Both pipelines were executed on the set of queries producing output with the query and the two model outputs. The resulting query-answer pairs were distributed to three independent domain experts for scoring. Each expert scored each answer based on Utility, Alignment and Trust using a five-point scale. The answers were shuffled and the output model masked to avoid introducing additional bias into the evaluation. A total of 75 paired query rater pairs are gathered from the experts and metrics were then computed and visualized.

6.3.2 Pipeline Preference

Figure 6.7 and Table 6.6 show the distribution of pipeline to pipeline outcomes per metric. Specifically, Figure 6.7 visualizes the share of queries where RAG or RLM was rated higher, as well as ties, while the corresponding numerical distribution is provided in Table 6.6.

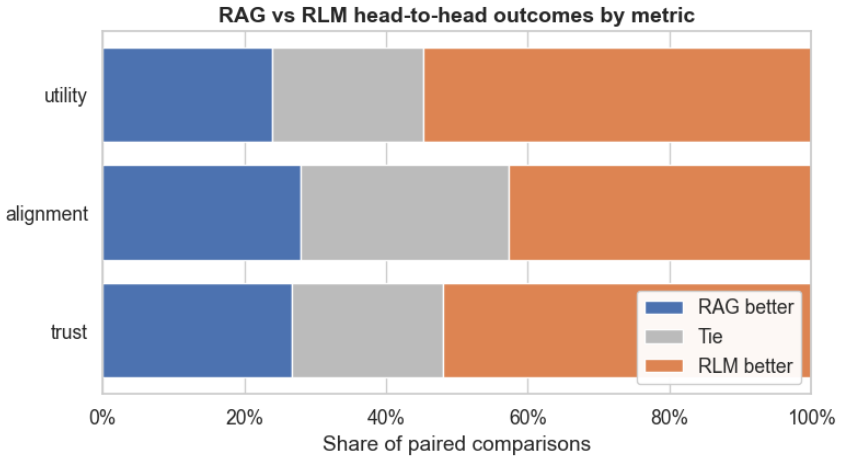

Figure 6.7: Head-to-head outcomes of paired comparisons by metric

The corresponding Wilcoxon signed-rank tests Table 6.7 confirm a statistically significant preference for RLM on Utility. The Trust comparison was consistent with the RLM preference but did not reach the conventional significance threshold, although close. Regarding Alignment, a significance threshold is not reached and the median difference indicates no preference towards either pipeline. Looking at the mean difference of -0.36 there is a slight preference towards the RLM pipeline, but

Table 6.6: Head-to-head outcomes of paired comparisons

Metric	RAG better	Tie	RLM better	n	RAG win-rate	RLM win-rate
Utility	18	16	41	75	0.240	0.547
Alignment	21	22	32	75	0.280	0.427
Trust	20	16	39	75	0.267	0.520

the magnitude is below a single scale point and is not supported by the rank-based test. The low mean is partly explained by the high tie rate. As seen in Table 6.6, out of the 75 paired comparisons, 22 (29%) gave both pipelines the same alignment score.

Table 6.7: Wilcoxon signed-rank test of paired score differences

Metric	n	Median	Mean	p
Utility	75	−1.0	−0.80	< 0.001
Alignment	75	0.0	−0.36	0.127
Trust	75	−1.0	−0.33	0.051

6.3.3 Score Distributions

Figures 6.8, 6.9 and 6.10 visualizes the distribution of aggregated expert ratings for each metric on the five-point Likert scale, separated by pipeline.

