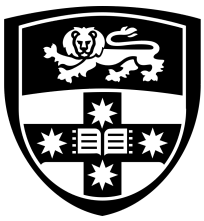


THE UNIVERSITY OF
SYDNEY

Does Size Really Matter?

A Comparative Analysis of Model Scaling across Domain-Specific Retrieval-Augmented Generation Architectures at Enterprise Scales

Submitted by

Adam David Schildkraut

Under the Supervision of

Academic Supervisor: Dr Mehala Balamurali
Industry Supervisors: Mr Oscar Fawkes
and Mr Jeremy Smith

Submitted to

School of Aerospace, Mechanical and Mechatronic Engineering
Faculty of Engineering
The University of Sydney
Australia

Submitted in fulfilment of the requirements for the degree of
Bachelor of Engineering (Honours) in Aerospace Engineering

December 2025

Declaration

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person, nor material which has been accepted, in whole or in part, for the award of any other degree or diploma at this or any other institution of higher learning, except where due acknowledgement has been made in the text. My contributions are as follows;

I conducted the primary research on State of the Art Retrieval-Augmented Generation (RAG) architectures under the guidance of my Academic Supervisor, Dr Mehala Balamurali, and Industry Supervisors, Oscar Fawkes and Jeremy Smith. I conducted the literature review, synthesising research across language model architectures, graph theory, and information retrieval.

I designed and implemented a custom Python-based RAG experimentation platform that performs data ingestion, indexing, and evaluation phases, including optical character recognition processing using Amazon Web Service's (AWS) Textract for image attachments. I created an experimental benchmark for Constantinople's internal Confluence workspace and confirmed the implementation through external validation using Amazon's public Haystack benchmark.

I performed experimental runs, collecting and analysing over 3,200 data points across cost, latency, and quality dimensions using the RAGAs framework. These runs were performed across four distinct retrieval architectures; Vector RAG, Vector RAG with Reranking, Hybrid RAG, and GraphRAG. These architectures were implemented using technologies including ChromaDB, Neo4j, and AWS Bedrock.

I analysed the experimental data to identify key phenomena, including the "Sufficiency Threshold" which indicates that smaller, cost-optimised models have reached a capability level sufficient for complex RAG tasks. In addition to the "Maintenance Trap" in high-frequency update scenarios.

I proposed the "Information Saturation" hypothesis as a theoretical framework to explain observed diminishing returns, noting explicitly that this hypothesis remains unvalidated by the present study, and conducted environmental and ethical analysis of RAG architectures, quantifying carbon emissions across model configurations and identifying ethical risks and mitigation strategies for responsible deployment.

I declare that while generative Artificial Intelligence (AI) tools were used for code assistance and copy-editing, the intellectual contribution, experimental design, and analytical conclusions are entirely my own.

Adam David Schildkraut

Acknowledgements

I would like to express my deepest gratitude to everyone who has supported me throughout my academic journey, and the completion of this thesis at Constantinople.

First and foremost, I am immensely grateful to my parents for their unconditional support in every way possible. A special mention to my mum, Natasha, who is an unwavering advocate for me and has been a constant source of advice, encouragement, and love.

To my friends Sam, Dean, Abby, Ellie, Emma, and Lara, thank you for your endless encouragement and for being there for me through it all. It means more than words can express.

I am deeply thankful to my supervisor, Mehala Balamurali, for her guidance and support throughout this thesis. She consistently encouraged me to bring my best self each week, and challenged me to grow throughout this process. I would not have been able to complete this thesis without her. I would also like to thank the E12, Dalyell, and ESIPS scholarship programs for their financial assistance, which enabled me to focus on my studies without financial stress.

I owe a particular debt of gratitude to my industry supervisors, Oscar Fawkes and Jeremy Smith. Beyond their roles as supervisors, their mentorship and friendship provided a foundation of support that allowed me to thrive. They are without a doubt the most intelligent and driven duo I have had the pleasure of knowing, and their enthusiasm was infectious. Working alongside you has been the highlight of my placement; the lessons I have learned from observing your work ethic and dedication are ones I hope to reflect and carry forward into my own career.

I would like to thank Alan Nguyen, who welcomed me warmly to the company during my placement. His ability to lead Jake, Mohitha and myself in the AI Squad is something to be admired! To Richard Lim, Benjamin Staite, Fergus Charles, Chester Burns, Kaito Nakamichi, Bennett Chapman, and Ross Bevins Cameron, who welcomed me enthusiastically to Constantinople and made my time there extremely rewarding, I want to say a massive heartfelt thank you.

I would also like to extend my appreciation to my colleagues Jake Marsden and Mohitha Mohan, who were by my side from start to finish, I could not have asked for a better team to work with.

To Constantinople, and especially its founders, Di Challenor and Macgregor Duncan, I extend a special thank you for supporting the ESIPS program and for providing me with the opportunity to contribute to the company's future.

Lastly, I am thankful to everyone who has directly or indirectly contributed to this research. Your assistance, support, and encouragement have enabled me to complete a thesis of which I am extremely proud.

Abstract

This thesis investigates the operational economics of a Retrieval-Augmented Generation (RAG) system in dynamic enterprise environments, challenging the prevailing assumption that larger models and complex architectures inherently yield superior performance. We evaluated four architectural variants, Vector RAG, Vector + Reranker, Hybrid RAG, and GraphRAG, across four embedding models and five LLMs accessed via AWS Bedrock. The experimental corpus comprised 10GB of technical documentation spanning 12,214 documents processed into 63,205 chunks. To ensure reproducibility and transparency, we validated key findings on the public `AmazonScience/document-haystack` benchmark.

The results reveal three findings that challenge conventional assumptions. First, we identify a "diminishing returns phenomenon" where the budget-tier embedding model, Titan v2 at $0.02 per million tokens, achieved statistically indistinguishable retrieval quality from the premium Cohere v3 model priced at $0.10 per million tokens for synthesis-oriented queries. No significant differences emerged across architectures, with all p-values exceeding 0.05 and Bayes Factors greater than 3. However, external validation on Haystack demonstrates that precision-oriented entity extraction tasks require premium embeddings to achieve 100% accuracy versus 92% for budget models, establishing task-dependent model selection guidance. Second, we demonstrate a "Sufficiency Threshold" where the smallest tested LLM achieved a RAGAs score of 0.381, matching the Claude Sonnet 4.5 score of 0.373 in GraphRAG answer relevance while reducing indexing costs by a factor of 72 from $292.98 to just $4.05. Third, we identify a "Maintenance Trap" where GraphRAG re-indexing costs under daily update scenarios can reach $58.60 per update and exceed $21,000 annually. This renders graph-based architectures cost-prohibitive for dynamic corpora without cost-optimised model selection.

These findings suggest that RAG architecture selection is fundamentally a problem of operational sustainability rather than pure retrieval quality. We propose a decision framework that prioritises update frequency and query type where Vector RAG with budget embeddings suits dynamic corpora requiring synthesis-oriented search, while GraphRAG with Nova Lite provides superior multi-hop reasoning for low-update, entity-rich knowledge bases at manageable cost. By establishing these economic boundaries, this thesis demonstrates that effective RAG systems need not be expensive RAG systems. This suggests, in the end, that size does not matter.

Executive Summary

Retrieval-Augmented Generation (RAG) bridges the gap between Large Language Models and proprietary enterprise data. While engineering teams often assume that maximising performance requires the largest models and most complex architectures, this research proves otherwise. By evaluating the cost-performance trade-offs of Vector RAG, Vector plus Re-ranker, Hybrid RAG, and GraphRAG across varying model sizes and update frequencies, this study reveals that bigger is rarely better.

Key Findings

Tests on a 10GB technical documentation corpus containing 63,205 chunks, detailed in Chapter 4, revealed three insights that reshape enterprise RAG deployment:

1. The Sufficiency Threshold

Graph-based retrieval offers superior reasoning capabilities but typically incurs prohibitive costs. The problem, however, lies in model selection rather than the architecture itself. Replacing the premium Claude Sonnet 4.5 model with the cost-optimised Amazon Nova Lite for entity extraction reduced indexing costs by a factor of 72, plummeting from $292.98 to just $4.05, as shown in Table 4.1. Yet retrieval quality held firm. Nova Lite achieved an Answer Relevance of 0.381, statistically identical to the premium model's 0.373 score in Table 4.8. This demonstrates that advanced graph reasoning is now economically viable for mid-tier applications.

2. The Maintenance Trap

Keeping an index fresh costs significantly more than building it in the first place. For dynamic corpora that require daily updates, re-indexing fees can exceed the initial investment within weeks. A GraphRAG system running on Claude Sonnet 4.5 costs over $21,000 annually to maintain, against just $47 for a baseline Vector RAG system as shown in Table 4.3. This 450-fold gap detailed in Table 4.4 creates a "Maintenance Trap" that makes technically superior architectures operationally unsustainable for rapidly changing knowledge bases.

3. Premium Embeddings Have Diminishing Returns

For synthesis-oriented technical documentation, budget-tier embedding models like Titan v2, priced at two cents per million tokens, match the quality of premium models like Cohere v3 that cost ten cents per million tokens, a finding supported by the metrics in Table 4.8 and statistical tests in Table 4.13. However, legal datasets still require premium models for precise entity extraction as demonstrated in Table 4.17. This divergence indicates that query type, rather than corpus size, should drive model selection.

This study reveals that while premium GraphRAG models incur exponential costs for marginal gains, the optimised Nova Lite configuration achieves the highest quality at a fraction of the price.

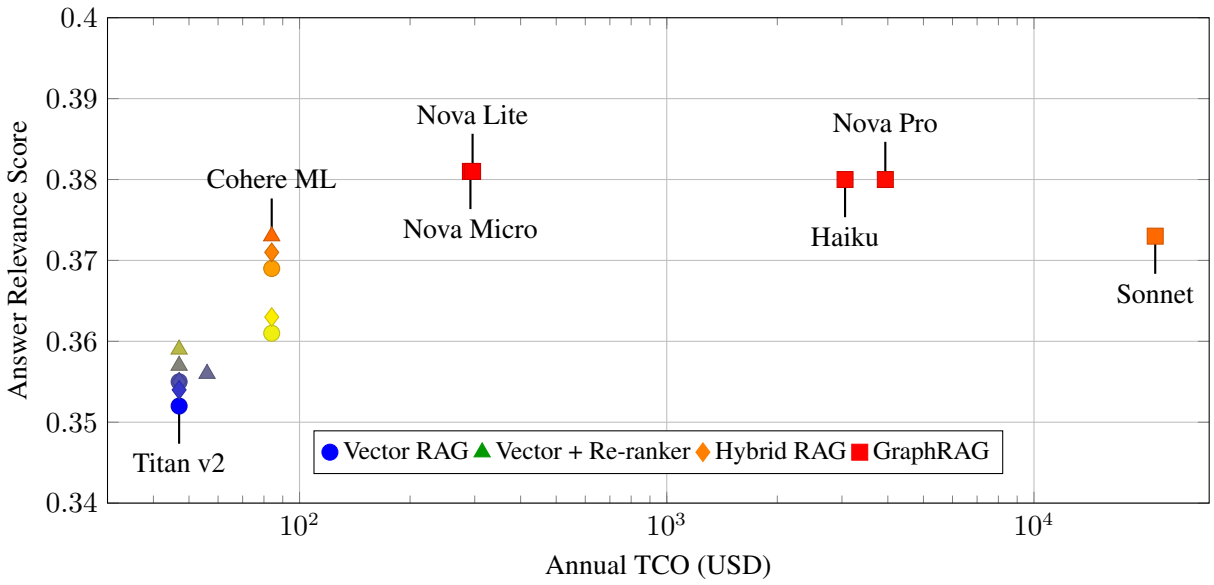


FIGURE 0.1. Cost-quality Pareto frontier showing annual TCO versus answer relevance.

Strategic Recommendations

Engineering leaders must adopt a new decision framework based on these findings, illustrated in Figure 4.7. For general enterprise search, Hybrid RAG, which combines vector and keyword search with budget embeddings, balances recall and cost effectively. It offers 95 percent of re-ranking performance at a fraction of the latency. GraphRAG suits multi-hop reasoning tasks, but only when using cost-optimised models like Nova Lite. Finally, teams must evaluate architectures based on Total Cost of Ownership over 12 months by explicitly factoring in update frequency. High-maintenance architectures should be avoided for rapidly changing knowledge bases unless the quality gains strictly justify the expense.

Effective enterprise RAG systems need not be expensive. By aligning architectural choices with operational constraints rather than chasing benchmarks, organisations build sustainable knowledge systems that deliver real business value.

Contents

Nomenclature

ACID: Atomicity, Consistency, Isolation, Durability.

AI: Artificial Intelligence.

AML: Anti-Money Laundering.

ANN: Approximate Nearest Neighbour.

ANOVA: Analysis of Variance.

APA: American Psychological Association.

API: Application Programming Interface.

APRA: Australian Prudential Regulation Authority.

ASME: American Society of Mechanical Engineers.

AWS: Amazon Web Services.

BERT: Bidirectional Encoder Representations from Transformers.

BM25: Best Match 25.

BPE: Byte Pair Encoding.

CAD: Computer-Aided Design.

CCPA: California Consumer Privacy Act.

CDATA: Character Data.

CER: Character Error Rate.

CFD: Computational Fluid Dynamics.

CNN: Convolutional Neural Network.

ColBERT: Contextualized Late Interaction over BERT.

CPU: Central Processing Unit.

CSV: Comma-Separated Values.

CXNPL: Constantinople.

DB: Database.

DeFi: Decentralised Finance.

DHT: Distributed Hash Table.

DLJ: Document Layout JSON.

DocVQA: Document Visual Question Answering.

DOM: Document Object Model.

DPO: Direct Preference Optimization.

DPR: Dense Passage Retrieval.

EC2: Elastic Compute Cloud.

ETL: Extract, Transform, Load.

FAISS: Facebook AI Similarity Search.

FEA: Finite Element Analysis.

FiD: Fusion-in-Decoder.

GCN: Graph Convolutional Network.

GDPR: General Data Protection Regulation.

GIF: Graphics Interchange Format.

GNN: Graph Neural Network.

GPT: Generative Pre-trained Transformer.

GPU: Graphics Processing Unit.

GraphRAG: Graph Retrieval-Augmented Generation.

HNSW: Hierarchical Navigable Small World.

HSD: Honestly Significant Difference.

HTML: HyperText Markup Language.

HTTP: Hypertext Transfer Protocol.

Hybrid RAG: Hybrid Retrieval-Augmented Generation.

IAM: Identity and Access Management.

IDC: International Data Corporation.

IEEE: Institute of Electrical and Electronics Engineers.

IGES: Initial Graphics Exchange Specification.

IR: Information Retrieval.

ISO: International Organization for Standardization.

IT: Information Technology.

JPEG: Joint Photographic Experts Group.

JSON: JavaScript Object Notation.

KV: Key-Value.

KYC: Know Your Customer.

LLM: Large Language Model.

LMM: Large Multimodal Model.

LoRA: Low-Rank Adaptation.

LSH: Locality-Sensitive Hashing.

LSTM: Long Short-Term Memory.

MCP: Model Context Protocol.

MENACE: Matchbox Educable Noughts and Crosses Engine.

MIME: Multipurpose Internet Mail Extensions.

ML: Machine Learning.

MMR: Maximal Marginal Relevance.

MRR: Mean Reciprocal Rank.

MS MARCO: Microsoft MAchine Reading COmprehension.

MTEB: Massive Text Embedding Benchmark.

NDCG: Normalised Discounted Cumulative Gain.

NIAH: Needle In A Haystack.

NLP: Natural Language Processing.

OCR: Optical Character Recognition.

OEM: OCR Engine Mode.

OSD: Orientation and Script Detection.

PDF: Portable Document Format.

PII: Personally Identifiable Information.

PNG: Portable Network Graphics.

PPO: Proximal Policy Optimization.

PSM: Page Segmentation Mode.

RAG: Retrieval-Augmented Generation.

RAGAs: Retrieval Augmented Generation Assessment.

RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval.

RDF: Resource Description Framework.

REALM: Retrieval-Augmented Language Model Pre-Training.

RLHF: Reinforcement Learning from Human Feedback.

RM: Reward Model.

RNN: Recurrent Neural Network.

RRF: Reciprocal Rank Fusion.

RSI: Repetitive Strain Injury.

S3: Simple Storage Service.

SFT: Supervised Fine-Tuning.

SLA: Service Level Agreement.

SLO: Service Level Objective.

SME: Subject Matter Expert.

SME: Small-Medium Enterprise.

SQL: Structured Query Language.

STEP: Standard for the Exchange of Product Model Data.

STL: Standard Tessellation Language.

SVM: Support Vector Machine.

TCO: Total Cost of Ownership.

TIFF: Tagged Image File Format.

TLS: Transport Layer Security.

TTFT: Time To First Token.

USD: United States Dollar.

UTF-8: Unicode Transformation Format - 8-bit.

VBA: Visual Basic for Applications.

Vector RAG: Vector Retrieval-Augmented Generation.

ViT: Vision Transformer.

VLM: Vision-Language Model.

VPN: Virtual Private Network.

WBS: Work Breakdown Structure.

WER: Word Error Rate.

WH&S: Work Health and Safety.

XAI: Explainable Artificial Intelligence.

XML: Extensible Markup Language.

Glossary

Ablation Study: An experimental procedure where components of a system are removed or deactivated to assess their individual contribution to overall performance.

Answer Relevance: A Retrieval Augmented Generation Assessment (RAGAs) metric that evaluates how pertinent the generated answer is to the original query, ensuring the system addresses the user's intent.

ChromaDB: An open-source embedding database used for storing and retrieving vector embeddings in Artificial Intelligence (AI) applications.

Chunking: The process of splitting long documents into smaller, manageable text segments, known as chunks, to fit within the context window of an Large Language Model (LLM) and improve retrieval precision.

Context Relevance: A RAGAs metric that measures the signal-to-noise ratio of the retrieved context, assessing whether the retrieved chunks contain the necessary information to answer the query.

Context Window: The maximum amount of text, measured in tokens, that an LLM can process in a single input sequence.

Cosine Similarity: A metric used to measure the semantic similarity between two vectors by calculating the cosine of the angle between them.

Cypher: A declarative graph query language used by Neo4j to query and manipulate data in a property graph.

Embedding: A dense vector representation of text in a high-dimensional space, where semantic similarity between texts corresponds to geometric proximity.

Faithfulness: A RAGAs metric that measures the factual consistency of the generated answer against the retrieved context, ensuring no hallucinations are introduced.

Fintech: Financial Technology. The application of technology to deliver financial services, including digital payments, lending platforms, blockchain, and regulatory compliance automation.

GraphRAG Architecture: A retrieval architecture that uses Knowledge Graphs, entity extraction, and community detection to enable multi-hop reasoning and global summarisation.

Ground Truth: The verified, correct answer or dataset used as a standard for evaluating the performance of a model.

Hallucination: The generation of incorrect or non-factual information by an LLM, often caused by a lack of relevant context or training data.

HNSW: Hierarchical Navigable Small World. A graph-based algorithm for Approximate Nearest Neighbour (ANN) search that enables fast high-dimensional vector retrieval.

Inference: The process of using a trained machine learning model to generate predictions or text output based on new input data.

Information Saturation: The theoretical point at which increasing embedding model dimensions yields diminishing returns in retrieval quality for a specific domain.

Knowledge Graph: A structured representation of knowledge representing entities as nodes, and relationships as edges, enabling complex reasoning and traversal.

LLMflation: The rapid deflationary trend in AI inference costs, where the price per token for frontier models decreases exponentially over time.

Louvain Algorithm: A hierarchical clustering algorithm for detecting communities in large networks by optimising modularity.

Maintenance Trap: The phenomenon where the cumulative cost of re-indexing a dynamic corpus over time exceeds the initial indexing cost by orders of magnitude.

Neo4j: A graph database management system that stores data nodes and relationships, enabling complex graph traversals and pattern matching.

RAG: A technique that enhances LLM output by retrieving relevant information from an external knowledge base before generation.

Reranking: A second-stage retrieval process where a more powerful model re-scores the initial set of retrieved documents to improve relevance.

Schema: The formal structure defining the types of nodes, relationships, and properties allowed in a Knowledge Graph.

Three-Phase Orchestrator: The custom software harness developed for this thesis to manage the Parsing, Indexing, and Evaluation phases of the experiment, ensuring isolation and reproducibility.

Token: The fundamental unit of text processing for LLMs, roughly equivalent to 0.75 words. Costs and limits are typically measured in tokens.

Vector Architecture: A retrieval architecture that uses dense vector embeddings and ANN search to find semantically similar documents.

Vector Database: A specialised database optimised for storing and querying high-dimensional vector embeddings.

Zero-shot Prompting: A prompting technique where the model is given a task without any specific examples of how to perform it.

List of Figures

List of Tables

CHAPTER 1

Introduction

Fintech firms operate under a constant tension between regulatory compliance, risk management, and client servicing, demanding rapid access to institutional expertise. Yet Subject Matter Experts (SMEs) remain a scarce and expensive bottleneck, and the cost for this rapid access scales with the size of the knowledge base (LexisNexis Risk Solutions 2024; Conference of State Bank Supervisors 2024; Deloitte 2024; Fourthline 2025). Retrieval-Augmented Generation (RAG) integrations with AI systems promise to democratise this expertise by grounding LLM responses in existing authoritative documentation, transforming months of analyst training into milliseconds of semantic retrieval. However, this promise obscures a critical challenge that existing research has largely ignored: *the operational economics of maintaining these systems over dynamic, continuously evolving corpora*. These are fundamentally *economic* decisions: the choice of embedding model or retrieval architecture directly determines re-indexing costs, which in turn determine whether a RAG system remains operationally viable as knowledge bases evolve. Currently, these are treated merely as technical selections. To quantify this oversight, we evaluate four RAG architectures against a 12,214-document enterprise corpus from Constantinople's Confluence workspace, measuring the "Maintenance Trap" phenomenon where re-indexing costs compound over time, and testing whether premium models, whose cost premiums command anywhere from 5 to 72$\times$ baseline, deliver proportional quality improvements.

This chapter introduces the research context, articulating why existing RAG literature, which is focused on static benchmark evaluations, provides insufficient guidance for production deployments where re-indexing costs can exceed initial implementation expenses by orders of magnitude (CIO 2024; Kenny 2024). We establish the unified research question driving this thesis: *given that model size determines operational costs, and operational costs determine system viability, what is the minimum model complexity required to achieve acceptable retrieval quality for enterprise knowledge systems?* This question spans architectural choices across Vector Retrieval-Augmented Generation (Vector RAG), Hybrid Retrieval-Augmented Generation (Hybrid RAG), and Graph Retrieval-Augmented Generation (GraphRAG) approaches alongside model selection between budget and premium embeddings and LLMs, treating them as interdependent variables in a cost-performance optimisation problem rather than independent technical decisions. The chapter frames these questions within Constantinople's fintech context, where the convergence of regulatory compliance demands, technical documentation complexity, and operational cost constraints creates representative challenges for the broader adoption of AI within existing enterprise systems.

1.1 Background and Context

The Fintech sector has reshaped global finance over the past decade, transforming how financial services are delivered and consumed (Boston Consulting Group 2024; Fortune Business Insights 2024). At the intersection of technology and finance, companies are increasingly required to manage vast ecosystems of interconnected systems, regulatory frameworks, and technical documentation. This convergence has created new challenges in knowledge management, particularly for organisations operating complex platform infrastructures that serve multiple stakeholders across varied regulatory jurisdictions.

Constantinople (CXNPL), an Australian Fintech company, exemplifies this challenge. As a provider of comprehensive "bank-in-a-box" platform services, the company manages an intricate ecosystem encompassing software infrastructure, regulatory compliance protocols, and extensive technical documentation (Constantinople 2024). Its intellectual property portfolio ranges from proprietary algorithmic implementations to nuanced compliance frameworks, and represents operational assets and critical strategic differentiators in the competitive Fintech landscape.

The emergence of artificial intelligence, particularly LLMs and RAG frameworks, presents a transformative opportunity for addressing knowledge management challenges in complex technical environments. These technologies promise to reshape how organisations access, synthesise, and deploy their institutional knowledge, moving from traditional document-based systems to intelligent, conversational interfaces that can understand context and provide precise, relevant information on demand.

1.2 Problem Statement

As James Somers observes in *The New Yorker*, "Every kind of thinking—whether Joycean, Proustian, or logical—depends on the relevant thing coming to mind at the right time. It's how we figure out what situation we're in" (Somers 2025). This fundamental truth about human cognition applies equally to organisational intelligence: an enterprise's effectiveness depends not merely on possessing knowledge, but on surfacing the *right* knowledge at the *right* moment.

Constantinople's rapid growth trajectory, while indicative of market success, has precipitated a critical operational challenge that threatens to constrain future scalability. As the company expands its client base and continuously evolves its platform capabilities, the volume and complexity of technical and regulatory information grows exponentially. This information explosion manifests across multiple dimensions. Technical documentation comprises thousands of pages of Application Programming Interface (API) specifications, implementation guides, architectural diagrams, and debugging procedures demanding constant maintenance. Regulatory compliance materials span multiple jurisdictions, from Australian Prudential Regulation Authority (APRA) guidelines to Anti-Money Laundering (AML) requirements and Know Your Customer (KYC) protocols. Client-specific configurations capture customised implementations, integration patterns, and operational procedures unique to each banking partner. Beneath it all lies historical context, years of accumulated decisions, rationales, and lessons learned that inform current practices yet exist primarily in unstructured formats.

The current state of knowledge management at Constantinople reflects a common enterprise pattern, where critical information remains fragmented across disparate systems, from Confluence (Atlassian 2024) wikis and SharePoint repositories to Slack conversations and email threads. Engineers searching for specific implementation details may need to navigate multiple platforms, often relying on tribal knowledge or interrupting colleagues for guidance. Support teams, responding to partner bank or customer queries, face similar challenges leading to increased response times and potential inconsistencies in information provided. A baseline RAG pipeline is illustrated in Figure 1.1.

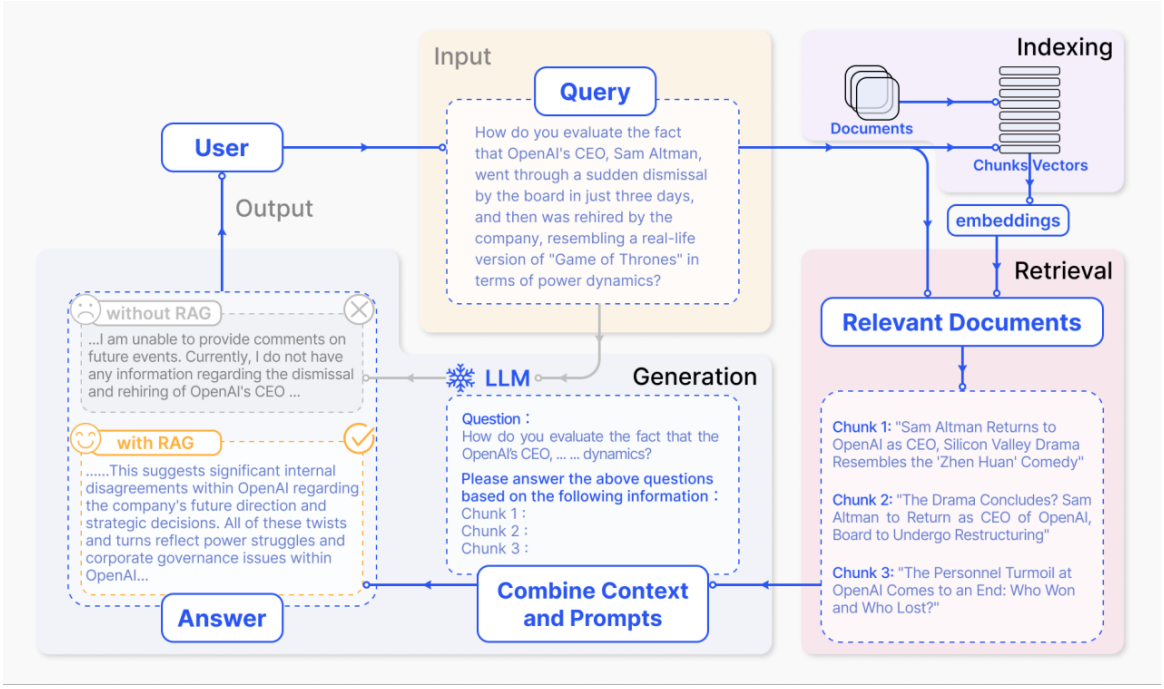


FIGURE 1.1. Baseline RAG pipeline illustrating how external context grounds responses (Gao et al. 2024).

This fragmentation creates a number of cascading effects throughout the organisation. For example, onboarding new engineers requires extensive mentorship periods as they navigate the labyrinth of documentation. Troubleshooting production issues becomes time-intensive when relevant information is buried in historical incident reports. Partner banks experience delays in receiving authoritative answers to technical queries, potentially impacting their own service delivery timelines.

1.3 The Dynamic Knowledge Problem

Traditional RAG research and deployment treat knowledge bases as *static artefacts* that are indexed once, and queried indefinitely. This assumption, while convenient for academic benchmarks and pilot projects, is fundamentally incompatible with the data reality within production fintech environments where knowledge is intrinsically *dynamic*. Confluence documentation updates daily, Jira tickets evolve hourly reflecting project progress, regulatory guidelines change weekly, and risk assessment reports refresh continuously. Each change invalidates portions of the existing index and triggers *re-indexing*, incurring computational and monetary costs as the corpus evolves.

Re-indexing costs represent a hidden operational expense that can dwarf initial implementation costs over the system's lifetime (CIO 2024), a phenomenon we term the "Maintenance Trap". Consider a bank deploying a GraphRAG system with 10,000 documents using a frontier model like Claude Sonnet 4.5. If the knowledge base receives daily updates, the *annual* re-indexing cost can exceed $100,000, as each update costs approximately $293. We quantify this phenomenon empirically in Chapter 4. Even with optimised models, these costs represent a significant recurring liability that must be managed.

Yet existing literature provides minimal guidance on this critical operational dimension. Benchmark studies focus on retrieval quality and query latency over static corpora, ignoring the economic viability of maintaining indices as knowledge evolves (P. Lewis et al. 2020; Gao et al. 2024). Research on GraphRAG demonstrates impressive reasoning capabilities but neglects to quantify the marginal cost of incremental updates (Edge et al. 2024); work on Vector RAG prioritises initial indexing speed without measuring update costs. This gap between static-corpus benchmarks and dynamic-corpus operations motivates the present research.

1.4 Research Questions and Objectives

This research investigates the operational viability of retrieval-augmented generation architectures when deployed against continuously evolving enterprise knowledge bases. Existing literature has largely solved high-quality retrieval on static benchmarks. The fundamental challenge for production systems is maintaining that performance under continuous update scenarios characteristic of enterprise environments. This distinction motivates our primary research question:

> ***Which retrieval-augmented generation architecture provides the optimal cost-performance-quality trade-off for enterprise knowledge systems requiring frequent index updates from continuous data streams, and what configuration heuristics enable practitioners to deploy these systems effectively?***

Research Objectives

The investigation proceeds through systematic empirical evaluation grounded in information retrieval theory and observations from production deployments. To answer the unified research question, this thesis pursues four specific objectives that collectively address the cost-quality relationship.

The first objective examines the relationship between model size and cost, specifically whether retrieval quality demonstrates independence from model size when architecture remains constant. If domain-specific technical corpora exhibit constrained vocabulary that can be captured by modest dimensional representations, then premium models commanding cost premiums would be economically unjustifiable, directly reducing operational costs without sacrificing quality.

The second objective investigates the interplay between the retrieval architecture and query complexity, determining whether GraphRAG demonstrates advantages specifically for multi-hop reasoning tasks requiring entity relationship traversal, while Vector RAG proves sufficient for straightforward factual retrieval. This distinction facilitates cost optimisation, allowing practitioners to select the most efficient architecture for each class of information need.

The third objective focuses on re-indexing cost quantification, measuring how re-indexing costs scale across architectures and model choices, determining whether GraphRAG costs scale disproportionately to corpus size due to the combinatorial complexity of mapping entity relationships, and identifying break-even points where graph-based retrieval justifies its higher maintenance costs. This objective directly measures the Maintenance Trap discussed in Section 1.3 that motivates the research.