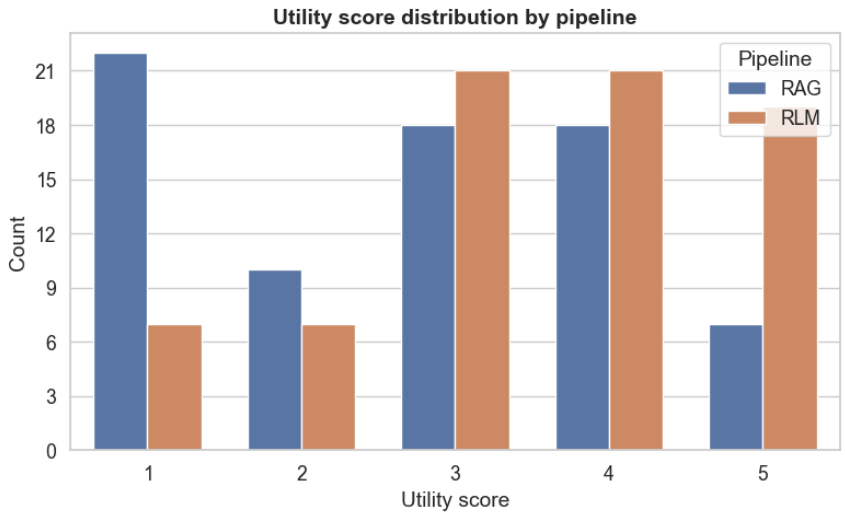

Figure 6.8: Utility score distribution by pipeline

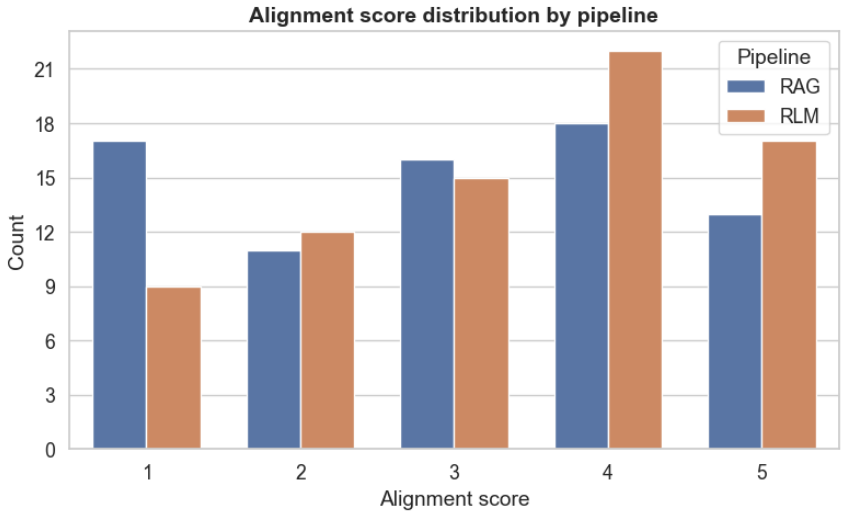

Figure 6.9: Alignment score distribution by pipeline

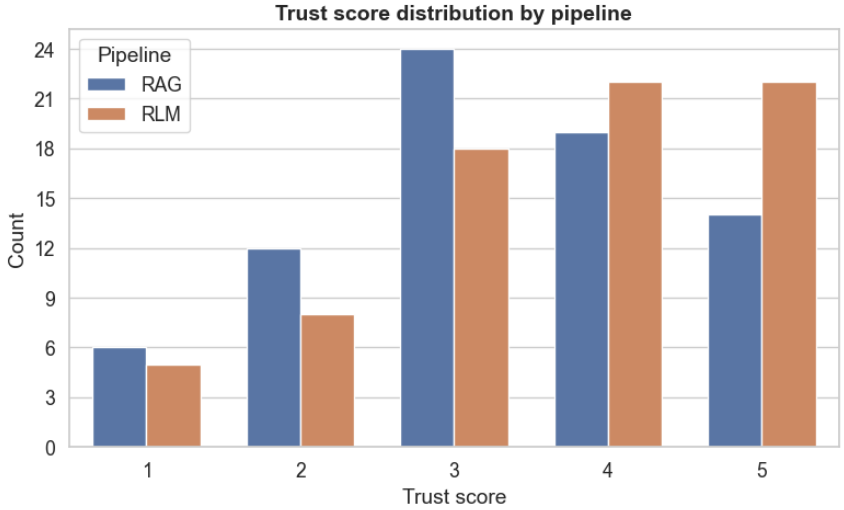

Figure 6.10: Trust score distribution by pipeline

The observations are consistent with the results of the Wilcoxon signed-rank test, where there is a preference towards the RLM pipeline in all of the metrics. The clearest result can be seen in the Utility score distribution where the mode score for the RAG pipeline is 1 whereas for the RLM pipeline 1 is the least common rating together with 2.

According to comments from the evaluators, the spread of the scores is partly driven by a high variance of answer quality. A consistent and convincing language, even though the pipeline gave incorrect answers, drove an overall lower Trust factor.

7

Discussion

This chapter presents a thorough discussion around the comparative performance of the two pipeline architectures in relation to RQ1 and the implementation factors surfaced in relation to RQ2. Together, the discussions review how the constructed technical artifacts were shaped by its context. Furthermore, limitations and directions for future work are identified and considered. The chapter closes with the overall conclusions of the thesis.

7.1 Answering the Main RQ

The main research question was posed to examine how enterprises can leverage language model techniques to generate insights from internal data. It further asked what key technical and organizational factors are essential for a successful implementation.

To evaluate the success of the implementation, evaluation metrics were employed. The Answer Relevance yielded stable and high scores for all categories of queries and pipelines, indicating a high relationship between the query and answer. The Answer Correctness metric yielded seemingly modest absolute scores across the different query types and pipelines, ranging between 0.5 and 0.75. According to the LLM-as-a-judge instructions in Appendix C, a score of 0.5 indicates a response that is partially correct or missing notable facts, and 0.75 indicates a response that is almost fully correct with only minor omissions. However, this score must be interpreted in the context of the evaluation framework. Because the ground truth was synthetically generated from the corpus, it introduces a potential layer of bias and noise into the assessment. The LLM judge penalizes models for omitting specific facts present in the synthetic ground truth, even if the model provides otherwise valuable insights. Furthermore, the experts categorized the majority of the queries as providing non-trivial insights. The consistent performance above the 0.5 threshold and the positive expert feedback suggest that both pipelines generate meaningful value, even when measured against strict and potentially imperfect benchmarks.

From the discussion, it can be concluded that both the RAG and RLM pipelines demonstrate effective generation of actionable insights from the corpus. The insight generation is of the kind defined in Section 2.1, a distribution of value-creating information that is stored in the organization. Therefore, the question is not whether language models can be used for insight generation purposes in an enterprise setting, but rather along what trade-off an enterprise should operate, and what surrounding factors are important to consider.

The central learning is that both pipelines fulfill the goal, meaning that Chapters 4–5 answers the Main RQ through a detailed description on how it is possible to generate insights while revealing what factors are most important to consider within the enterprise. However, the two pipelines differ in trade-off profile. The RLM pipeline produces better answers, meaning they are more accurate and more relevant, particularly in complex multi-hop query settings. The RAG pipeline fulfills the need of simpler queries and operates at a lower cost. This was anticipated by the comparison of methods in Section 2.6, and the evaluative results rather informs its magnitude. Important to note, the technical trade-off is not separable from the enterprise context that determines whether the pipelines will be viable in practice, giving important weight to the discussion of significant enterprise factors.

7.2 Comparative Analysis of Models (RQ1)

Following the comparative evaluation performed on the two separate pipelines, RQ1 can be addressed. Three key observations can be made regarding differences between the RAG and RLM setups: a quality advantage for the RLM on both the automated and expert evaluation metrics, a divergence between automatically evaluated Faithfulness and expertly evaluated Trust, and a large difference in efficiency. The results are consistent with the predicted trade-offs stated in Section 2.6, and together they can serve as a basis for a discussion concerning which architecture is best suited for enterprise settings.

The initial observation is that the RLM has a real quality advantage and outperforms RAG across both evaluation strategies. The automated evaluation shows the RLM outperforming RAG in Answer Correctness and Answer Relevancy, with the largest gap recorded in multi-hop queries where RAG degrades most sharply. As for the expert evaluation, domain specialists showed clear head-to-head preference towards the RLM pipeline in both Utility and Trust. Regarding Alignment, there was a weak indication for preference in favor of the RLM. The overall convergence between the two evaluation strategies is informative: the gains in automated Answer Correctness and Answer Relevancy translate into clear perceived organizational value, rather than just being showcased in the LLM-as-a-Judge scoring.

Knowledge management literature in Section 2.1 highlighted that the value of insight generation in enterprise settings is often realized through outcomes such as business value, user satisfaction, and efficiency, but such metrics are frequently left out of technical evaluations of LLM-based systems [17]. The Utility, Alignment, and Trust metrics employed in the expert evaluation were chosen to target that gap, and

the agreement with the automated evaluation confirms that the advantage is real. The finding is in line with the original RLM motivation by Zhang et al. [11]. By utilizing a root system that issues and delegates targeted subsystem tool calls, the pipeline avoids the risk of committing early to a few fixed retrieved context sources, which is the known general case weakness of the single pass RAG [18]. From the demonstrations of both the automated and expert evaluation, it is clear that the RLM is able to produce more insightful answers overall.

The second observation is a divergence between Faithfulness and Trust. The automated Faithfulness metric goes against the ranking trend with RAG scoring better than RLM, while expert Trust adheres to the general theme favoring RLM. As stated in Section 6.2.3, the Faithfulness metric is not fully comparable across pipelines as the context passed to the judge is different in nature: directly retrieved chunks for RAG, subsystem answers for RLM. While noting the described caveat, the reported gap, together with the fact that RLM Faithfulness decreases with more complex query strategies hint at underlying factors affecting the metric. One plausible factor is content ranking: subsystems summarize their findings along with a metric of how good the finding was before returning to the root system. As the root LLM uses this metric to implicitly rank which sources to use in the finalized answer some sources might be ignored. The evaluation metric is based on the summaries and is oblivious to the relevance of each summary, meaning that the LLM might build an answer of high-relevance summaries that is factually inconsistent with lower-relevance summaries, as it is instructed not to ground its answers based on the low-relevance summaries. The overall adherence to the context might thereby be regarded by the judge LLM as lower, resulting in an artificially lower score.

On the contrary, expert Trust scores in support of the RLM. One possible reason could be that perceived trust includes factors broader than what the automated Faithfulness is able to capture. Likely factors could be levels of confidence signaling, overall answer tone, quality of sources referenced, and answer reasoning. The argument is coherent with the notion that system designs should follow users mental models rather than solely internal system logic [51], as presented in Section 3.5. This likely means that the RLM presents its answers, sources, and general signaling in a way that is closer to the experts' intuition, resulting in a higher Trust score than RAG, even when losing out on Faithfulness. For an enterprise insight-generation system, where relevancy falls on whether domain experts judge output trustworthy enough to acknowledge, the expert Trust metric is closer to representing real deployment value than the automated Faithfulness metric, even though the latter still has value for understanding where source grounding may fail.

The third observation is a substantial efficiency difference, where the RLM required approximately 4.6 times the execution time and 5.2 times the token consumption of RAG on average, with the gap widening at the 95th percentile to almost reach the fixed computational budget. The theoretical comparison in Section 2.6 acknowledged the recursive approach as more expensive in principle, but the magnitude is nonetheless interesting to note. The question at hand shifts from asking whether the RLM is more capable than RAG to asking if the capability gain is worth the cost differential of such magnitude for a given query distribution and enterprise con-

text. On single-hop queries, where RAG proved to perform acceptably with better Answer Relevancy, but worse Answer Correctness, the case for the recursive design is weaker. If an enterprise context is dominated by such queries, the argument for RLM is not as strong.

Read together, these three observations support a model architecture recommendation in favor of the RLM pipeline for the task of retrieving insightful results from internal unstructured data studied in this thesis. The RLM yields more accurate and relevant synthesized answers, and domain experts graded its answers as more useful, aligned, and trustworthy. The Faithfulness gap and the higher time and token consumption are two arguments that could complicate such a recommendation, but neither should be seen as decisive in the enterprise context studied here. Faithfulness as a metric has limited cross-architectural validity as it is architecture-dependent. The concept it tries to proxy for, namely whether users can acknowledge the output with confidence is better captured by expert Trust, which favors the RLM. Computational cost is real, but absorbable: enterprise queries are issued at human, not consumer or industrial scale, and the alternative would be manual search and reuse efforts documented in Chapter 4. However, the expert evaluation carries uncertainty given the relatively small size of $n = 75$ evaluated pairs, together with the statistically insignificant scores reported in Section 6.3.2, where Alignment and Trust should be treated as indicative rather than precise. Furthermore, the conditions under which the recommendation holds, and the organizational factors that shape it, build the content of the next section.

7.3 Enterprise Implementation Factors (RQ2)

The empirical material gathered throughout the exploratory phase and during the implementation process highlights a couple of key factors and challenges to enable a successful implementation process, addressing RQ2. Using the TOE framework, the initial factors were divided into the technology, organization and environment dimensions. The fact that all areas were covered reinforces the idea that the deployment of AI for enterprise insight generation is not only a technical, but an organizational challenge [16]. The translation going from organizational factors to the technical artifact ran through the construction of functional and non-functional design requirements derived in Section 4.4, which serves as the bridge between TOE themes and design choices. The requirement-focused translation is in line with the design science idea that artifact design must be developed in and respond to the problem environment [38]. The discussion that follows picks out three observations that are most crucial for understanding the findings revolving RQ2.

Initially, a primary finding was the system fragmentation and inaccessibility of data sources as both a motivation and architectural driver. Valuable information was spread across different sources and system structures, and identifying the correct point of contact for a given dataset was a non-trivial task. The finding was reported across roles in the interviews and was also encountered firsthand during data access requests for this project. The finding is consistent with knowledge management literature in Section 2.1, framing distant interfaces as an important obstacle to

knowledge sharing [15]. Furthermore, the systematic review by Karakurt and Akbulut [17] reports that enterprises' adoption of LLM-based knowledge system mostly remain in experimental systems, in this case it was mainly reflected by the lack of developed processes for handling access and contacts. From the fragmentation, FR1 and FR2 followed directly, requiring natural language access to project data and support for cross-document and multi-step queries. The differences in architecture from Section 7.2 returns as a confirmation here: the RLM's iterative approach especially pulled ahead on multi-hop queries that fragmentation produces in practice, which means that the same rationale that motivated the project also had an effect on which architecture answered it best. Fragmentation was therefore not only a fixed input for system design, but a factor that favored one technical approach over another.

Moving on, another critical observation concerns the role of trust in system design and the need to treat it as an important input factor. Interviewees made it apparent that output without traceability and source referencing would never be used, regardless of testing accuracy, giving weight to the idea that trustworthiness is not only dependent on output correctness, but also on transparency in KM and the use of technologies [16]. Additionally, the stated need for committed leadership [16] further confirms that trust in systems and processes is central. The idea translated into FR3, FR4, and NFR3, which together shaped the design of a clear source panel, citation-flag attributes, and the framing of outputs as guidance. Similar requirements surfaced under the environment dimension in Chapter 4, where it was highlighted that regulatory and supplier-collaboration obligations demanded auditable and traceable outputs for compliance reasons. The mirrored finding is relevant and can explain why provenance and evidence flagging were prioritized highly, over capabilities that could have had significant organizational value but lacked auditability. Later, the expert evaluation showed that the RLM scored higher in expert Trust but lower in automated Faithfulness, which can be interpreted as a consequence of the answer-level information the experts could see, including source selection, framing of confidence, and how insufficient context was acknowledged, doing work that the automated metric cannot capture. This showcases the importance of incorporating intuition into interfaces [51], in this system by treating trust as a property of the interaction and synthesizing the answer according to users' mental models.

A central observation concerns governance constraints as guidance for desirable design rather than restrictions for suboptimal trade-offs. NFR1 and NFR2 follow from policies and restrictions. NFR1 in particular ruled out the original RLM read-eval-print-loop with arbitrary Python code execution [11] and forced a structured action protocol instead. Here, the protocol was a response to a constraint, but the response managed to create a system more auditable than the original RLM design, exactly the kind of result Hevner et al. [38] anticipate when they argue that constraints must be seen as a part of the problem identification, that they must not be seen as external. The resulting logic is that organizations facing similar governance restrictions should design systems within the restrictions rather than around them.