The fourth objective develops an operational decision framework, synthesising findings into actionable guidance for practitioners. Beyond high-level architecture selection, this framework provides specific configuration heuristics, including optimal retrieval depth, k, derived from sensitivity analysis, embedding dimension guidance for different query types, chunk size-to-overlap ratios balancing context coherence against retrieval precision, and hybrid fusion weightings. These parameter recommendations translate empirical findings into deployable configurations, addressing the gap between academic benchmarks and production system tuning. The framework operationalises the thesis argument that model selection is fundamentally an economic optimisation problem.

1.5 Significance and Contributions

This research introduces an economic dimension to the study of retrieval-augmented generation, shifting the discourse from pure performance maximisation toward sustainable operational economics. By quantifying the practical trade-offs required to maintain high-fidelity knowledge systems in regulated environments, it provides a blueprint for viable enterprise AI adoption.

Practical Significance

For Constantinople, the successful implementation of an AI-powered knowledge management system promises substantial operational improvements. By enabling conversational access to institutional knowledge, the system can dramatically reduce the time required for engineers to find critical information, accelerate onboarding processes for new team members, and improve the consistency and accuracy of support provided to customers, and partner banks. These improvements translate directly to competitive advantages by breaking the linear relationship between headcount and knowledge capacity. Strategically, the ability to rapidly synthesise technical and regulatory context accelerates feature development cycles, reducing time-to-market for critical banking capabilities. Operationally, decoupling knowledge access from human availability reduces the mean time to resolution for production issues and lowers the overhead of scaling support teams. Ultimately, this creates a virtuous economic cycle: accelerated delivery drives faster revenue realisation, while reduced operational overhead improves margins, positioning the platform not just as a technology provider, but as a trusted regulatory partner.

Theoretical Contributions

Beyond its immediate practical applications, this research contributes to the growing body of knowledge surrounding RAG system implementation in enterprise environments. While much existing literature focuses on general-purpose applications or academic benchmarks, this project provides insights specific to highly technical, regulated domains where accuracy and reliability are paramount. The systematic evaluation of different retrieval strategies and architectural patterns adds empirical evidence to guide future implementations.

Industry Implications

The lessons learned from this project extend beyond Constantinople to the broader fintech and enterprise software sectors, addressing challenges that empirical evidence suggests are widespread. Industry surveys consistently identify knowledge management as a critical operational bottleneck. For instance, McKinsey estimates that employees spend 19% of their working time searching for and gathering information (Chui et al. 2012), while Gartner projects that 30% of generative AI projects will be abandoned after proof-of-concept by 2025 due to poor data quality, inadequate risk controls, or escalating costs (Gartner 2024). The International Data Corporation (IDC) forecasts enterprise data volumes growing at 23% annually through 2025 (International Data Corporation 2021), creating precisely the dynamic corpus challenge that forms the central investigative focus of this thesis.

Constantinople's operational environment exemplifies the challenges of modern knowledge management, mirroring patterns observed across enterprise software, healthcare informatics, legal technology, and financial services. By isolating a representative 10GB experimental slice of its technical documentation to model daily update requirements and multi-jurisdictional compliance constraints, this research targets the specific intersection of scale and volatility found in modern enterprise systems. The fragmentation of knowledge across Confluence, Jira, Slack, and email that Constantinople experiences reflects the "dark matter" of enterprise knowledge identified in knowledge management literature (Davenport and Prusak 1998). By systematically quantifying the cost-quality trade-offs in this representative environment, the research provides actionable guidance transferable to organisations facing analogous challenges.

1.6 Thesis Outline

This thesis is organised into six chapters that progressively build from theoretical foundations to empirical findings and practical recommendations.

Chapter 2: Literature Review surveys the theoretical foundations of retrieval-augmented generation, tracing the evolution from traditional information retrieval through dense vector search to contemporary graph-augmented approaches. This chapter frames the research gap by contrasting the field's maturity in static retrieval evaluation with its nascent understanding of the operational costs inherent in dynamic environments.

Chapter 3: Methodology describes the experimental design, detailing the evaluation of four RAG architectures: Vector RAG, Vector + Re-ranker, Hybrid RAG, and GraphRAG. It further characterises the 63,205-chunk corpus derived from Constantinople's technical documentation and establishes measurement protocols for cost, latency, and quality metrics. The chapter details the RAGAs evaluation framework and acknowledges its limitations for structured retrieval outputs.

Chapter 4: Experimental Results presents empirical findings covering indexing and re-indexing costs, query latency distributions, and retrieval quality metrics. Key results include the orders-of-magnitude cost disparity between GraphRAG configurations, the model size independence phenomenon within architectures, and the external validation on the Haystack benchmark that reveals task-dependent model selection requirements.

Chapter 5: Analysis and Discussion interprets the empirical findings through an operational economics framework, establishing a formal Total Cost of Ownership model and identifying the "Maintenance Trap" where re-indexing costs dominate long-term expenses. The chapter compares findings with prior literature, acknowledges limitations and threats to validity, and discusses ethical and environmental considerations.

Chapter 6: Conclusion synthesises the research contributions, answers the research questions posed in this introduction, and provides actionable recommendations for practitioners. The chapter concludes with directions for future research, including incremental graph update algorithms and evaluation metrics for structured retrieval.

CHAPTER 2

Literature Review

This chapter examines the theoretical and empirical foundations underlying RAG systems for enterprise applications. The review synthesises research across five key domains including language model architectures and their enterprise deployment considerations alongside embedding methods for semantic retrieval. It further explores graph-theoretic approaches to multi-hop reasoning and reranking strategies that combine complementary signals with hybrid retrieval. The final domain covers document understanding pipelines for processing heterogeneous corpora. A critical gap in existing literature emerges from this review as static benchmark evaluations dominate RAG research while the operational economics of maintaining indices over dynamic knowledge bases remain largely unexplored. This oversight is particularly important because re-indexing costs compound over time in such environments. This gap motivates the empirical evaluation presented in subsequent chapters which systematically measures retrieval quality and the marginal costs of continuous updates across different architectural choices.

2.1 History of Artificial Intelligence

Understanding the historical development of AI provides crucial context for current applications in banking and finance. This section traces the evolution of AI from its theoretical foundations to modern implementations, highlighting key milestones and paradigm shifts that have shaped the field.

Early Development and Theoretical Foundations

The conceptual foundations of AI can be traced back to the work of Alan Turing, who proposed the Turing Test in 1950 as a criterion for machine intelligence (Turing 1950). The formal establishment of AI as an academic discipline occurred at the Dartmouth Conference in 1956. John McCarthy, alongside Marvin Minsky, Nathaniel Rochester, and Claude Shannon, proposed the conference to explore how machines could simulate human thinking, leading to the establishment of AI as a formal area of study (McCarthy et al. 2006). It was here that McCarthy coined the term "artificial intelligence" and outlined ambitious goals for the field.

The early decades of AI research were characterised by symbolic approaches and expert systems. These rule-based systems attempted to encode human knowledge explicitly, achieving notable successes in narrow domains such as chess playing and medical diagnosis. An illustrative example of early reinforcement learning was Donald Michie's Matchbox Educable Noughts and Crosses Engine (MENACE), constructed in 1961 using 304 matchboxes to mechanically learn optimal strategies for Tic-Tac-Toe through trial and error (Michie 1961). However, the limitations of these approaches became apparent in the 1970s and 1980s as they proved brittle when facing ambiguous inputs and required a combinatorial explosion of rules for complex tasks. This failure to scale to real-world complexity led to the first "AI winter", a period of reduced funding and diminished expectations (Bubeck et al. 2023).

The resurgence of AI in the 1990s was driven by advances in machine learning, particularly the development of Support Vector Machines (SVMs). SVMs introduced a robust method for classification by finding the optimal hyperplane that maximises the margin between classes, offering superior generalisation capabilities compared to earlier neural networks (Cortes and Vapnik 1995). The increasing availability of computational resources further accelerated this progress. The breakthrough came with the deep learning revolution of the 2010s, initiated by innovations in neural network architectures such as Convolutional Neural Networks (CNNs) alongside the utilisation of Graphics Processing Units (GPUs) for efficient training. This era was marked by the success of AlexNet, which demonstrated that deep architectures could learn complex feature hierarchies from massive datasets (Krizhevsky, Sutskever and Hinton 2012; B. Min et al. 2023).

Evolution of AI in Financial Services

The adoption of AI in financial services has followed a distinct trajectory, beginning with simple automation tools in the 1980s and evolving to sophisticated machine learning systems today (Huang, Hui Wang and Y. Yang 2023). Early applications focused on automating routine tasks such as cheque processing and basic data entry. The 1990s saw the introduction of algorithmic trading systems, which could execute trades based on predefined rules and market conditions.

The 2000s marked a significant shift with the adoption of machine learning for credit scoring and risk assessment (Khandani, A. J. Kim and Lo 2010). Financial institutions began leveraging vast amounts of historical data to build predictive models that outperformed traditional statistical approaches. The 2008 financial crisis accelerated this trend, as regulators and institutions sought more sophisticated tools for risk management and stress testing (Butaru et al. 2016).

The current era of AI in finance is characterised by the deployment of deep learning models and large language models for complex tasks such as sentiment analysis, document processing, and automated advisory services (Shijie Wu et al. 2023). The integration of these technologies has enabled new business models and services, from robo-advisors to AI-powered trading strategies. This historical trajectory reveals a consistent pattern where each generation of AI technology has required financial institutions to balance capability gains against operational complexity and cost. This tension reaches its apex with retrieval-augmented generation systems, where the infrastructure overhead of maintaining knowledge indices can dominate total cost of ownership.

2.2 AI in Banking

Artificial intelligence has redefined the operational fabric of modern banking, shifting the industry from legacy, heuristic-based processes to automated, data-driven decision systems. This transition is propelled by the triple imperatives of enhanced efficiency, rigorous risk management, and superior customer experiences (Leo, Sharma and Maddulety 2019). This section examines the current state of AI adoption in banking, focusing on customer service automation, risk management, and regulatory compliance applications.

Current Applications

The contemporary banking sector has witnessed widespread deployment of AI across various operational domains. Machine learning algorithms have revolutionised credit risk assessment, enabling more accurate prediction of loan defaults and creditworthiness evaluation (Khandani, A. J. Kim and Lo 2010). These systems analyse vast amounts of structured and unstructured data, including transaction histories, social media activity, and alternative data sources, to generate comprehensive risk profiles (Butaru et al. 2016).

In the realm of customer service, AI-powered chatbots and virtual assistants have become ubiquitous, handling millions of customer queries daily (Y. Deng et al. 2023). These systems leverage natural language processing to understand customer intent and provide personalised responses, significantly reducing operational costs while improving response times. Industry analyses suggest that AI-driven customer service solutions can resolve a substantial proportion of routine queries without human intervention, with some implementations achieving resolution rates exceeding 70% (Kanaparthi 2024).

Fraud detection represents another critical application area where AI has demonstrated substantial value. Advanced machine learning models can identify fraudulent transactions in real-time by detecting anomalous patterns that would be impossible for human analysts to recognise (Kovacevic 2024). These systems continuously learn from new data, adapting to evolving fraud tactics and maintaining high accuracy rates while minimising false positives.

Challenges and Opportunities

Despite the promising applications, the implementation of AI in banking faces significant challenges. Regulatory compliance remains a primary concern, as financial institutions must ensure their AI systems adhere to stringent regulations regarding fairness, transparency, and accountability (Fuster et al. 2022). The “black box” nature of many machine learning models poses particular challenges for regulatory approval and audit requirements (S. Bi and Bao 2024).

Data privacy and security concerns represent additional obstacles to AI adoption. Banks must balance the need for comprehensive data analysis with strict data protection regulations such as General Data Protection Regulation (GDPR) and California Consumer Privacy Act (CCPA) (T. Li et al. 2022). Furthermore, the risk of adversarial attacks on AI systems has emerged as a critical security consideration, requiring robust defence mechanisms and continuous monitoring (Kovacevic 2024).

Looking forward, the opportunities for AI in banking remain substantial. The emergence of explainable Explainable Artificial Intelligence (XAI) techniques promises to address transparency concerns, making AI decisions more interpretable and auditable. Additionally, the integration of AI with blockchain technology and Decentralised Finance (DeFi) platforms presents new possibilities for secure, transparent, and efficient financial services. These considerations directly inform the present research. Financial institutions operate under strict regulatory constraints and auditability requirements. These factors create deployment environments where operational costs are as critical as accuracy metrics for determining system viability. This thesis addresses these gaps by systematically evaluating the cost-performance trade-offs of vector-based and graph-augmented RAG architectures. By quantifying the marginal utility of graph reasoning against its operational overhead, this research provides an evidence-based framework for selecting retrieval architectures that balance accuracy requirements with economic constraints, ensuring that AI systems remain viable for meeting strict Service Level Objectives (SLOs) for query latency and freshness in high-stakes banking environments.

2.3 Language Models & Large Language Models

The emergence of Large Language Models represents a paradigm shift in natural language processing capabilities, fundamentally altering how machines understand and generate human language. This section explores the architecture, training methodologies, and capabilities of modern language models, with particular emphasis on their applications in banking and finance.

Transformer Architecture and Attention Mechanisms

The transformer architecture, introduced by Vaswani et al. (2017), revolutionised natural language processing by replacing recurrent neural networks with self-attention mechanisms. This architecture enables parallel processing of sequence data, dramatically improving training efficiency and model performance. The key innovation of transformers lies in their ability to capture long-range dependencies in text through multi-head attention, allowing models to understand complex contextual relationships.

The self-attention mechanism computes relationships between all positions in a sequence simultaneously, producing contextualised representations that capture both local and global dependencies (Devlin et al. 2018). This approach overcomes the limitations of previous architectures, which struggled with long sequences and suffered from vanishing gradient problems. The transformer's encoder-decoder structure has proven remarkably versatile, adapting to various NLP tasks from translation to text generation.

Building on the transformer foundation, subsequent innovations have focused on scaling and optimisation. Models like Bidirectional Encoder Representations from Transformers (BERT) introduced bidirectional pre-training, enabling deeper contextual understanding (Devlin et al. 2018). Generative Pre-trained Transformer (GPT) models demonstrated the power of autoregressive language modelling at scale, showing that larger models trained on more data exhibit emergent capabilities not present in smaller versions (Radford et al. 2019).

Pre-training and Fine-tuning Strategies

The pre-training and fine-tuning paradigm has become the dominant approach for developing language models. During pre-training, models learn general language representations from vast corpora of unlabelled text, typically using objectives such as masked language modelling or next-token prediction (Raffel et al. 2020). This unsupervised learning phase enables models to capture linguistic patterns, factual knowledge, and reasoning capabilities without task-specific annotations.

Fine-tuning adapts pre-trained models to specific downstream tasks through supervised learning on labelled datasets (Chung et al. 2022). This transfer learning approach has proven remarkably effective, as the general knowledge acquired during pre-training provides a strong foundation for task-specific adaptation. Recent research has explored various fine-tuning strategies, including parameter-efficient methods like Low-Rank Adaptation (LoRA) and prompt tuning, which update only a small subset of model parameters (W. Wang et al. 2020).

The scale of pre-training has expanded dramatically, with models like GPT-3 containing 175 billion parameters and being trained on hundreds of billions of tokens (Brown et al. 2020). This scaling has revealed emergent abilities in large language models, including few-shot learning, chain-of-thought reasoning, and in-context learning capabilities (Wei, Tay et al. 2022). However, the computational costs and environmental impact of training such models have raised concerns about sustainability and accessibility.

Alignment and Reinforcement Learning

While pre-training and fine-tuning equip models with linguistic capabilities and domain knowledge, they do not inherently align model outputs with human intent or safety constraints. Reinforcement Learning from Human Feedback (RLHF) has emerged as the standard methodology for this alignment process (Christiano et al. 2017; Ouyang, Jeff Wu, X. Jiang et al. 2022). The typical RLHF pipeline involves three stages: Supervised Fine-Tuning (SFT) on high-quality demonstration data, training a Reward Model (RM) on human preference rankings, and finally optimizing the language model policy against this reward model using reinforcement learning algorithms such as Proximal Policy Optimization (PPO) (Schulman et al. 2017).

Recently, Direct Preference Optimization (DPO) has been proposed as a more stable and computationally efficient alternative, optimizing the policy directly from preference data without an explicit reward model (Rafailov et al. 2023). In the context of banking, alignment is critical for ensuring that models not only generate fluent text but also adhere to strict safety guidelines, avoid toxic outputs, and faithfully follow complex instructions regarding regulatory compliance.