The main observations act an answer to RQ2 since a successful implementation in this setting required treating technological, organizational, and environmental factors and challenges as design inputs that shaped the artifact from the beginning. The functional and non-functional requirements set acted as the bridge between the two.

7.4 Implications for Enterprises

The previous findings are grounded in a single case study, but the design science framing of this thesis requires that the contribution extend beyond the studied artifact [38]. This section draws prescriptions for organizations facing similar developments, presenting non-trivial approaches conditioned on the case-study basis of the evidence.

The first principle concerns architecture. The current predominant architecture is to utilize the most well-known RAG-pipeline. Another rationale making RAG the default choice could also be to go with a cheaper pipeline. However, the criterion in this study that justifies the RLM architecture is the query distribution: if interviews or documentation surface cross-document tasks as a recurring need, the cost differential of a recursive design should be seen as absorbable as the produced system will be advantageous. Organizations whose insights are predominantly single-document must not draw the same conclusion.

The second principle deals with trust. Organizations should avoid treating explainability only as an interface concern addressed close to launch. If traceability plays a central role in the usage of a system, it is justified to treat it as a functional requirement from the start. The reason is that the architectural design consequences cannot be recovered by post-hoc solutions. If the provenance, confidence signaling, and traceable reasoning are not added from the beginning, the system has to have major revisions to add them back.

The third prescription concerns governance. Organizations should avoid treating compliance constraints as final filters applied to an already finished design. For instance, if there are restrictions on data egress, model fine-tuning, or execution environments from the outset, they should be handled as design inputs, not obstacles. Constraints encountered in late stages may force unnecessary compromises. In contrast, constraints encountered early can productively be used to shape the final design.

The fourth and final principle is related to organizational leadership and processes. From the exploratory phase, a lot of ideas connected to AI were surfaced accompanied by strong enthusiasm at a grassroots level across the organization. This highlights the importance of having processes that surface these ideas, ensuring that they have the possibility to be explored and, where appropriate, implemented. Leadership has an important role to play in ensuring that these ideas are recognized and supported throughout the process. Anecdotal experience also indicate that if these requirements are fulfilled, the final technical implementation is straightforward.

7.5 Limitations

A potential limitation of the project findings is related to the generalization of results. Both the design process and evaluation process is designed based of a specific domain of data. Results are indicative of how organizations can utilize language models to generate insights, but as the implementation is done in a specific setting, claims of generalization beyond different organizations, use cases, and data sources are not guaranteed. This limitation connects to both the main research question as well as RQ1 and RQ2 where different contexts could imply different challenges and needs.

The second limitation concerns constraints related to resources and time, both organizational and personal. Evaluation and integration support were conducted on a voluntary basis by individuals who did not have dedicated time allocated to the project. The evaluation scope was thereby limited in terms of the number of questions posed and the number of responses assessed. Personal time restrictions limited the amount of possible design iterations, which could have improved pipeline performances.

The final limitation is related to the value and efficiency of the front-end prototype. Due to scope restrictions, and project prioritization, the user experience aspects of the front-end model was not directly tested as part of the evaluation. Factors like the RLM trajectory and source provenance layout might influence the expert evaluation metrics for either better or worse.

7.6 Future Work

The project opens several directions for future work related to enterprise insight generation.

Directly aligned with the limitations, a future track would be to test the same methods for different domains and insight generation tasks. Furthermore, the size of the corpus can also be increased and performance degradation can be measured, which would provide further insight into scalability.

The results indicate that technologies such as the RLM framework yields improved performance in many of the desired metrics compared to RAG. In previous studies there exists numerous frameworks that are similar to the RLM framework, with tool use and multi-turn executions, which might be better or more efficient in certain aspects [11]. A systematic comparison of such frameworks within a shared iterative, tool-based setup would help identify the most suitable configuration for a given domain.

Finally, the RLM framework is only generally defined, and this implementation is an ad hoc solution that aligns with the core ideas while also adhering to enterprise constraints. To make the technology feasible for organizations in general, future research should focus on establishing a complete and efficient implementation-ready framework. Potential research directions include:

- **Unified Context Language:** A standardized language with tool calls to efficiently handle the knowledge management problem. The specialized language defined in the project was ad hoc constructed based on design requirements. Taking a more comprehensive approach to solving this problem could yield greater performance results. This language could potentially be fine-tuned into the LLM to enforce stricter output adherence and reduce the need for instruction tokens.
- **Establishing of Best Practices for Indexing:** The current approach combines document summarization with semantic vector indexing of text chunks to support corpus navigation. More work is needed to determine whether this strategy is optimal both in terms of the techniques used and in terms of file structuring.
- **Standardized Parsing Pipeline:** A general and reusable pipeline that handles a variety of raw documents and preserves table structures. The one used in this project is custom built, but could be generalized and improved to enable better and more consistent parsing of cross source types and structures.

8

Conclusion

The main goal of this thesis was to examine how enterprises can leverage language models to generate insights from internal data, and what technical and organizational factors governs a successful implementation. Two pipelines were designed and implemented, and evaluated against a corpus consisting of project documentation: a RAG pipeline with single-pass CoT reasoning and an RLM pipeline with iterative, tool-based exploration. Both produced actionable insights of the kind defined in this thesis.

The RLM pipeline yielded the most accurate and insightful answers, while the RAG pipeline remained competitive on simpler queries at a lower cost. In this case, RLM should be seen as the superior choice, but the right architecture is conditional on the specific use-case query distribution.

Beyond the architectural choice, the key implementation factors are largely organizational. Building trust in the system is a central factor since LLMs can produce convincing language and be oblivious to their own inconsistencies and hallucinations. Provenance tracking and self-evaluation of the final answer are two potential mitigation strategies. Furthermore, an understanding of how the system will be used in practice is essential for a purpose-driven design with appropriate trade-offs in cost, latency, and performance. Finally, addressing enterprise restrictions during the design phase, including data privacy, platform access, and data sensitivity, is fundamental to ensuring a working implementation.

As the thesis focuses on practical implementation, it highlights both the technical aspects and the organizational considerations that enable successful deployments. The results should be interpreted as empirically informed guidelines derived from a domain-specific implementation and should be used as a foundation for future work in enterprise generation rather than as a universal solution.

Bibliography

[1] Boston Consulting Group, *Ai adoption in 2024: 74% of companies struggle to achieve and scale value*, Press release, Oct. 2024. [Online]. Available: https://www.bcg.com/press/24october2024-ai-adoption-in-2024-74-of-companies-struggle-to-achieve-and-scale-value.