LLMs in Banking Context

The application of large language models in banking has opened new possibilities for automating complex language-dependent tasks. Financial institutions are deploying LLMs for document analysis, regulatory compliance checking, and automated report generation (Huang, Hui Wang and Y. Yang 2023). These models can process vast amounts of unstructured financial text, extracting insights and identifying patterns that would be impractical for human analysts to discover.

Specialised financial language models have emerged to address the unique requirements of the banking sector. BloombergGPT, trained on a massive corpus of financial documents, demonstrates superior performance on finance-specific tasks while maintaining general language capabilities (Shijie Wu et al. 2023). Similarly, FinBERT and FinGPT have been developed to handle the specialised vocabulary and concepts prevalent in financial texts (H. Yang, X.-Y. Liu and C. D. Wang 2023).

The integration of LLMs with existing banking systems presents both opportunities and challenges. While these models can enhance decision-making and automate routine tasks, concerns about hallucination, bias, and regulatory compliance must be carefully addressed (Xie et al. 2023). Financial institutions are developing robust validation frameworks and human-in-the-loop systems to ensure the reliability and accuracy of LLM-generated outputs (Shah et al. 2022).

Hallucinations and Reliability

Despite the advances in alignment, large language models remain prone to "hallucinations", which is the generation of text that is fluent and plausible but factually incorrect or nonsensical (Ji et al. 2023). This phenomenon stems from the probabilistic nature of these models, which are trained to maximise the likelihood of the next token rather than to verify factual accuracy. Hallucinations can be categorised into intrinsic hallucinations that contradict the source input, and extrinsic hallucinations that generate unverifiable information not present in the source.

In the highly regulated banking sector, the tolerance for hallucination is effectively zero. A model that invents non-existent regulations or misinterprets financial data poses severe reputational and legal risks. While alignment techniques like RLHF can reduce the frequency of harmful outputs, they do not fundamentally solve the grounding problem. To ensure factual reliability, models must be anchored to verifiable external information sources. This necessity motivates the integration of retrieval systems and structured knowledge bases, which provide the ground truth against which model generations can be verified.

2.4 Knowledge Bases

Effective knowledge management is crucial for LLM applications in banking, where accurate, up-to-date information is essential for decision-making and regulatory compliance. This section examines various approaches to structuring and accessing domain-specific knowledge, from traditional databases to modern vector stores and hybrid systems.

Traditional Database Structures

Relational databases have long served as the backbone of banking information systems, providing structured storage for transactional data, customer records, and regulatory information. These systems excel at maintaining data integrity through Atomicity, Consistency, Isolation, Durability (ACID) properties and supporting complex queries through Structured Query Language (SQL) (Sun et al. 2023). However, their rigid schema requirements and limited support for unstructured data pose challenges for modern AI applications.

NoSQL databases emerged to address the limitations of relational systems, offering flexible schema designs and horizontal scalability. Document stores like MongoDB enable storage of semi-structured financial documents, while graph databases facilitate analysis of complex relationships in financial networks (Yue Li et al. 2023). These systems have proven valuable for specific use cases but often require multiple database types to handle diverse data requirements comprehensively.

The integration of traditional databases with AI systems requires careful consideration of data access patterns and performance requirements. Extract, Transform, Load (ETL) pipelines must be designed to prepare data for machine learning models while maintaining data quality and consistency. Real-time data synchronisation and caching strategies become critical when supporting latency-sensitive AI applications in banking environments.

Graph Databases and Knowledge Graphs

Graph databases, such as property graphs and Resource Description Framework (RDF) stores, model data as vertices and edges, making them well-suited for domains where relationships are first-class citizens. Examples include customer–product linkages, counterparty exposure, and transaction networks. Early surveys formalised graph database models and their query languages, highlighting differences in expressivity and performance across approaches (Angles and Gutiérrez 2008). Knowledge graphs build on these foundations to provide semantically rich, typed relations over entities, supporting reasoning, entity resolution, and provenance through standards and shared vocabularies (Hogan et al. 2021).

For banking, knowledge graphs enable multi-hop analyses, such as beneficial ownership discovery and AML/KYC checks, and context-aware retrieval by connecting heterogeneous sources under a unified schema. When paired with text corpora, they serve as structured scaffolds that complement unstructured evidence, improving precision and explainability in downstream tasks.

Graph Neural Networks and Graph Reasoning

Graph Neural Networks (GNNs) generalise deep learning to non-Euclidean domains by iteratively propagating and aggregating information over edges. Pioneering architectures such as Graph Convolutional Networks (GCNs) and GraphSAGE demonstrated effective representation learning on graphs, enabling node classification, link prediction, and subgraph-level reasoning (Kipf and Welling 2017; Hamilton, Ying and Leskovec 2017). Comprehensive surveys document rapid progress across message-passing variants, scalability techniques, and applications (Z. Wu et al. 2021).

2.5 Graph Theory Foundations

Graph theory provides the mathematical language for modelling relationships between entities, representing objects as vertices, also known as nodes, and relations as edges or links. This framework allows for the abstraction of complex systems into structured networks where the properties of connections are as significant as the entities themselves.

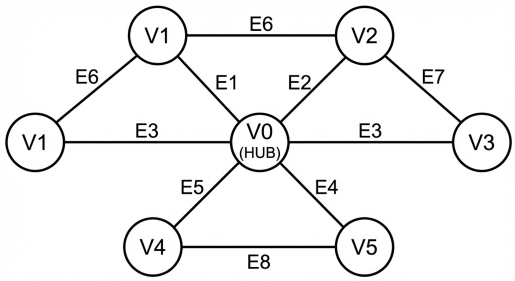


FIGURE 2.1. General graph structure illustrating nodes, edges, and a central hub demonstrating high connectivity.

Core concepts in graph theory characterise the structure and behaviour of these networks, directly informing algorithmic design for information retrieval. *Degree distribution* describes the probability distribution of connections across the network; in many real-world financial networks, this follows a power law where a few "hub" nodes, such as major banks or central counterparties, possess a disproportionately high number of connections compared to the periphery. *Connectivity* refers to the resilience of the network and the existence of paths between node pairs, determining how information or risk flows through the system.

Navigating these structures involves understanding *paths* and *cycles*. A path represents a sequence of distinct edges connecting two nodes, analogous to a chain of transactions or a provenance trail in a knowledge graph. Cycles, where a path returns to its starting node, can indicate self-dealing loops in financial crime analysis or recursive dependencies in software systems.

Centrality measures quantify the importance of specific nodes within the graph. Betweenness Centrality identifies nodes that act as bridges on the shortest paths between other nodes, representing critical control points or bottlenecks for information flow. Closeness Centrality measures how quickly a node can access all other nodes in the network, indicating entities that are well-positioned to disseminate information. Finally, Eigenvector Centrality, along with its variant PageRank, assigns importance based on connections to other high-scoring nodes, a principle used to rank the authority of documents or entities based on the quality of their references (Diestel 2017; Bondy and Murty 2008).

In information access systems, these properties determine how efficiently relevant material can be discovered and traversed, especially when reasoning requires multi-hop navigation across related items. Classical traversal algorithms, including breadth-first and depth-first search, shortest-path formulations, and flow-based techniques underpin practical retrieval over interconnected corpora, from citation networks to transaction graphs (Bondy and Murty 2008). Notably, small-world navigability is exploited by graph-based indexing structures such as Hierarchical Navigable Small World (HNSW) graphs, which enable sub-linear ANN search in high-dimensional spaces (Malkov and Yashunin 2018). These graph-theoretic primitives reappear throughout modern retrieval stacks, including vector search and knowledge graph reasoning.

2.6 Vector Databases and Embeddings

Vector databases represent a paradigm shift in information storage and retrieval, optimised for similarity search operations essential to modern AI applications (Johnson, Douze and Jégou 2019). These systems store text as high-dimensional vector representations, known as embeddings, enabling semantic search capabilities that transcend keyword matching. The ability to find conceptually similar information, regardless of exact wording, makes vector databases particularly valuable for enterprise applications where terminology varies across documents and regulations.

Foundations of Text Embeddings

The modern era of dense vector representations traces its origins to Mikolov et al. (2013), whose Word2Vec model demonstrated that neural networks could learn meaningful word embeddings from large text corpora. The key insight that semantic relationships could be captured through vector arithmetic, such as "king" minus "man" plus "woman" approximating "queen", established the foundation for all subsequent embedding research. Shortly thereafter, Pennington, Socher and Manning (2014) introduced GloVe, or Global Vectors, which combined the benefits of global matrix factorisation with local context window methods, achieving strong performance on word analogy and similarity tasks.

These word-level embeddings evolved into sentence and document representations with models like Sentence-BERT (Reimers and Gurevych 2019), which fine-tunes BERT using siamese and triplet networks to produce semantically meaningful sentence embeddings. Unlike word embeddings that require aggregation strategies for longer text, Sentence-BERT generates fixed-dimensional vectors that directly capture sentence-level semantics, making it practical for semantic search applications. The progression from static word embeddings such as Word2Vec and GloVe to contextualised embeddings like BERT and Sentence-BERT represents a fundamental shift where modern embeddings capture polysemy and context-dependent meaning rather than assigning a single vector to each word regardless of usage.

Embedding Model Evaluation and Selection

Selecting embedding models for production RAG systems requires systematic evaluation. The Massive Text Embedding Benchmark (MTEB) consolidates evaluation across eight task categories including classification, clustering, and retrieval spanning 58 datasets and 112 languages (Muennighoff et al. 2022). A critical finding from MTEB is that no single embedding model dominates across all tasks, suggesting that model selection should be task-specific rather than defaulting to the highest-ranked model. This finding directly informs the experimental design of this thesis, which evaluates multiple embedding models to assess whether performance differences observed in benchmarks translate to operational RAG contexts.

For retrieval-focused applications, the choice between embedding dimensionality, model architecture, and cost presents non-trivial trade-offs. Higher-dimensional embeddings, such as those with 3072 dimensions, provide greater representational capacity but increase storage costs and retrieval latency. Recent advances in embedding

models have focused on domain adaptation, with financial-specific embeddings showing improved performance on banking-related retrieval tasks (Yu et al. 2023). The experimental results presented in Chapter 4 directly test whether premium embeddings provide measurable quality improvements that justify their higher costs.

Approximate Nearest Neighbour Search

Efficient similarity search in high-dimensional spaces presents significant computational challenges. Exact nearest neighbour search scales as $\mathcal{O}(nd)$ for n vectors of dimension d, becoming prohibitive for large corpora. ANN algorithms trade modest accuracy reductions for dramatic speedups, enabling sub-linear search times while maintaining acceptable recall (Malkov and Yashunin 2018).

HNSW graphs have emerged as the dominant indexing structure for production vector databases. HNSW constructs a multi-layer graph where higher layers contain longer-range connections for coarse navigation and lower layers enable fine-grained local search, achieving $\mathcal{O}(\log n)$ query complexity (Malkov and Yashunin 2018). Alternative approaches include Locality-Sensitive Hashing (LSH), which uses hash functions to bucket similar vectors, and Product Quantization, which compresses vectors to reduce memory footprint at the cost of some precision (Jégou, Douze and Schmid 2011). Systems like Facebook AI Similarity Search (FAISS) implement these algorithms with optimisations for different hardware configurations, including GPU acceleration that can achieve 5–10$\times$ speedups over CPU implementations (Douze et al. 2024).

2.7 Retrieval-Augmented Generation

RAG couples parametric knowledge in LLMs with non-parametric retrieval to ground responses in external evidence, improving factuality and currency (P. Lewis et al. 2020; Borgeaud et al. 2022; Y. Wang et al. 2023; Gao et al. 2024). In practice, production systems often employ hybrid retrieval, combining sparse and dense methods with re-ranking to manage large corpora under tight latency constraints. Figure 2.2 summarises common evolutions from naive to modular RAG.

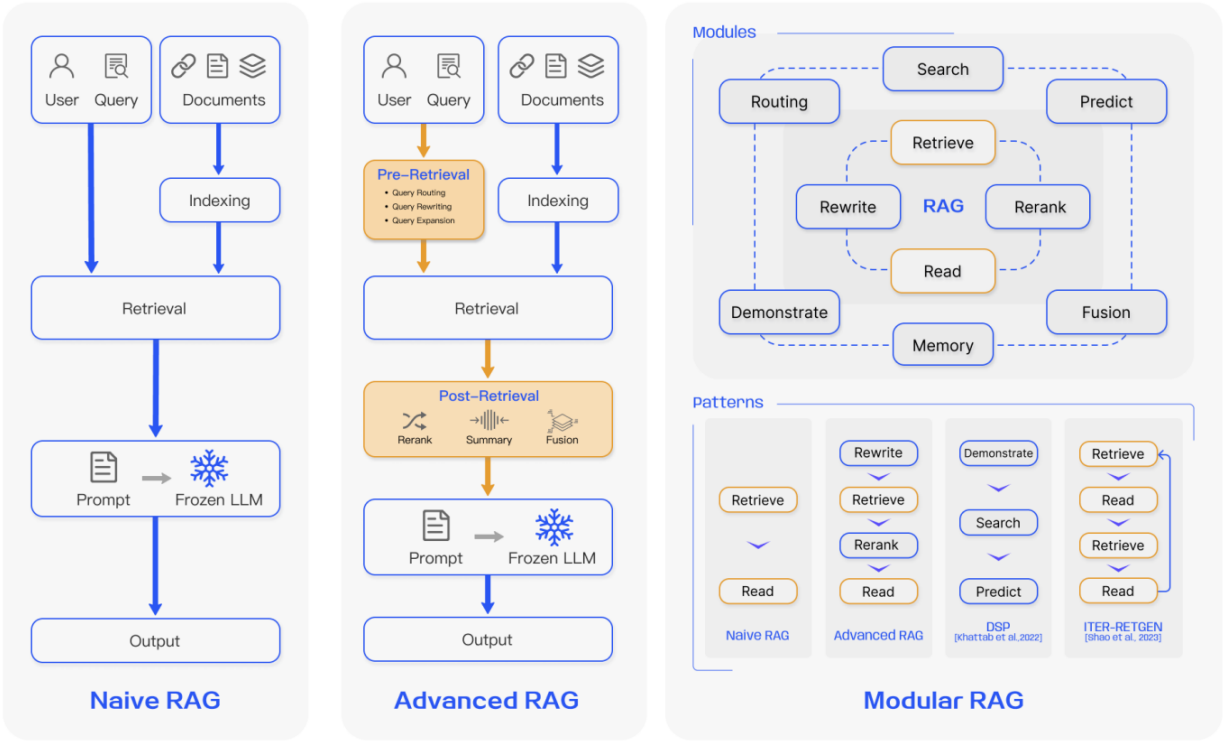


FIGURE 2.2. Evolution of RAG architectures from naive to modular patterns (Gao et al. 2024).

VectorRAG

Vector RAG represents the baseline architecture upon which more sophisticated approaches build. The concept of retrieval-augmented language models was pioneered by Guu et al. (2020), whose Retrieval-Augmented Language Model Pre-Training (REALM) demonstrated that jointly pre-training a retriever and language model could significantly improve performance on knowledge-intensive tasks. Building on this foundation, the seminal work by P. Lewis et al. (2020) introduced the retrieve-then-generate paradigm, demonstrating that coupling a pre-trained sequence-to-sequence model with a dense retrieval component significantly outperformed purely parametric approaches on knowledge-intensive tasks. This foundational architecture established three core stages comprising document chunking and embedding, vector indexing, and retrieval-augmented generation.

The retrieval component builds upon Dense Passage Retrieval (DPR), which Karpukhin et al. (2020) showed could outperform traditional sparse methods like Best Match 25 (BM25) by 9–19% on open-domain question answering benchmarks. DPR employs a dual-encoder or bi-encoder architecture that utilizes one BERT-based encoder for

queries and another for passages, with similarity computed via dot product between the resulting `[CLS]` token embeddings. This design enables efficient indexing where passage embeddings are computed once offline while capturing semantic similarity beyond keyword matching. Documents are split into semantically coherent chunks, typically 256–1024 tokens, then encoded into dense vector representations. While the original DPR used BERT-based encoders, modern implementations employ models such as OpenAI's text-embedding-ada-002 or open-source alternatives like Sentence-BERT (Reimers and Gurevych 2019), which optimises for semantic similarity through contrastive learning on sentence pairs. These embeddings are stored in vector databases optimised for ANN search, typically using HNSW indices that achieve sub-linear query complexity (Malkov and Yashunin 2018).

At query time, the user's question is embedded using the same model, and the k most similar chunks are retrieved via cosine similarity or dot product search. Izacard and Grave (2021) demonstrated that retrieval quality directly impacts generation performance, with their Fusion-in-Decoder (FiD) approach showing that aggregating evidence from multiple retrieved passages rather than concatenating them improves answer accuracy by enabling cross-attention over independent passage encodings. The retrieved chunks are concatenated into a context window and passed to a generative LLM along with the original query, producing a grounded response. The original RAG paper reported 44.5% exact match on Natural Questions, outperforming the previous state-of-the-art by 4 percentage points (P. Lewis et al. 2020).

From an operational perspective, Vector RAG offers predictable costs and minimal infrastructure requirements. Indexing requires only embedding computation, a single forward pass per chunk, and re-indexing scales linearly with the number of changed documents. Borgeaud et al. (2022) demonstrated that retrieval-augmented approaches can match the performance of models $25\times$ larger while requiring significantly less compute, establishing the cost-effectiveness argument that motivates enterprise adoption. This efficiency gain arises because retrieval externalises factual knowledge from model parameters to an updateable index, reducing the need for expensive model retraining when knowledge changes.

However, this flat retrieval approach exhibits fundamental limitations. Single-stage dense retrieval struggles with multi-hop reasoning tasks that require synthesising information across multiple documents or following chains of relationships (Karpukhin et al. 2020). The architecture assumes that relevant information exists within a single contiguous chunk, failing when answers require connecting disparate facts or understanding entity relationships. Additionally, the semantic similarity assumption underlying dense retrieval can retrieve topically related but factually irrelevant passages, a failure mode that motivated two-stage approaches incorporating reranking (Nogueira and Cho 2019). Recent work by Reichman and Heck (2024) critically examines the behaviour of DPR, demonstrating that dense retrievers may not always retrieve passages that directly answer the query, raising important questions about the distinction between semantic similarity and factual relevance in retrieval systems. These limitations motivated the development of more sophisticated architectures discussed in subsequent sections, including hierarchical retrieval such as RAPTOR, adaptive retrieval such as Self-RAG, and graph-augmented approaches such as GraphRAG.

Graph-Augmented Retrieval

Graph-augmented retrieval extends RAG with explicit relational structure. Knowledge graphs provide typed edges that enable multi-hop reasoning paths between query entities and candidate answers. Methods such as QA-GNN integrate graph reasoning with language models to select and articulate relevant relational evidence (Yasunaga et al. 2021). Complementarily, graph-based indices such as HNSW already power efficient dense retrieval, illustrating how graph-theoretic constructs underpin both the retrieval substrate and the reasoning layer (Malkov and Yashunin 2018). The overall process flow is depicted in Figure 2.3.

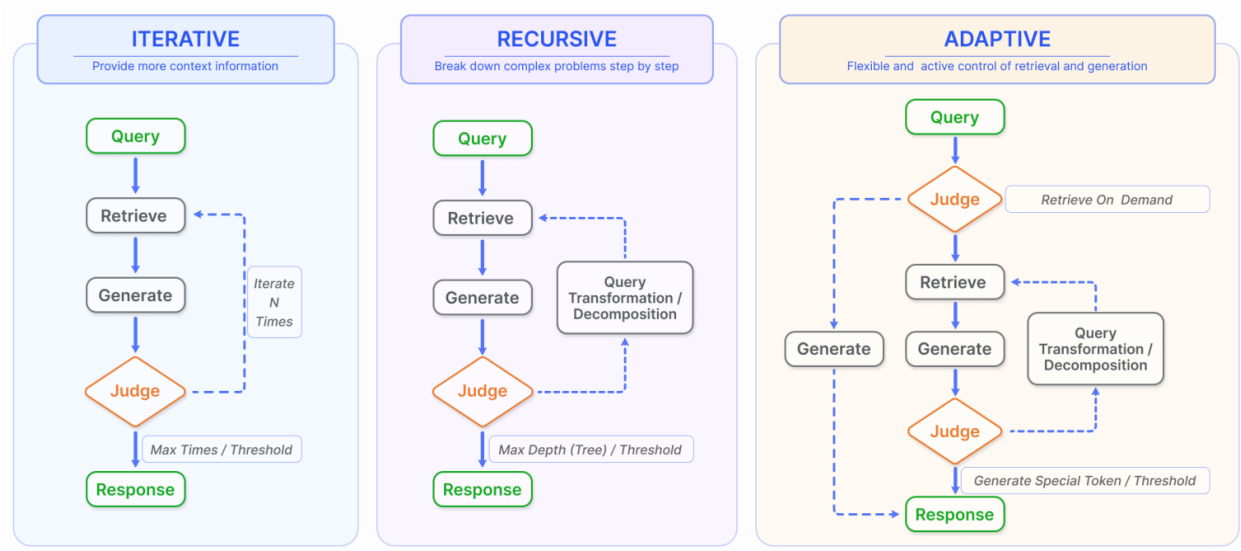


FIGURE 2.3. Standard RAG process flow encompassing indexing, retrieval, and generation phases (Gao et al. 2024).

For banking, graph-augmented RAG supports compliance and risk workflows by tracing provenance across interconnected regulations, counterparties, and transactions. Compared with purely vector-based retrieval, graph-aware systems offer improved controllability and explainability via explicit paths and constraints, while remaining compatible with the hybrid pipelines commonly deployed in enterprise environments.

These graph-augmented approaches represent a significant advance in retrieval capability, yet the literature exhibits a notable blind spot regarding operational costs. Microsoft's GraphRAG architecture (Edge et al. 2024), which employs Louvain community detection and LLM-based entity extraction, demonstrates impressive multi-hop reasoning on complex queries but does not report indexing costs or update strategies. The entity extraction phase requires an LLM call for each document chunk, resulting in a cost structure that scales linearly with corpus size and necessitates repetition when documents change. Similarly, QA-GNN's integration of graph reasoning with language models focuses on accuracy improvements but fails to acknowledge the graph construction and maintenance overhead. This omission is consequential, as the compelling accuracy gains reported in the literature for graph-augmented retrieval may obscure operational costs that render these systems economically unviable for frequently-updated knowledge bases. Quantifying these trade-offs forms a central contribution of the present research.

Prompt engineering plays a critical but underexplored role in GraphRAG performance and cost. The entity extraction prompt determines which entities and relationships are captured from source documents, directly affecting graph quality and downstream retrieval accuracy. Similarly, community summarisation prompts control how hierarchical graph communities are condensed into retrievable text, balancing comprehensiveness against token costs. Microsoft's reference implementation provides default prompts optimised for general-purpose corpora, but enterprise deployments may require domain-specific prompt engineering to capture technical terminology, regulatory entities, or organisational hierarchies absent from generic extraction patterns. These architectural choices carry significant cost implications. A poorly engineered extraction prompt may require re-indexing the entire corpus when revised, while an overly verbose summarisation prompt increases both indexing costs by generating more tokens per community and query costs by retrieving excessive context. Despite these considerations, the GraphRAG literature provides minimal guidance on prompt optimisation strategies. Existing studies typically treat prompts as fixed hyperparameters rather than tuneable components with distinct cost-quality trade-offs.

Networking Perspectives for Graph-Theoretic RAG

Networking concepts sharpen both the architecture and performance of graph-theoretic retrieval. Small-world structure explains the short routing paths and high clustering often leveraged in overlay indices and peer-to-peer search (Watts and Strogatz 1998). Decentralised navigability provides theoretical guarantees for greedy routing given suitable local coordinates or long-range link distributions (Kleinberg 2000). These ideas motivate practical designs for distributed retrieval backends.

Distributed Hash Tables (DHTs) such as Chord and Kademlia offer scalable, fault-tolerant key-based lookup with logarithmic hop complexity (Stoica et al. 2003; Maymounkov and Mazieres 2002). In modern dense retrieval, product quantisation and graph-based ANN structures reduce network and storage costs while maintaining accuracy (Jégou, Douze and Schmid 2011; Malkov and Yashunin 2018). Together, these networking and indexing paradigms inform RAG system design under strict latency SLOs, enabling balanced trade-offs across recall, throughput, and tail latency in enterprise deployments.

2.8 Advanced RAG Architectures

The landscape of retrieval-augmented generation has evolved rapidly, with recent innovations addressing fundamental limitations of early RAG systems. This section examines cutting-edge architectures and techniques that have emerged in the past two years, with particular emphasis on their operational implications for enterprise deployments.

Self-RAG and Adaptive Retrieval

Self-RAG represents a paradigm shift in retrieval-augmented generation by introducing self-reflective tokens that enable models to adaptively decide when to retrieve, what to retrieve, and how to use retrieved information (Asai et al. 2024). Unlike traditional RAG systems that retrieve for every query, Self-RAG learns to generate special tokens indicating retrieval necessity, relevance assessment, and response quality. This selective retrieval approach reduces unnecessary computational overhead by up to 40% while maintaining or improving response quality, directly addressing the operational cost concerns central to this thesis.

The architecture introduces four types of reflection tokens, where retrieval tokens determine whether external information is needed, relevance tokens assess retrieved passage quality, support tokens evaluate factual grounding, and utility tokens judge overall response helpfulness. Through reinforcement learning from human feedback, Self-RAG learns optimal retrieval strategies that balance information needs against computational costs, achieving state-of-the-art performance on knowledge-intensive tasks while reducing retrieval frequency by 30-50% compared to always-retrieve baselines.

However, several methodological limitations constrain the generalisability of these findings to enterprise contexts. The evaluation focuses on open-domain question answering benchmarks such as Natural Questions and TriviaQA where queries are well-formed and unambiguous, which are conditions rarely met in enterprise knowledge management where queries may be vague, incomplete, or require domain expertise to interpret. The reinforcement learning training also requires substantial human annotation to define "when retrieval helps," an expensive prerequisite that may not transfer across domains. Most critically for the present research, the reported cost savings address *query-time* retrieval but not the *indexing costs* that dominate operational expenses for dynamic corpora. A system that reduces retrieval calls by 40% but still requires full re-indexing when documents change provides limited benefit in high-update environments.

Recursive Abstract Processing for Tree-Organised Retrieval

Recursive Abstractive Processing for Tree-Organized Retrieval (RAPTOR) addresses the limitation of flat document retrieval by constructing hierarchical document representations through recursive summarisation (Sarthi et al. 2024). The system clusters related text chunks and generates increasingly abstract summaries at multiple levels, creating a tree structure that enables retrieval at varying levels of granularity. This hierarchical approach proves particularly effective for multi-hop reasoning tasks where information synthesis across documents is required.

The recursive clustering and summarisation process creates semantic layers that capture both local details and global themes. When querying, RAPTOR traverses this tree structure to retrieve information at the appropriate abstraction

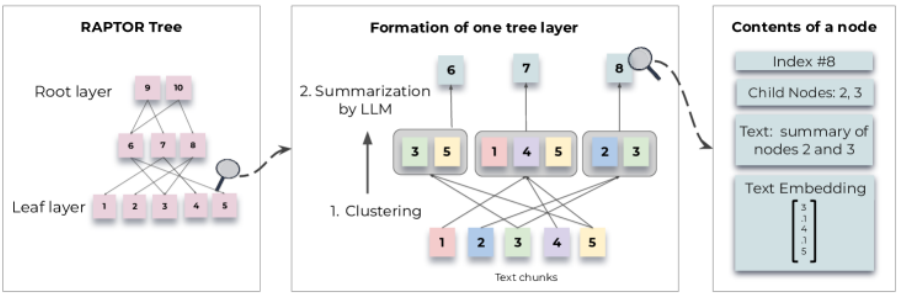


FIGURE 2.4. Tree construction process: RAPTOR recursively clusters chunks of text based on their vector embeddings and generates text summaries of those clusters, constructing a tree from the bottom up. Nodes clustered together are siblings; a parent node contains the text summary of that cluster (Sarthi et al. 2024).

level, improving retrieval precision by 25% on complex reasoning benchmarks. For enterprise applications with large document corpora, this hierarchical organisation reduces search space complexity from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$ for many queries, though it incurs additional indexing costs that must be amortised over query volume.

This architectural choice introduces a significant but unquantified trade-off, as the recursive summarisation requires LLM calls proportional to document count during indexing, creating cost structures potentially similar to GraphRAG's entity extraction overhead. While the authors report 25% precision improvement on multi-hop reasoning, they do not disclose indexing costs or update strategies. For dynamic corpora, a single document modification may invalidate multiple tree layers, triggering cascading re-summarisation. The absence of incremental update mechanisms in the published architecture suggests RAPTOR may exhibit the same "Maintenance Trap" this thesis identifies for GraphRAG, though empirical verification remains an open question for future research.

Memory-Augmented Retrieval

MemoRAG introduces a dual-encoder architecture that maintains both short-term and long-term memory components for more effective context management (Q. Zhang et al. 2024). The short-term memory handles recent conversational context while the long-term memory stores persistent knowledge representations. This separation enables efficient updates to conversational state without re-indexing the entire knowledge base, addressing a critical operational challenge in dynamic environments.

Inspired by human cognitive processes, MemoRAG mimics how humans tackle complex tasks by first forming a global memory of the document, then recalling specific clues, and finally checking details to formulate a response. This contrasts with standard RAG pipelines which treat retrieval and generation as purely sequential steps. By constructing a global memory of long contexts, MemoRAG generates clues that guide the retrieval of relevant evidence, enabling it to handle million-token contexts and vague queries unsuitable for direct search, capabilities where standard LLMs often struggle.

The memory-augmented approach achieves 35% reduction in re-indexing costs for conversational applications by isolating volatile conversational state from stable knowledge representations. MemoRAG's attention mechanisms learn to route queries to appropriate memory components, reducing unnecessary database accesses by 45% while

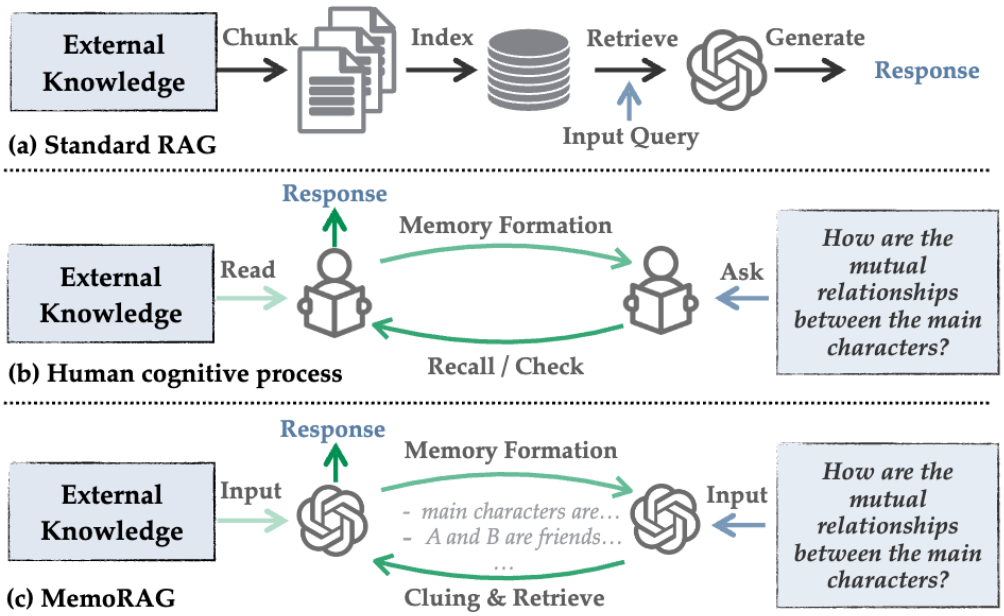


FIGURE 2.5. Comparison of MemoRAG with Standard RAG and human cognition of a long document. Sequential processing in Standard RAG (a) contrasts with human cognition (b) and MemoRAG's global memory approach (c), which enables non-sequential information retrieval (Q. Zhang et al. 2024).

maintaining response quality. This architecture proves particularly valuable for banking applications where customer interactions span multiple sessions and require both historical context and current information.