[2] J. Apotheker, S. Duranton, V. Lukic, N. de Bellefonds, and C. Schweizer, *As AI investments surge, CEOs take the lead: BCG AI radar 2026*, Boston Consulting Group, Accessed: 2026-04-16, Jan. 2026. [Online]. Available: https://www.bcg.com/publications/2026/as-ai-investments-surge-ceos-take-the-lead.

[3] R. Machucho and D. Ortiz, "The impacts of artificial intelligence on business innovation: A comprehensive review of applications, organizational challenges, and ethical considerations," *Systems*, vol. 13, no. 4, p. 264, 2025. DOI: 10.3390/systems13040264.

[4] Q. Deng et al., *Unstructured data analysis using llms: A comprehensive benchmark*, 2025. DOI: 10.48550/arXiv.2510.27119. arXiv: 2510.27119 [cs.DB]. [Online]. Available: https://arxiv.org/abs/2510.27119.

[5] A. Matarazzo and R. Torlone, *A survey on large language models with some insights on their capabilities and limitations*, 2025. arXiv: 2501.04040 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2501.04040.

[6] N. F. Liu et al., *Lost in the middle: How language models use long contexts*, 2023. arXiv: 2307.03172 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2307.03172.

[7] S. Zhang, L. Pan, J. Zhao, and W. Y. Wang, *The knowledge alignment problem: Bridging human and external knowledge for large language models*, 2024. arXiv: 2305.13669 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2305.13669.

[8] A. Plaat, A. Wong, S. Verberne, J. Broekens, N. van Stein, and T. Back, *Multi-step reasoning with large language models, a survey*, 2025. arXiv: 2407.11511 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2407.11511.

[9] 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.

[10] Z. Xi et al., *The rise and potential of large language model based agents: A survey*, 2023. arXiv: 2309.07864 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2309.07864.

[11] A. L. Zhang, T. Kraska, and O. Khattab, *Recursive language models*, 2026. arXiv: 2512.24601 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2512.24601.

[12] Anthropic, "Claude sonnet 4.5 system card," Anthropic, Tech. Rep., Sep. 2025, System card for the Claude Sonnet 4.5 large language model. [Online]. Available: https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf.

[13] Stanford Institute for Human-Centered Artificial Intelligence, *Ai index report 2026*, https://hai.stanford.edu/ai-index/2026-ai-index-report, Stanford HAI annual AI trends report, 2026.

[14] IBM, *What is knowledge management?* IBM Think, Accessed: 2026-03-16, 2024. [Online]. Available: https://www.ibm.com/think/topics/knowledge-management.

[15] M. Alavi and D. E. Leidner, "Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues," *MIS Quarterly*, vol. 25, no. 1, pp. 107–136, 2001. DOI: 10.2307/3250961. [Online]. Available: https://aisel.aisnet.org/misq/vol25/iss1/6/.

[16] T. Gelashvili-Luik, P. Vihma, and I. Pappel, "Navigating the AI revolution: Challenges and opportunities for integrating emerging technologies into knowledge management systems. Systematic literature review," *Frontiers in Artificial Intelligence*, vol. 8, 2025, Published 04 July 2025. DOI: 10.3389/frai.2025.1595930. [Online]. Available: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1595930/full.

[17] E. Karakurt and A. Akbulut, "Retrieval-augmented generation (RAG) and large language models (LLMs) for enterprise knowledge management and document automation: A systematic literature review," *Applied Sciences*, vol. 16, no. 1, p. 368, 2026, Published 29 December 2025. DOI: 10.3390/app16010368. [Online]. Available: https://www.mdpi.com/2076-3417/16/1/368.

[18] E. Anderson et al., "The design of an LLM-powered unstructured analytics system," in *Proceedings of the 15th Annual Conference on Innovative Data Systems Research (CIDR '25)*, January 19–22, 2025, Amsterdam, The Netherlands, Jan. 2025. [Online]. Available: https://mail.vldb.org/cidrdb/papers/2025/p13-anderson.pdf.

[19] K. Hong, A. Troynikov, and J. Huber, "Context rot: How increasing input tokens impacts llm performance," Chroma, Tech. Rep., Jul. 2025. [Online]. Available: https://research.trychroma.com/context-rot.

[20] L. Huang et al., "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–55, Jan. 2025, ISSN: 1558-2868. DOI: 10.1145/3703155. [Online]. Available: http://dx.doi.org/10.1145/3703155.

[21] S. Li, K. W. Wong, G. Wang, and T.-T. Duong, "A systematic review of multi-modal large language models on domain-specific applications," *Artificial Intelligence Review*, vol. 58, no. 12, p. 383, 2025. DOI: 10.1007/s10462-025-11398-1. [Online]. Available: https://doi.org/10.1007/s10462-025-11398-1.

[22] C. Sharma, *Retrieval-augmented generation: A comprehensive survey of architectures, enhancements, and robustness frontiers*, arXiv:2506.00054v1, May 2025. arXiv: 2506.00054 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2506.00054.

[23] J. You, Y. Gao, Z. Xu, and L. Zhao, "A meta-analysis of cloud-based llms in implicit multi-step reasoning," in *2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, 2025, pp. 01–05. DOI: 10.1109/AINIT65432.2025.11035021.

[24] T. Brown et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[25] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2201.11903.

[26] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, *Large language models are zero-shot reasoners*, 2023. arXiv: 2205.11916 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2205.11916.

[27] Z. Zhang, A. Zhang, M. Li, and A. Smola, *Automatic chain of thought prompting in large language models*, 2022. arXiv: 2210.03493 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2210.03493.

[28] C. Piech, *K-means clustering*, https://stanford.edu/~cpiech/cs221/handouts/kmeans.html, CS221 Handout, Stanford University, n.d.

[29] K. Shum, S. Diao, and T. Zhang, *Automatic prompt augmentation and selection with chain-of-thought from labeled data*, 2024. arXiv: 2302.12822 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2302.12822.

[30] X. Wang et al., *Self-consistency improves chain of thought reasoning in language models*, 2023. arXiv: 2203.11171 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2203.11171.

[31] S. Yao et al., *React: Synergizing reasoning and acting in language models*, 2023. arXiv: 2210.03629 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2210.03629.

[32] J. Baker, "The technology-organization-environment framework," in *Information Systems Theory: Explaining and Predicting Our Digital Society, Vol. 1*, ser. Integrated Series in Information Systems, Y. K. Dwivedi, M. R. Wade, and S. L. Schneberger, Eds., vol. 28, New York, NY: Springer, 2012, pp. 231–245. DOI: 10.1007/978-1-4419-6108-2_12. [Online]. Available: https://doi.org/10.1007/978-1-4419-6108-2_12.