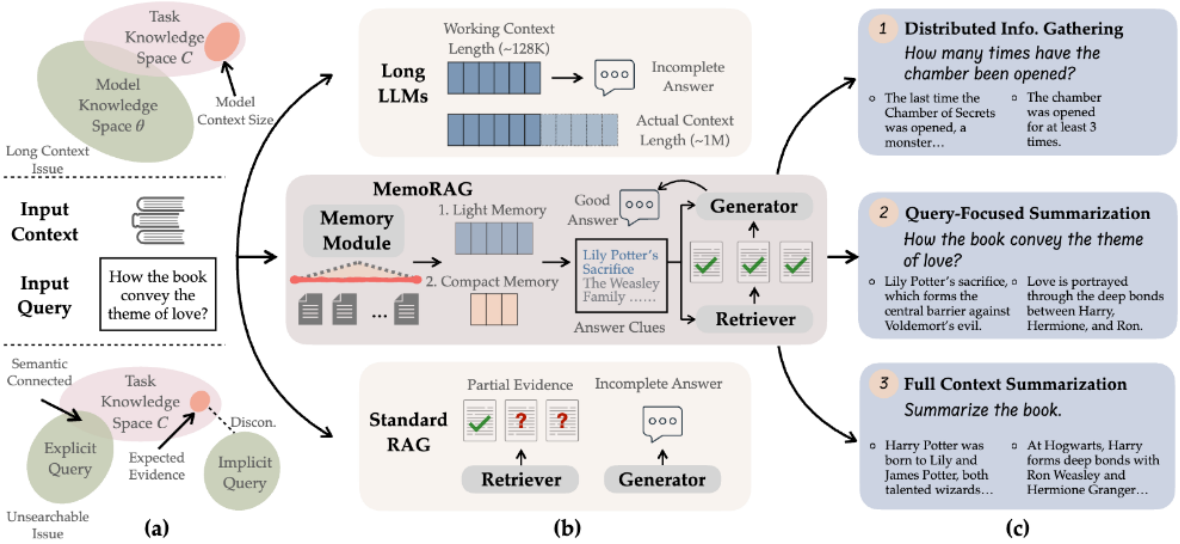


FIGURE 2.6. MemoRAG application scenarios illustrating (a) context length challenges, (b) query suitability issues in standard RAG, and (c) MemoRAG's solution using global memory to guide retrieval (Q. Zhang et al. 2024).

RAG versus Fine-tuning

While RAG has gained prominence for knowledge-intensive tasks, alternative approaches warrant consideration for specific use cases. Fine-tuning remains competitive for static, domain-specific knowledge where the corpus changes infrequently (E. J. Hu et al. 2021). Recent studies show that fine-tuned models can outperform RAG on closed-domain tasks with stable knowledge bases, while requiring no retrieval infrastructure.

Parameter-efficient fine-tuning methods like LoRA enable domain adaptation with minimal computational overhead, updating less than 1% of model parameters while achieving 95% of full fine-tuning performance (E. J. Hu et al. 2021). For banking applications with well-defined, slowly-changing regulatory knowledge, fine-tuning may offer superior cost-effectiveness compared to maintaining retrieval infrastructure.

Advanced prompt engineering techniques, including chain-of-thought prompting and few-shot learning, can elicit knowledge already encoded in large language models without external retrieval (Wei, Xuezhi Wang et al. 2022). These approaches prove particularly effective when queries align with training data distributions, achieving comparable performance to RAG on tasks where the model has sufficient internal knowledge while eliminating retrieval latency and infrastructure costs entirely.

These comparisons between RAG and alternatives warrant careful interpretation, as the literature suffers from inconsistent evaluation conditions. Fine-tuning studies typically evaluate on closed-book benchmarks with stable knowledge, while RAG studies emphasise open-domain scenarios with evolving information needs. The claim that fine-tuning achieves superior performance obscures critical context because fine-tuned models cannot accommodate new information without retraining, making direct comparison with RAG problematic for dynamic knowledge bases. Similarly, the assertion that prompt engineering achieves comparable performance to RAG reflects performance parity only on the subset of tasks where LLMs already encode relevant knowledge, and does not imply general substitutability. For enterprise deployments handling proprietary, post-training-cutoff information, RAG remains necessary regardless of these benchmark comparisons.

2.9 Reranking and Hybrid Retrieval Methods

Production retrieval systems commonly employ multi-stage pipelines that balance efficiency and accuracy through complementary retrieval mechanisms. This section examines reranking strategies and hybrid retrieval approaches that combine sparse lexical matching with dense semantic search, techniques central to modern RAG deployments and directly relevant to the architectural variants evaluated in this thesis.

Theoretical Foundations from Information Retrieval

Modern RAG systems build upon decades of information retrieval research, yet recent literature often overlooks these theoretical foundations. Classical IR metrics like BM25 remain competitive with neural approaches in many settings, particularly for exact match and keyword-based queries (Robertson and Zaragoza 2009). The theoretical framework of BM25, based on probabilistic relevance models and term frequency-inverse document frequency weighting, provides important insights for hybrid retrieval systems.

The Normalised Discounted Cumulative Gain (NDCG) metric, widely used in IR evaluation, offers a principled approach to measuring retrieval quality that accounts for graded relevance and position bias (Järvelin and Kekäläinen 2002). Recent RAG evaluations using NDCG reveal that simple BM25 retrieval achieves within 15% of neural retrieval performance on 60% of queries while requiring $100\times$ less computational resources (Thakur et al. 2021). This finding challenges the assumption that neural retrieval universally outperforms classical methods and suggests hybrid approaches may offer optimal cost-quality trade-offs.

Two-Stage Retrieval

Two-stage retrieval architectures decouple the dual objectives of maximizing recall during candidate generation and optimizing precision during relevance scoring. The initial retrieval stage prioritizes speed over accuracy to identify a broad set of potentially relevant documents, often targeting a candidate pool ranging from 50 to 1000 documents depending on system latency constraints. Subsequently, the second stage applies computationally intensive reranking models to re-order this candidate set, selecting only the highest-scoring documents for the generation phase (Nogueira and Cho 2019).

This staged approach enables systems to handle document collections scaling to millions of entries while maintaining high precision. Dual-encoder models, often referred to as bi-encoders, dominate the first stage due to their ability to pre-compute document representations offline and perform retrieval via efficient approximate nearest neighbour search. Cross-encoder models, which jointly encode query and document pairs, provide superior relevance judgements in the second stage at the cost of requiring $O(n)$ forward passes for n candidates (Reimers and Gurevych 2019).

Cross-Encoder Reranking

Cross-encoder architectures process query-document pairs through a single transformer encoder, enabling fine-grained attention between query and document tokens. This joint encoding captures subtle relevance signals invisible to dual-encoders, which compute query and document representations independently. Models such as MonoT5, trained on passage ranking datasets like MS MARCO, demonstrate substantial improvements over dual-encoder baselines, particularly for queries requiring semantic nuance or negation handling (Nogueira, Z. Jiang and Lin 2020).

Recent cross-encoder variants optimise for production constraints. BGE-re-ranker models employ distillation and quantisation to reduce inference latency while maintaining ranking quality, achieving sub-100ms reranking for top-20 candidates on consumer hardware (J. Chen et al. 2024). These efficiency improvements enable reranking stages in latency-sensitive applications such as conversational search and real-time question answering, where total response time budgets may be 1–2 seconds.

The inherent trade-off between cost and quality in reranking necessitates careful tuning of the candidate set size. Retrieving too few candidates creates a recall ceiling where relevant documents are missed entirely, whereas retrieving an excessive number increases latency and computational cost without yielding proportional improvements in final precision. Empirical research suggests that retrieving between 50 and 100 candidates for reranking to a final set of size top-k, where $k \in \{10, 20\}$, effectively balances these concerns for the majority of enterprise search tasks (Khattab and Zaharia 2020; Nogueira and Cho 2019).

Late Interaction Models

Late interaction architectures such as Contextualized Late Interaction over BERT (ColBERT) occupy the design space between dual-encoders and cross-encoders, decomposing relevance scoring into independent token-level representations and delayed query-document interaction (Khattab and Zaharia 2020). Unlike dual-encoders, which collapse document representations into single vectors, ColBERT retains per-token embeddings and computes relevance through maximum similarity matching between query and document token sets. This preserves some of the expressiveness of cross-encoders while maintaining the efficiency of pre-computed document representations.

ColBERT's late interaction mechanism enables fine-grained matching, such as identifying that "payment gateway" in a query corresponds to "API for processing payments" in a document, without requiring query-time joint encoding. The model achieves cross-encoder-level effectiveness at dual-encoder-level efficiency, making it particularly suitable for domains with specialised vocabulary where exact term matching is insufficient but full cross-attention is prohibitively expensive (Santhanam et al. 2022).

Hybrid Retrieval

Hybrid retrieval systems fuse lexical sparse signals with semantic dense retrieval signals, leveraging the complementary strengths of BM25 and learned embeddings. BM25 excels at exact keyword matching, proper noun retrieval, and queries with rare or domain-specific terms, while dense retrieval captures semantic similarity, handles paraphrasing, and generalises across synonyms (Robertson and Zaragoza 2009; Karpukhin et al. 2020).

Fusion strategies combine scores from multiple retrievers into unified rankings. Linear interpolation with learned or grid-searched weights α provides a simple baseline:

$$s_{\text{final}} = \alpha s_{\text{dense}} + (1 - \alpha) s_{\text{sparse}} \tag{2.1}$$

Reciprocal Rank Fusion (RRF), which aggregates rankings rather than scores, offers robustness to score distribution differences and requires no training (Cormack, Clarke and Buettcher 2009). Advanced fusion approaches employ learned-to-rank models that treat retrieval scores as features, optimising ranking objectives directly (Burges 2010).

Empirical comparisons consistently demonstrate hybrid methods outperform single-retriever baselines across diverse benchmarks. On MS MARCO passage ranking, BM25+Dense fusion achieves 5–10% relative gains over dense-only retrieval, with larger improvements on queries containing rare entities or technical terminology (Y. Luan et al. 2021). Hybrid approaches prove particularly valuable for enterprise search within specialised domains such as legal, medical, and technical documentation, as they combine domain-agnostic semantic understanding with precise term matching for jargon and entity names.

Operational Considerations for Production Deployments

Deploying reranking and hybrid retrieval in production systems requires careful engineering to balance quality, latency, and cost. BM25 indexing via inverted indices incurs minimal storage overhead, typically requiring only 10–20% of the raw corpus size, and imposes negligible query-time costs of less than 10ms for million-document collections. These characteristics make hybrid approaches attractive for cost-constrained deployments. Conversely, cross-encoder reranking adds between 50 and 200ms of latency per query depending on candidate set size and model complexity. This necessitates GPU acceleration or model quantisation for interactive applications (Guo et al. 2020).

Index maintenance costs differ substantially between sparse and dense components. BM25 indices support efficient incremental updates where adding or modifying a document requires only updating postings lists for affected terms, which is an $O(d)$ operation for a document of d terms. Dense indices require re-embedding changed documents and updating approximate nearest neighbour structures such as HNSW. This process incurs embedding API costs and index reorganisation overhead. This asymmetry makes hybrid approaches particularly compelling for dynamic knowledge bases requiring frequent updates, as explored in this thesis.

Caching strategies provide additional optimisation opportunities. Query result caching benefits both sparse and dense components, while re-ranker output caching is especially valuable given cross-encoder latency. Negative caching, which involves storing queries that yield zero results, prevents repeated searches over irrelevant corpus segments. For enterprise applications with predictable query distributions, proactive warming of popular query caches can eliminate cold-start latency (Cambazoglu and Kayaaslan 2010).

2.10 Document Ingestion and Optical Character Recognition Models

The ability to extract text and structured information from visual documents represents a foundational capability for modern information retrieval systems. Enterprise corpora in banking and financial services inevitably include scanned documents, legacy Portable Document Formats (PDFs) without embedded text, form submissions, and mixed-layout regulatory filings. Reliable Optical Character Recognition (OCR) and document understanding technologies are therefore first-class dependencies for retrieval systems that must operate over heterogeneous document collections (Smith 2007).

Evolution of Document Understanding

OCR technology has evolved from early template matching systems to modern deep learning architectures. Traditional systems like Tesseract relied on manual feature engineering and struggled with complex layouts (Smith 2007). The deep learning revolution introduced CNNs and Recurrent Neural Networks (RNNs) that treat text recognition as a sequence-to-sequence problem, significantly improving accuracy on degraded documents (Shi et al. 2017).

Recent advances have moved towards multi-modal vision-language models that jointly understand text, layout, and visual features. Models like LayoutLM and Donut demonstrate state-of-the-art performance by processing document images directly, eliminating the error propagation inherent in multi-stage pipelines (Xu et al. 2020; G. Kim et al. 2022). Cloud-based services such as Amazon Web Services (AWS) Textract operationalise these advances, offering production-grade reliability and specialised models for tables and forms (Amazon Web Services 2024b).

OCR Quality and Retrieval Impact

For retrieval-augmented generation, OCR errors present distinct challenges. Character-level errors can degrade sparse retrieval, or BM25, performance by preventing exact term matching, while semantic retrieval using dense embeddings exhibits greater tolerance to noise (Robertson and Zaragoza 2009; Karpukhin et al. 2020). However, systematic errors in domain-specific terminology or numerical data can severely compromise retrieval precision and generation factuality.

Engineering mitigations include preserving spatial coordinates for citation verification, propagating confidence scores to down-weight unreliable chunks, and employing hybrid retrieval strategies to balance precision and recall. For financial documents, validating numerical accuracy is particularly critical, as OCR misrecognitions, such as errors with decimal points, can alter semantic meaning entirely.

Approaches to Document Ingestion in Enterprise Contexts

Effective document ingestion systems must handle diverse financial and technical documents with varying quality levels, layouts, and formats. Cloud-native solutions, such as AWS Textract, provide comprehensive document understanding capabilities that extend beyond basic character recognition to include form recognition, table structure extraction, and key-value pair identification. These services employ multiple specialised neural models optimised

for different document types, automatically selecting appropriate models based on input characteristics to achieve robust performance across heterogeneous collections.

Synchronous APIs offer low-latency processing suitable for interactive retrieval applications, returning structured responses containing recognised text, bounding boxes, and confidence scores. For production deployments requiring higher throughput, asynchronous processing via object storage supports batch processing of large document collections. Confidence scoring at the line and word level enables quality-aware retrieval strategies, allowing systems to flag low-confidence documents for manual review, a critical feature for high-stakes financial applications.

Alternative approaches include open-source solutions based on Tesseract or fully end-to-end models like Donut. Tesseract offers the advantages of local execution and transparent processing but may struggle with complex layouts compared to cloud-native alternatives. The choice between these approaches typically involves balancing accuracy requirements, data privacy constraints, and integration complexity within the broader enterprise architecture.

Implications for Retrieval-Augmented Generation

The characteristics of OCR output directly impact retrieval system design and performance. Unlike born-digital text, OCR-extracted content contains recognition errors that affect both lexical and semantic matching. Sparse retrieval methods based on exact term matching suffer disproportionately from OCR errors, as character substitutions, insertions, and deletions prevent lexical overlap between queries and relevant documents (Robertson and Zaragoza 2009).

Dense retrieval using learned embeddings demonstrates greater robustness to OCR noise, as pre-trained language models can map corrupted text to semantically meaningful representations (Karpukhin et al. 2020). However, systematic OCR errors, such as consistent misrecognition of domain-specific terminology, mathematical notation, or financial symbols, can still significantly degrade retrieval performance. Hybrid approaches that combine lexical and semantic matching benefit from complementary strengths where lexical matching provides high precision for correctly recognised terms, while semantic matching provides recall on passages with recognition errors.

For financial documents, preservation of numerical accuracy is critical. OCR errors in numerical data, such as misrecognition of decimal points, confusion between digits such as 0 and O, 1 and l, or 5 and S, or transposition of digits, can completely change the meaning of financial information. The document ingestion pipeline must implement validation and error detection for numerical content, potentially employing checksum verification, format validation, and consistency checking against structured data sources.

2.11 Summary

This literature review has examined the foundational technologies necessary for evaluating retrieval-augmented generation systems in enterprise contexts, including language model architectures, embedding methods, graph-theoretic retrieval, reranking strategies, hybrid approaches, and document understanding pipelines. The integration of dense semantic retrieval, sparse lexical matching, cross-encoder reranking, and graph-augmented reasoning enables the development of intelligent systems that can understand complex financial contexts, retrieve relevant information dynamically, and generate accurate, compliant responses.

However, a critical gap exists in existing literature regarding the *operational economics* of maintaining RAG systems over time. While initial indexing costs are occasionally mentioned, the marginal costs of re-indexing as knowledge bases evolve remain largely unexplored in academic literature. This gap is partially explained by the incentive structures of academic publishing, which reward novel architectures and benchmark improvements over operational considerations. The MLOps literature addresses infrastructure cost modeling for machine learning systems generally (Sculley et al. 2015; Paleyes, Urma and Lawrence 2022), identifying "hidden technical debt" in ML systems and the challenge of maintaining models in production. However, this work predates the RAG paradigm and does not address the specific cost structures of retrieval-augmented systems, particularly the distinction between query-time and index-time costs that proves critical for dynamic corpora.

Industry analyses from cloud providers offer pricing documentation but not systematic cost-performance comparisons across architectures (Amazon Web Services 2024a). Practitioner blog posts and conference talks frequently discuss RAG deployment challenges anecdotally, but lack the rigorous methodology required for generalisable conclusions. Enterprise knowledge bases are not static artefacts but living systems that receive continuous updates through document modifications, regulatory changes, and corpus expansion. The cost-performance trade-offs for dynamic corpora may differ substantially from those observed in static benchmark evaluations (W. X. Zhao et al. 2023), yet limited published work systematically quantifies these differences.

The need for explainability and auditability in financial applications presents additional challenges. Graph-augmented approaches offer promise through explicit reasoning paths and entity provenance tracking, but incur higher computational costs. Hybrid retrieval methods combining complementary signals may provide favourable trade-offs. This applies when exact keyword matching is critical for compliance terminology and regulatory citations. Understanding when these additional costs are justified and when simpler vector-based approaches suffice remains an open question with significant practical implications for production deployments.

The synthesis of these techniques enables systematic comparison of architectural variants across the dimensions most relevant to production deployments, encompassing not only retrieval quality, but also indexing costs, re-indexing costs, query latency, and operational viability under continuous update scenarios. The principles and techniques discussed here provide the foundation for the empirical evaluation presented in subsequent chapters.

CHAPTER 3

Research Design

This chapter presents a constructive research methodology (Kasanen, Lukka and Siitonen 1993) for systematically evaluating RAG architectures under operationally realistic conditions. The study designs and implements four architectural variants comprising Vector RAG, Vector + Re-ranker, Hybrid RAG, and GraphRAG. Each variant undergoes controlled experiments measuring initial indexing costs, projected re-indexing costs under different update frequencies, query latency distributions, and retrieval quality across query categories. The methodology addresses the gap identified in Chapter 2 where existing evaluations focus on static benchmark performance while production deployments require continuous index maintenance as knowledge bases evolve. The experimental design isolates the cost-performance trade-offs of each architecture by holding corpus characteristics constant while simulating update patterns including 5% minor edits, 20% major edits, and 10% corpus growth. The study varies model choices across four AWS Bedrock embedding models spanning a significant cost gradient for vector-based architectures. It also evaluates five AWS Bedrock LLMs for the entity extraction pipeline in GraphRAG. This structured approach enables direct comparison of operational economics to determine the financial viability of maintaining these systems alongside their functional performance.

Traditional RAG research treats knowledge bases as static artefacts—inscribed once, queried indefinitely. Yet this assumption fractures in production environments where Confluence pages update daily, Jira tickets evolve hourly, regulatory documents change weekly, and risk reports refresh continuously. The research question therefore pivots to *re-indexing costs* as a critical factor in architectural selection. This metric captures the marginal expense and computational overhead of maintaining indices as documents change—a factor that has received limited attention in existing literature despite its operational significance.

3.1 Experiment Outline

To ensure reproducibility and strict isolation of experimental variables, the study employs a custom **Three-Phase Orchestrator** implemented in `experiment_orchestrator.py` which manages the end-to-end lifecycle of each experimental run. As illustrated in Figure 3.1, this software harness enforces a clean separation between data preparation in Phase 1, system indexing in Phase 2, and performance evaluation in Phase 3 to prevent state leakage between configurations.

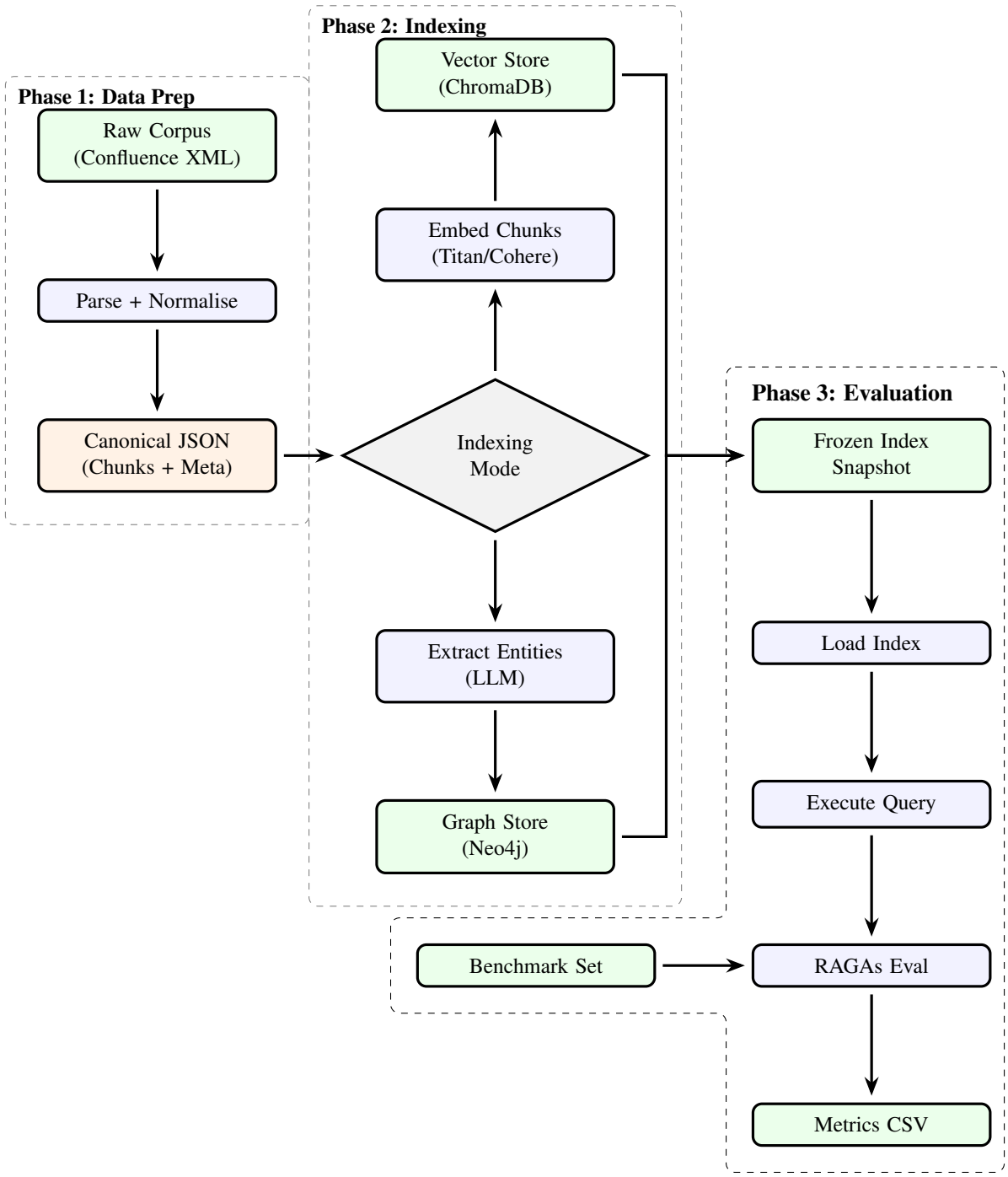


FIGURE 3.1. GraphRAG evaluation pipeline from raw Confluence XML to evaluation metrics.

Phase 1: Data Preparation

Thtransforms raw source artefacts into an intermediate representation, ensuring all variants of architecture operate on an identical data substrate. As detailed in the **Phase 1: Data Prep** group of Figure 3.1, the process begins with the **Raw Corpus** comprising an export from Confluence, formatted in Extensible Markup Language (XML). A custom **Parse + Normalise** module traverses the Document Object Model (DOM) tree to extract page content, metadata, and attachment references. Following extraction, the raw content undergoes a rigorous sanitisation process to strip navigation elements and non-semantic markup. The final processed documents are then serialised into the **Canonical JSON** store, creating an immutable dataset that serves as the single source of truth for all subsequent experimental phases.

Phase 2: Indexing

The second phase orchestrates the systematic construction of databases. Here, the orchestrator's **Indexing Mode** decision node iterates through the defined matrix of architectures to provision isolated data stores. For vector architectures, chunks flow through the **Embed Chunks** process using models like Titan v2 or Cohere v3 before being persisted in the **Vector Store**, which in this case uses the ChromaDB software framework. For GraphRAG configurations, the **Extract Entities** pipeline uses an LLM to identify nodes and relationships, populating the **Graph Store**, which for this experiment is using the Neo4j database. This phase isolates the computationally intensive indexing operations from query-time performance measurements.

Phase 3: Evaluation

In the final phase, the system executes the **Benchmark Set** against frozen indices, strictly enforcing isolation to prevent query-time contamination. To ensure this strict isolation, the system creates a **Frozen Index Snapshot** of the completed state. The **Load Index** process restores this snapshot before the **Execute Query** engine runs the contributed benchmark questions. High-precision timers record latencies for embedding, retrieval, and generation. Generated answers and retrieved contexts are then passed to the **RAGAs Eval** framework, which computes quality scores. Finally, all performance data is aggregated into an output CSV file, known as the **Metrics CSV** report for later statistical analysis.

3.2 Experimental Architecture Design

The study juxtaposes four retrieval architectures under strictly controlled conditions. This design isolates the effect of architectural choices on cost, latency, and retrieval quality while eliminating confounding variables. The selected architectures represent a progression in complexity that begins with baseline vector retrieval and advances to two-stage retrieval with reranking, hybrid sparse-dense fusion, and finally graph-augmented retrieval.

Vector RAG

The baseline architecture implements dense embedding-based semantic search using an HNSW approximate nearest neighbour index in ChromaDB. The retrieval pipeline follows a straightforward sequence starting with the embedding of the incoming query using the configured model. A cosine similarity search then retrieves the top $k = 20$ most semantically similar chunks which are ranked by similarity score. Finally, Claude Sonnet 4.5 (Anthropic 2024) generates an answer grounded in the retrieved context.

Serving as the experimental baseline, this architecture anchors the analysis, establishing referential performance for cost, latency, and quality against which more sophisticated approaches are measured. Vector RAG represents the simplest production-viable architecture with minimal operational overhead, making it the natural starting point for enterprises exploring retrieval-augmented generation. The implementation resides in `Experimentation/vector_rag/` with modular components for embeddings via AWS Bedrock, `tiktoken`-based chunking, ChromaDB vector storage, and AWS Bedrock generation.

Vector RAG + Cross-Encoder Reranking

The second architecture extends the baseline with a two-stage retrieval approach incorporating cross-encoder reranking. The first stage retrieves a larger candidate set of 100 chunks using the same vector retrieval mechanism as the baseline architecture. The second stage then applies the AWS Bedrock Cohere Rerank v3.5 model which jointly encodes query-document pairs and computes relevance scores through fine-grained cross-attention. The system filters this set to retain only the top 20 reranked chunks for generation.

Cross-encoders achieve superior relevance judgements compared to dual-encoder or bi-encoder models by enabling direct interaction between query and document representations during scoring. This architectural choice introduces additional query-time cost because reranking via the API adds latency and per-query expense compared to pure vector retrieval. Critically, however, indexing and re-indexing costs remain identical to the baseline architecture as reranking occurs only at query time without requiring index modifications.

This architecture tests whether the improved precision from reranking justifies the added query-time latency and cost by evaluating a cost-quality trade-off common in production retrieval systems. The implementation, located in `reranker.py`, uses the AWS Bedrock Runtime API with automatic fallback to vector scores on failure. It limits the batch size to 100 documents to comply with Cohere API constraints.

Hybrid RAG

The third architecture combines lexical sparse retrieval using BM25 (Robertson and Zaragoza 2009) with dense semantic retrieval through score fusion. The BM25 component uses a local inverted index built with the `rank_bm25` Python library and applies standard Okapi BM25 parameters where k_1 equals 1.5 and b equals 0.75. Dense retrieval proceeds identically to the baseline architecture. Both retrievers independently return their top 50 candidates. The system then normalises these scores to the zero-to-one range via min-max scaling and combines them using a weighted linear combination.

$$s_{\text{final}}(d, q) = \beta \cdot s_{\text{dense}}(d, q) + (1 - \beta) \cdot s_{\text{BM25}}(d, q)$$

The weighting parameter β is set to 0.7 to favour dense scores based on empirical tuning on a held-out validation set. The top 20 fused candidates then proceed to generation. Algorithm 1 details this retrieval procedure.

Algorithm 1 Hybrid RAG Retrieval

Require: Query q, vector index $\mathcal{V}$, BM25 index $\mathcal{B}$, candidates k, output size r, weight β
Ensure: Top-r ranked documents $\mathcal{D}_r$

$\mathcal{D}_{\text{dense}} \leftarrow \text{VectorSearch}(\mathcal{V}, q, k)$ {Dense retrieval}
$\mathcal{D}_{\text{sparse}} \leftarrow \text{BM25Search}(\mathcal{B}, q, k)$ {Sparse retrieval}
$\mathcal{D}_{\text{union}} \leftarrow \mathcal{D}_{\text{dense}} \cup \mathcal{D}_{\text{sparse}}$
$S_{\text{dense}} \leftarrow \{s_{\text{dense}}(d) : d \in \mathcal{D}_{\text{union}}\}$ {Collect raw scores}
$S_{\text{BM25}} \leftarrow \{s_{\text{BM25}}(d) : d \in \mathcal{D}_{\text{union}}\}$
$S_{\text{dense}} \leftarrow \text{MinMaxScale}(S_{\text{dense}})$ {Normalise to [0,1]}
$S_{\text{BM25}} \leftarrow \text{MinMaxScale}(S_{\text{BM25}})$
for each document $d \in \mathcal{D}_{\text{union}}$ **do**
    $s_{\text{final}}(d) \leftarrow \beta \cdot S_{\text{dense}}[d] + (1 - \beta) \cdot S_{\text{BM25}}[d]$
end for
$\mathcal{D}_r \leftarrow \text{TopK}(\mathcal{D}_{\text{union}}, r, s_{\text{final}})$ {Select top-r}
return $\mathcal{D}_r$

This hybrid approach leverages complementary retrieval strengths where BM25 excels at exact keyword matching and proper noun retrieval while dense embeddings capture semantic similarity and handle paraphrasing effectively. The combination proves particularly valuable for enterprise domains with specialised terminology where exact term matching complements semantic understanding.

From an operational perspective, BM25 indexing incurs minimal overhead. The inverted index requires minimal additional storage and builds rapidly. Query-time BM25 scoring adds negligible latency since it operates on precomputed postings lists. Most importantly for dynamic knowledge bases, BM25 index updates are incremental and fast which preserves the low re-indexing costs characteristic of the baseline architecture. The implementation extends `vector_rag/` with `hybrid_retriever.py` which coordinates parallel BM25 and vector retrieval, score normalisation, and weighted fusion. The BM25 index is stored alongside ChromaDB and automatically synchronised.

GraphRAG

The fourth and most sophisticated architecture abandons vector retrieval for a knowledge graph, capturing entity relationships and semantic communities to mimic human associative reasoning. Implementing Microsoft's GraphRAG methodology (Edge et al. 2024), the system constructs a semantic graph stored in Neo4j through entity extraction, relationship modelling, and Louvain community detection (Blondel et al. 2008). This algorithm groups densely connected graph nodes into "communities" representing thematically related concepts.

The graph construction follows a five-stage pipeline where entities and relationships are first extracted from each chunk using an LLM with schema-constrained prompts. Entities appearing across multiple chunks are then consolidated and summarised. Relationships between entities are similarly summarised to capture cross-document connections. The Louvain algorithm subsequently detects communities of related entities before community summaries are generated to provide high-level semantic context for retrieval. Algorithm 2 formalises this indexing procedure.