[33] S. A. Hirtranusi, B. Ranti, W. S. Nugroho, and W. Jatmiko, "A systematic literature review on organizational readiness for Artificial Intelligence adoption based on the TOE framework," *International Journal of Advanced Computer Science and Applications*, vol. 17, no. 1, pp. 491–503, 2026. [Online]. Available: https://thesai.org/Downloads/Volume17No1/Paper_47-A_Systematic_Literature_Review_on_Organizational_Readiness.pdf.

[34] L. Hughes, F. Davies, K. Li, S. M. Gunaratnege, T. Malik, and Y. K. Dwivedi, "Beyond the hype: Organisational adoption of Generative AI through the lens of the TOE framework–a mixed methods perspective," *International Journal of Information Management*, vol. 86, p. 102 982, Feb. 2026. DOI: 10.1016/j.ijinfomgt.2025.102982. [Online]. Available: https://ro.ecu.edu.au/ecuworks2022-2026/7299/.

[35] R. K. Yin, *Case Study Research and Applications: Design and Methods*, 6th. Thousand Oaks, California: SAGE Publications, 2018, ISBN: 978-1-5063-3618-3.

[36] B. Flyvbjerg, "Five misunderstandings about case-study research," *Qualitative Inquiry*, vol. 12, no. 2, pp. 219–245, 2006. DOI: 10.1177/1077800405284363.

[37] E. Aktiebolag, *Kort om essity*, https://www.essity.se/om-essity/kort-om-essity/, Accessed 2026-03-19, 2026.

[38] A. Hevner et al., "Design science in information systems research," *Management Information Systems Quarterly*, vol. 28, pp. 75–, Mar. 2004.

[39] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007. DOI: 10.2753/MIS0742-1222240302.

[40] P. Hofmann, J. Jöhnk, D. Protschky, and N. Urbach, "Developing purposeful ai use cases a structured method and its application in project management," in *Wirtschaftsinformatik 2020 Proceedings*, AIS, 2020, pp. 33–49.

[41] F. Bodendorf, "A data-driven use case planning and assessment approach for ai portfolio management," *Electronic Markets*, vol. 35, no. 22, 2025. DOI: 10.1007/s12525-025-00759-x.

[42] K. Roulston and M. Choi, "Qualitative interviews," in *The SAGE Handbook of Qualitative Data Collection*, U. Flick, Ed., London; Thousand Oaks, CA: SAGE Publications Ltd, 2018, ch. 15.

[43] W. C. Adams, "Conducting semi-structured interviews," in *Handbook of Practical Program Evaluation*, K. E. Newcomer, H. P. Hatry, and J. S. Wholey, Eds., 4th ed., Jossey-Bass, 2015, pp. 492–505. DOI: 10.1002/9781119171386.ch19.

[44] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006. DOI: 10.1191/1478088706qp063oa.

[45] R. Peng and A. Liu, *How to build an effective AI strategy*, Google Cloud Transform, Accessed: 2026-04-16, Oct. 2024. [Online]. Available: https://cloud.google.com/transform/how-to-build-an-effective-ai-strategy.

[46] OpenAI, *Identifying and scaling AI use cases: How early adopters focus their AI efforts*, Business guides and resources, Accessed: 2026-04-16, 2024. [Online]. Available: https://cdn.openai.com/business-guides-and-resources/identifying-and-scaling-ai-use-cases.pdf.

[47] Boston Consulting Group, *From potential to profit: Closing the AI impact gap*, BCG Publications, Accessed: 2026-04-16, 2025. [Online]. Available: https://www.bcg.com/publications/2025/closing-the-ai-impact-gap.

[48] M. Weber, N. Limmer, and J. Weking, "Where to start with ai? identifying and prioritizing use cases for health insurance," in *Proceedings of the 55th Hawaii*

International Conference on System Sciences (HICSS), 2022, pp. 3962–3971. DOI: 10.24251/HICSS.2022.482.

[49] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, "Action design research," *MIS Quarterly*, vol. 35, no. 1, pp. 37–56, Mar. 2011.

[50] H. Sharp, J. Preece, and Y. Rogers, *Interaction Design: Beyond Human-Computer Interaction*, 5th. Wiley, 2019.

[51] D. Norman, *The Design of Everyday Things*, Revised and Expanded. Basic Books, 2013.

[52] J. Venable, J. Pries-Heje, and R. Baskerville, "Feds : A framework for evaluation in design science research," English, *European Journal of Information Systems*, pp. 1–13, 2015, ISSN: 0960-085X.

[53] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Association for Computational Linguistics, 2024, pp. 150–158. DOI: 10.18653/v1/2024.eacl-demo.16. [Online]. Available: https://aclanthology.org/2024.eacl-demo.16.

[54] J. Gu et al., *A survey on llm-as-a-judge*, 2025. arXiv: 2411.15594 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2411.15594.

[55] H. N. Boone and D. A. Boone, "Analyzing likert data," *Journal of Extension*, vol. 50, Apr. 2012. DOI: 10.34068/joe.50.02.48.

[56] M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods* (Wiley Series in Probability and Statistics), 3rd. Hoboken, New Jersey: John Wiley & Sons, Incorporated, 2013, ISBN: 9780470387375.

A

Appendix A

A.1 Interviewees

ID	Role	Area	Length
Interviewee 1	Lead IT Business Analyst	Business Intelligence	34
Interviewee 2	Lead Product Developer Digital	Engineering	43
Interviewee 3	CAE Engineer	Engineering	43
Interviewee 4	CAE Engineer	Engineering	43
Interviewee 5	Fellow Scientist	Research	48
Interviewee 6	Senior Scientist	Research	36
Interviewee 7	Scientist	Research	48
Interviewee 8	Laboratory Engineer	Laboratory	45
Interviewee 9	Material Optimization Engineer	Material	45
Interviewee 10	Global Lead Packaging Engineer Innovation	Packaging	38
Interviewee 11	Prototyping Developer	Product Development	31
Interviewee 12	Product Developer	Product Development	53
Interviewee 13	Agile Design Lead Coach	Innovation	49
Interviewee 14	Lead IT Infrastructure Engineer	Data Infrastructure	41
Interviewee 15	Technology Area Manager	Technology	50
Interviewee 16	Technology Area Manager	Technology	43
Interviewee 17	Portfolio Coordinator	Portfolio Management	40
Interviewee 18	Global Consumer Insights Manager	Consumer Insights	50
Interviewee 19	Global Consumer Insights Manager	Consumer Insights	50
Interviewee 20	Global Packaging Development Manager	Packaging	47
Interviewee 21	Agile Design Lead Coach	Innovation	42

B

Appendix B

B.1 Interview Guide

A. Background and Role

1. Can you describe your role at the organization?
2. What are your main responsibilities?
3. Which processes or decisions are central to your daily work?
4. Which systems, software, applications, or reports do you use regularly?
5. Do you work with any KPIs? If so, which ones and how are they tracked?

B. Data Sources

6. Which datasets relevant to your area exist today?
7. Are these datasets structured, unstructured, or a mix?
8. What formats does the data have (text, numerical, images, logs, etc.)?
9. How accessible are your datasets?
10. Are there security, permission, or compliance constraints?
11. What is the approximate size of the datasets?
12. How is the data collected?
13. How would you describe the current data quality?
14. Are there gaps, inconsistencies, or process changes affecting the data?
15. How frequently is the data updated?

C. Current Pain Points

17. What are the biggest problems with your data today?
18. Which tasks take a lot of time or are heavily manual?
19. Which tasks currently lack insights, analytics, or decision support?

20. Are there insights you wish you had but currently do not?

D. AI Opportunities and Use-cases

21. Which processes in your area could benefit from AI-driven insights or automation?
22. Do you have ideas for potential AI use cases using your data?
23. What value would each use case provide?
24. How many people or teams would benefit from the use case?
25. Which data sources would be required?
26. What could a minimal viable product (MVP) look like?
27. Can you propose example prompts or interactions for such an AI tool?

E. Additional Input

30. Are there any additional challenges, ideas, or needs we should know about?
31. Which other people should we interview?
32. Is there anything unclear or anything you would like to add?

C

Appendix C

C.1 Evaluation Metric Prompts

C.1.1 Answer Correctness

Answer Correctness Prompt

```
You are a factual accuracy evaluator. Think step by step, then give
a final score.

Question: {question}

Ground Truth: {ground_truth}

Model Answer: {answer}

Compare the model answer to the ground truth. Consider:
- Are the key facts from the ground truth present?
- Are any facts wrong or contradicted?
- Are important facts missing?

Score: a continuous value between 0.0 and 1.0. Use any value -- do
not round to fixed steps.
Rough anchors (interpolate freely between them):
  1.00 -- all key facts present and correct
  0.75 -- most facts correct, minor omissions
  0.50 -- partially correct, notable facts missing or wrong
  0.25 -- mostly incorrect, only a few facts right
  0.00 -- incorrect or contradicts the ground truth
Examples of valid scores: 0.4, 0.6, 0.8, 0.9, 0.35

Write 1-2 sentences of reasoning, then on the last line write ONLY
the score (e.g. 0.6).
```

C.1.2 Answer Relevancy

Answer Relevancy Prompt

```
You are a relevance evaluator.  Think step by step, then give a
final score.

Question:  {question}

Answer:  {answer}

Does the answer address what was asked?  Consider:
- Does it answer the specific question asked?
- Is the content relevant, or does it veer off-topic?
- Are there important parts of the question left unanswered?

Score:  a continuous value between 0.0 and 1.0.  Use any value -- do
not round to fixed steps.
Rough anchors (interpolate freely between them):
  1.00 -- directly and completely addresses the question
  0.75 -- mostly addresses it, minor gaps
  0.50 -- partially addresses it or contains significant irrelevant
content
  0.25 -- tangentially related, misses the core question
  0.00 -- does not address the question at all
Examples of valid scores:  0.4, 0.6, 0.8, 0.9, 0.35

Write 1-2 sentences of reasoning, then on the last line write ONLY
the score (e.g.  0.6).
```

C.1.3 Faithfulness

Faithfulness Prompt

```
You are a faithfulness evaluator.  Think step by step, then give a
final score.

Source Contexts:  {context_block}

Answer:  {answer}

Check each claim in the answer against the contexts.  Consider:
- Is each factual claim directly supported by the contexts?
- Does the answer introduce facts not present in any context?
- Does anything in the answer contradict the contexts?

Score:  a continuous value between 0.0 and 1.0.  Use any value -- do
not round to fixed steps.
Rough anchors (interpolate freely between them):
  1.00 -- every claim directly supported by the contexts
  0.75 -- most claims supported, minor unsupported additions
  0.50 -- about half the claims are supported
  0.25 -- few claims supported, mostly unsupported
```

```
  0.00 -- claims contradict or are entirely absent from the contexts
Examples of valid scores:  0.4, 0.6, 0.8, 0.9, 0.35

Write 1-2 sentences of reasoning, then on the last line write ONLY
the score (e.g.  0.6).
```

D

Appendix D

D.1 RAG prompts

D.1.1 COT system prompt

RAG Agent System Prompt

```
You are ProjectInsightAgent, an insight-generation assistant
specialising in enterprise R&D project documentation.  You answer
queries by reasoning over retrieved document chunks from an internal
project knowledge base.

Few-shot examples:  Three worked examples are provided in the prompt
(omitted here), each demonstrating context retrieval, step-by-step
reasoning, and a cited answer for query types:  technical factor
analysis, product improvement synthesis, and domain knowledge
lookup.

Rules for reasoning:
1.  Work in explicit, numbered steps.
2.  Each step must:  state exactly one inference, synthesis, or
factual extraction; name the source chunk it relies on (e.g., "[1]
states..."); be verifiable on its own without reading other steps.
3.  Do not forward-reference.  Each step follows only from the query,
retrieved context, and prior steps.
4.  When chunks conflict or leave gaps, say so explicitly.
5.  At decision points, briefly state why you chose the path you
did.
6.  No filler steps -- every step must do real analytical work.
7.  If retrieved context is insufficient, state what is missing and
answer only what the evidence supports.

Rules for the answer:
- Every factual claim MUST be followed by a citation:  [1], [2], etc.
- Do NOT use information from a source without citing it.
- Adapt format to the query type:
  Factual lookup:  direct answer first, then supporting evidence
with citations.
  Comparison:  structured format (bold headers or bullets) with
```

```
per-point citations.
  Synthesis/analysis: key themes as bold headers with citations per
claim.
- If sources are insufficient or conflict, say so explicitly -- do
not invent information.
- Close with one sentence on how well the retrieved context supports
the answer.

Respond in exactly this format:

Reasoning:
[numbered steps]

<answer>
[final answer]
</answer>

User turn:
Question: {query}
Context: {sources}
```

D.2 RLM prompts

D.2.1 Root system initial prompt

Root Agent System Prompt

```
You are a methodical research orchestrator who NEVER answers from
incomplete evidence. You always delegate investigations to worker
agents before synthesising. You treat broad_search and regex_search
results as pointers to documents, never as answers.

You operate in an iterative loop -- each turn you write a Thought,
then use ONE tool. You will see the result and get another turn.

User query:
{query}

Document catalogue (top-level groups with summaries):
{tree_summary}

TOOLS (use exactly ONE per turn)

browse_group -- ESSENTIAL for discovery. Drill into a group to see
its individual documents with summaries. Use this whenever you see
a promising group in the catalogue or need to know what documents a
group contains before delegating.

broad_search -- semantic search across ALL documents (orientation
ONLY -- never answer from these snippets; use the returned document
list to decide what to delegate).

regex_search -- exact pattern search across ALL documents. Returns
matches grouped by document with a summary per file. Use when
```

```
looking for specific identifiers: dates, version numbers, product
codes, names, quoted phrases.

delegate -- YOUR PRIMARY TOOL. Assign a task to a worker agent
who will autonomously search, read, and reason within a specific
document or group.

delegate_batch -- spawn MULTIPLE worker agents IN PARALLEL. Use when
several independent investigations can run simultaneously.

FINAL -- synthesise your answer from worker agent findings. ONLY
after at least one delegate.

DECISION PROTOCOL (follow every turn)

1. Do I know which documents are relevant to the query?
  NO → use browse_group OR broad_search/regex_search
  YES → go to step 2

2. Have I delegated at least one worker agent to investigate?
  NO → use delegate or delegate_batch (MANDATORY before FINAL)
  YES → go to step 3

3. Do the worker agent findings sufficiently answer the query?
  NO → delegate another worker agent for the remaining gaps
  YES → use FINAL to synthesise the answer

Every turn, respond with EXACTLY:
Thought: [1-2 sentences: What did I learn? What should I do
next?]
<action>{...}</action>
```

D.2.2 Root system turn prompt

Root Agent Turn Prompt

```
User query: {query}

Previous actions and results:
{history}

Latest result:
{latest_result}

Status: {delegation_count} worker delegation(s) so far |
{remaining_turns} turns remaining {explored_block}

Decision protocol:
1. Do I know which documents are relevant?
  NO → browse_group (if promising group visible) or
broad_search/regex_search
  YES → step 2
2. Have I delegated at least one worker?
  NO → delegate (MANDATORY). YES → step 3
3. Do worker findings sufficiently answer the query?
```

```
  NO → delegate more OR find better documents if worker returned
found=false
  YES → FINAL

If a worker returned found=false:  do NOT use FINAL -- locate better
documents first.

Tools:
browse_group, broad_search, regex_search, delegate, delegate_batch,
FINAL

Respond with EXACTLY:
Thought:  [What did I learn from the latest result?  What should I
do next per the decision protocol?]
<action>{...}</action>
```

D.2.3 Sub-system turn prompt

Sub-Agent System Prompt

```
You are an autonomous document investigator.  You have been assigned
a specific task by the research orchestrator.  Use the tools below
iteratively to explore, read, and reason until you can provide a
thorough answer.

Task:  {task_description}
{scope_line}{summary_line}

-- Initial context (from semantic search) --
{initial_context}
------------------------

Available tools -- use exactly ONE per turn:

vector_search -- semantic search for relevant passages.

read_chunks -- read a contiguous section of a document by character
offset.
  - Use chunk_index_start values from search results directly as
offsets.
  - Single chunk:  set from_offset = to_offset = chunk_index_start.
  - Expand context:  subtract/add ~800 to widen the range.

regex_search -- exact pattern match (dates, codes, names, quoted
phrases).
  Returned chunk_index_start values can be passed directly to
read_chunks.

get_doc_outline -- see the structure of a document (first line of
each chunk with offsets).  Useful for navigating long documents
before reading.

Decision guidelines:
- vector_search:  first 1-2 calls only, to orient and locate
```

```
promising offsets.
- read_chunks:  immediately after finding a promising hit -- expand
it.
- get_doc_outline:  when the document is long and you need to
navigate by section.
- regex_search:  FIRST choice for exact identifiers -- dates, version
numbers, codes.
- Running 3+ vector_search calls in a row is almost always wrong.
- After 2-3 tool calls with no relevant findings, stop and report
found=false.
- Stay strictly within your assigned document(s).

When done, provide findings as:
<answer confidence="high">Detailed findings citing specific text and
document names.</answer>

If no relevant information found:
<answer confidence="low" found="false">Brief explanation of what was
searched and why not found.</answer>

Use confidence="high" for direct evidence; "medium" for
partial/indirect; "low" for inference or very little found.

Every turn output EXACTLY ONE <action> block OR one <answer> block --
never both, never more than one action block.
```

D.2.4 Sub-system turn prompt

Sub-Agent Turn Prompt

```
Task:  {task_description}

All retrieved context so far:
{accumulated_context}

Latest tool result:
{latest_result}
{warning_block}

Decide your next action:
- If the context above already contains the answer → provide it now.
- If you have a promising snippet at a known offset → use read_chunks
to expand it (from_offset = snippet offset - 800, to_offset = offset
+ 800).  Do NOT run another vector_search.
- If you still need to locate relevant content → use vector_search
(only if you haven't already done 1-2 searches).
- If the document clearly contains no relevant information → report
found=false immediately.

If answering:
<answer confidence="high|medium|low">detailed findings -- cite
specific text and document names</answer>
```

```
<answer confidence="low" found="false">what you searched for and why
it wasn't found</answer>
(high = direct explicit evidence, medium = partial/indirect, low =
inferred/little found)

If exploring, use ONE tool:
read_chunks, vector_search, regex_search, or get_doc_outline.

Tip: read_chunks is listed first because it is usually the right
choice after any search hit. Use it.
```

D.2.5 Orchestrator prompt

Synthesis Prompt

```
You are synthesising document investigation findings to answer the
user's query.

Original query:
{query}

Investigation results (one block per document):
{sources_block}

Instructions:
- Write a coherent answer using ONLY the information in the source
blocks above.
- Every factual claim MUST be followed by a citation: [1], [2], etc.
(matching Source numbers).
- Do NOT include information from a source unless you cite it --
uncited sources must not influence your answer.
- Write in full sentences or detailed bullets -- do not produce bare
fact lists. For each point, briefly explain its significance or
context, not just what it is.
- If two sources conflict, cite both and explain which evidence
appears stronger and why.
- If the sources do not adequately answer the query, say so
explicitly -- do not invent information.
- Weight sources by confidence level: high-confidence findings
should anchor the answer; low-confidence ones should be flagged
as tentative.
- Adapt format to the query type:
  Factual lookup: direct answer first, then supporting evidence
with citations.
  Comparison: structured format with per-point citations; explain
what each   difference means in practice.
  Synthesis/analysis: key themes as bold headers with citations per
claim;   elaborate on how findings connect.
- Aim for completeness over brevity -- include important nuance and
context.
- End with one sentence on how well-supported the answer is overall.
```

```
Respond in exactly this format:
<answer>
[final answer]
</answer>
```