Algorithm 2 GraphRAG Index Construction

Require: Corpus chunks $\mathcal{C}$, LLM $\mathcal{M}$, entity types $\mathcal{T}$
Ensure: Knowledge graph $\mathcal{G} = (V, E)$, community summaries $\mathcal{S}$
$V \leftarrow \varnothing$; $E \leftarrow \varnothing$
for each chunk $c \in \mathcal{C}$ **do**
  $(entities, relations) \leftarrow$ LLM$(\mathcal{M}, c, \mathcal{T})$ {Entity extraction}
  Accumulate descriptions for $entities$ and $relations$
  $V \leftarrow V \cup entities$
  $E \leftarrow E \cup relations$
end for
$\mathcal{G} \leftarrow (V, E)$ {Construct graph}
for each entity $e \in V$ **do**
  $descriptions \leftarrow$ GetAccumulatedDescriptions(e)
  $e.summary \leftarrow$ LLM$(\mathcal{M}, descriptions)$ {Entity summarisation}
end for
for each relation $r \in E$ **do**
  $descriptions \leftarrow$ GetAccumulatedDescriptions(r)
  $r.summary \leftarrow$ LLM$(\mathcal{M}, descriptions)$ {Relationship summarisation}
end for
$communities \leftarrow$ Louvain$(\mathcal{G})$ {Community detection}
for each community $k \in communities$ **do**
  $\mathcal{S}_k \leftarrow$ LLM$(\mathcal{M}, \text{entities}(k), \text{relations}(k))$ {Community summaries}
end for
return $\mathcal{G}, \mathcal{S}$

At query time, the system performs hybrid local-global search where local search links query terms to graph entities via keyword matching and performs multi-hop graph traversal with a default depth of $h = 2$. Global search retrieves community summaries to provide high-level context. The combined context consisting of entity descriptions, relationship summaries, and community overviews is then passed to the LLM for answer generation. Algorithm 3 formalises this query procedure. Unlike vector-based architectures, GraphRAG retrieval operates entirely through graph traversal and text matching where no embedding similarity is computed at query time. This architecture prioritises complex reasoning and multi-hop question answering by excelling at queries requiring entity relationship traversal, such as determining connections between distinct projects or teams, and cross-document synthesis.

Algorithm 3 GraphRAG Query Retrieval

Require: Query q, graph $\mathcal{G}$, community summaries $\mathcal{S}$, traversal depth h
Ensure: Context $\mathcal{X}$ for answer generation
$keywords \leftarrow$ ExtractKeywords(q) {Query analysis}
$E_{\text{match}} \leftarrow \{e \in V : \text{Match}(e, keywords)\}$ {Entity linking}
$\mathcal{X}_{\text{local}} \leftarrow \varnothing$
for each entity $e \in E_{\text{match}}$ **do**
  $neighbors \leftarrow$ Traverse$(\mathcal{G}, e, h)$ {Multi-hop traversal}
  $\mathcal{X}_{\text{local}} \leftarrow \mathcal{X}_{\text{local}} \cup \{e.summary\} \cup \{n.summary : n \in neighbors\}$
end for
$K_{\text{relevant}} \leftarrow \{k \in \text{Communities} : \text{Score}(\mathcal{S}_k, q) > \theta\}$ {Global search}
$\mathcal{X}_{\text{global}} \leftarrow \{\mathcal{S}_k : k \in K_{\text{relevant}}\}$
$\mathcal{X} \leftarrow \mathcal{X}_{\text{local}} \cup \mathcal{X}_{\text{global}}$ {Combine contexts}
return $\mathcal{X}$

The enhanced capabilities come at substantial operational cost because entity extraction requires LLM calls on every chunk during indexing. Relationship summarisation and community detection add computational overhead while Neo4j storage increases infrastructure requirements. Re-indexing costs prove particularly significant as document changes may trigger community recomputation across the entire graph, which is a key consideration for dynamic knowledge bases. The implementation resides in `Experimentation/layer2_rag/` where `pipeline.py` orchestrates extraction, community detection, and Neo4j integration. Additionally, `layer1_integration.py` provides access to the parsed text chunks from the first architecture for entity extraction.

GraphRAG Entity Extraction Models

Unlike vector-based architectures where costs vary only by embedding model choice, GraphRAG operational expense is dominated by the LLM used for entity extraction during graph construction. To investigate this critical cost driver, the study evaluates five AWS Bedrock LLMs spanning a 72× cost gradient as detailed in Table 3.1.

TABLE 3.1. AWS Bedrock LLM configurations for GraphRAG entity extraction.

Model	Provider	Input Cost	Output Cost	Price Tier
us.amazon.nova-micro-v1:0	Amazon	$0.037/1M	$0.148/1M	Budget
us.amazon.nova-lite-v1:0	Amazon	$0.063/1M	$0.252/1M	Budget
us.amazon.nova-pro-v1:0	Amazon	$0.84/1M	$3.36/1M	Mid-tier
us.anthropic.claude-3-5-haiku-20241022-v1:0	Anthropic	$1.00/1M	$5.00/1M	Premium
us.anthropic.claude-sonnet-4-5-20250929-v1:0	Anthropic	$3.00/1M	$15.00/1M	Premium

This selection tests whether smaller and cost-efficient models such as Nova Micro or Nova Lite can match the entity extraction quality of frontier models like Claude Sonnet 4.5. This comparison evaluates the potential to reduce GraphRAG prohibitive indexing costs while maintaining retrieval quality.

Controlled Variables

To enable valid comparison across architectures, all experimental conditions share a common set of controlled variables spanning corpus, generation, evaluation, hardware, and measurement dimensions.

The document corpus consists of a Confluence AI Squad workspace export containing 63,205 chunks totalling approximately 10 GB of raw documents. Chunking employs a token-based strategy using `tiktoken` (OpenAI 2024b) with cl100k_base encoding, producing chunks of 512 tokens with 50-token overlap to preserve context across chunk boundaries.

For answer generation, all architectures use Claude Sonnet 4.5 via AWS Bedrock with temperature set to zero for deterministic evaluation, max_tokens of 2048, and identical system prompt templates. The model identifier is `us.anthropic.claude-sonnet-4-5-20250929-v1:0`. The evaluation benchmark comprises 40 questions distributed across five categories including factual, detail, synthesis, status, and cross-reference, each annotated with ground-truth relevant document IDs.

Hardware and network conditions are standardised across all runs using a MacBook M2 Pro with 32 GB RAM running macOS Tahoe 26.1 build 25B78. Network bandwidth is validated at 323 Mbps download and 360 Mbps upload with all API endpoints accessed from the same geographic region. Measurement instrumentation employs the `evaluation.py` framework coordinated by the `experiment_orchestrator.py` Three-Phase Orchestrator to ensure consistent timing via `time.perf_counter()`, token counting via `tiktoken`, and cost tracking across all runs. All pricing calculations reference AWS Bedrock rates as of 2025-11-30. The test protocol executes five complete runs per condition under both cold and warm cache states, with sequential query execution and reporting of the 50th, 95th, and 99th percentiles.

Independent Variables

The experimental design manipulates five key independent variables to isolate the effects of architectural and model choices on RAG performance.

The first variable is the retrieval mechanism which varies between vector cosine similarity for Architecture 1, cross-encoder reranking for Architecture 2, hybrid sparse-dense fusion for Architecture 3, and graph-augmented retrieval with entity linking and multi-hop traversal for Architecture 4. The second variable is indexing complexity which ranges from embeddings only for the first three architectures to the computationally intensive combination of embeddings, entity extraction, relationship extraction, and community detection required by Architecture 4. Notably, Architectures 2 and 3 add no indexing overhead since reranking and BM25 scoring occur at query time rather than during index construction.

The third variable is index storage requirements which differ across architectures. Architectures 1 and 2 utilise ChromaDB vector store alone while Architecture 3 adds a BM25 inverted index and Architecture 4 incorporates a Neo4j graph database alongside ChromaDB. The fourth variable is context composition determined by each architecture's ranking method. This includes cosine similarity scores for Architecture 1, cross-encoder relevance scores for Architecture 2, fused BM25 and vector scores for Architecture 3, or graph-enriched context incorporating entity coverage and path support for Architecture 4.

The fifth and final variable is the embedding model which is varied across four AWS Bedrock models per architecture. These include Amazon Titan v2 identified as `titan-embed-text-v2:0` with 1024 dimensions at a cost of $0.02 per million tokens, Amazon Titan v1 identified as `titan-embed-text-v1` with 1536 dimensions at $0.10 per million tokens, Cohere English v3 identified as `embed-english-v3` with 1024 dimensions at $0.10 per million tokens, and Cohere Multilingual v3 identified as `embed-multilingual-v3` with 1024 dimensions at $0.10 per million tokens. This design yields 17 experimental conditions comprising 12 vector-based combinations and 5 GraphRAG variants varying by entity extraction LLM.

3.3 Measurement Methodology

3.3.1 Data Preparation

Corpus Selection Rationale

The decision to use Constantinople's proprietary Confluence workspace as the primary corpus, despite reproducibility limitations, reflects deliberate methodological priorities that favour ecological validity. While standard RAG benchmarks such as MS MARCO, Natural Questions, and HotpotQA offer clean, well-structured text primarily derived from Wikipedia, they fail to capture the chaotic reality of corporate knowledge bases. Enterprise repositories are characterised by heterogeneous formats—ranging from HyperText Markup Language (HTML) and PDFs to scanned images—and inconsistent structural quality, where informal meeting notes coexist with formal specifications and domain-specific terminology. Consequently, evaluating RAG architectures exclusively on sanitised academic benchmarks would yield findings with limited applicability to the production deployments that this research targets.

Furthermore, the core research question regarding re-indexing costs in dynamic environments necessitates a corpus with realistic update patterns. Static academic datasets cannot simulate the operational complexity of a living knowledge base. Constantinople's Confluence workspace provides these authentic update scenarios, including Jira integrations, continuous wiki edits, and complex document versioning, which are essential for accurately measuring the computational and financial impact of maintaining up-to-date retrieval indices.

To address the limitations inherent in using a proprietary corpus, the study incorporates external validation using the publicly available Haystack benchmark as detailed in Section 3.4, which provides independent verification of the key findings. Additionally, methodological reproducibility is supported through the public release of all experimental code, configuration files, and aggregated metrics in the thesis repository, allowing for the replication of the experimental apparatus even without access to the primary data. The constraints imposed by the proprietary nature of the corpus are explicitly acknowledged and discussed in the Limitations section of Chapter 6.

Sources and Staging

The primary corpus consists of a Confluence AI Squad workspace export containing exactly 13,390 pages across technical specifications, meeting notes, project documentation, and engineering runbooks. Documents include HTML and PDF exports with attachments of various format; Portable Network Graphics (PNG), Joint Photographic Experts Group (JPEG), PDF, etc, representing heterogeneous formats common in enterprise knowledge bases. Each document is assigned a stable `doc_id` and monotonically increasing `rev` to support versioning and incremental updates. The corpus totals approximately 10GB of raw content, which when processed yields 63,205 chunks for retrieval.

TABLE 3.2. Corpus statistics for the Confluence AI Squad workspace

Metric	Value
Total documents	12,214
Total pages	3,841
Raw corpus size	∼10.03 GB
Total chunks (512 tokens, 50 overlap)	63,205
Average chunks per document	5.2
Document formats	HTML, PDF, PNG, JPEG
Content types	Technical specs, meeting notes, runbooks, API docs

Secondary sources include Jira exports in Comma-Separated Values (CSV) and JavaScript Object Notation (JSON) formats, engineering handbooks as PDF files, and internal markdown repositories. The corpus provides realistic enterprise content with varied layouts including multi-column documents, both ruled and borderless tables, embedded figures, code blocks, and mixed-quality scans.

Document Ingestion Pipeline

The document ingestion follows a multi-stage pipeline optimised for enterprise content diversity. First, format detection classifies documents by Multipurpose Internet Mail Extensions (MIME) type and magic bytes to route them to appropriate parsers. Specifically, Confluence XML exports identified as `entities.xml` undergo custom parsing to extract page content while preserving hierarchical relationships and metadata. Second, content extraction employs native parsers for structured formats including BeautifulSoup for HTML, PyPDF2 for PDFs, and python-docx for Office documents. Confluence pages embedded in XML are extracted using regex patterns to isolate Character Data (CDATA) sections containing page body content. Third, OCR processing handles image-based content such as scanned PDFs, screenshots, and diagrams via AWS Textract (Amazon Web Services 2024b), with mixed documents receiving dual processing and result merging based on confidence scores. Fourth, a chunking strategy applies token-based chunking using `tiktoken` with the cl100k_base encoding, producing 512-token chunks with a 50-token overlap to balance context window constraints with semantic coherence. Finally, metadata enrichment tags each chunk with source document ID, page number, extraction method, confidence score, timestamp, and structural context to enable filtered retrieval and quality-aware ranking.

Document Processing and OCR

The study employs AWS Textract for optical character recognition and document understanding (Amazon Web Services 2024b). Textract provides multi-format support for PDF, PNG, JPEG, and Tagged Image File Format (TIFF) files without format-specific preprocessing. It performs layout analysis to automatically detect reading order and hierarchical block structure, alongside structured table recognition with cell-level content and coordinates. The service also includes form recognition for key-value pair extraction and provides confidence scoring at the word, line, and block levels for quality assessment.

The synchronous API endpoint `detect_document_text` accepts document bytes and returns structured JSON containing recognised text, page-relative bounding box coordinates, hierarchical block relationships spanning page, block, line, and word levels, and per-element confidence scores. This structured output enables quality-aware retrieval strategies and precise citation generation with page and coordinate references.

The implementation in `ocr.py` detects document formats via magic bytes including PNG, JPEG, and PDF signatures, invokes the Textract API, and extracts text from LINE-level blocks to preserve reading order while avoiding word-level concatenation errors.

Structured Content Preservation

Textract's hierarchical block structure is serialised to a Document Layout JSON (DLJ) format:

LISTING 3.1. Document Layout JSON structure

```
{
  "doc_id": "abc123",
  "pages": [
    {
      "page_num": 1,
      "blocks": [
        {"type": "LINE", "text": "...", "bbox": [x0,y0,x1,y1],
         "confidence": 0.98}
      ]
    }
  ]
}
```

This format preserves spatial information for citation generation, enabling precise attribution such as locating content on specific pages, and enables table-aware chunking where table cells are kept within single chunks to maintain semantic coherence.

OCR Processing for Image Attachments

The Confluence corpus contains numerous image attachments, including screenshots, diagrams, and charts embedded within page content. To ensure information contained solely in visual media is retrievable, the document processing pipeline integrates AWS Textract for optical character recognition during Stage 1 ingestion. The Confluence loader detects image attachments referenced via XML tags in page HTML. For each detected image; PNG, JPEG, Graphics Interchange Format (GIF), TIFF, AWS Textract's synchronous API extracts text content. Processing occurs in parallel using a ThreadPoolExecutor with up to 10 concurrent workers to minimise ingestion latency.

Extracted OCR text is injected directly into the page body, replacing the original image reference tags with formatted text blocks:

LISTING 3.2. OCR text injection format

```
[Image: screenshot.png]
<extracted text content>
[End of image]
```

This injection strategy ensures OCR-derived content is chunked and embedded alongside surrounding textual context, preserving semantic coherence and enabling retrieval of information that exists only in visual form. The AI Squad corpus contained 48 image attachments that underwent OCR processing, contributing extracted text to the final 63,205 chunks. OCR was enabled by default in all experimental runs.

A notable limitation is that AWS Textract performs text extraction only, without semantic understanding of charts, diagrams, or table structures. Complex visual content such as flowcharts and architecture diagrams yields partial text extraction that may lack relational context present in the original visual layout.

Quality Control and Confidence Handling

Textract returns mean confidence scores at block and line levels. Pages with mean confidence below 85% are flagged in chunk metadata with a `low_confidence` tag. These flags propagate to retrieval fusion scoring described in Section 3.3.3, enabling quality-aware ranking that down-weights unreliable chunks. For high-stakes applications, low-confidence pages can be queued for manual review or alternative processing.

Confidence scores also inform re-processing decisions where documents with systematic low confidence across multiple pages may indicate format issues such as password-protected PDFs or corrupted files requiring fallback extraction strategies or exclusion from the corpus.

The pipeline performs several additional processing steps. Normalisation converts text to Unicode Transformation Format - 8-bit (UTF-8) while preserving structure including headings, lists, and tables and removing navigation elements. De-duplication using MinHash/LSH on shingles keeps a canonical copy and records duplicates for audit. Personally Identifiable Information (PII) and secret guards provide optional masking for emails, account IDs, and API keys, using reversible salted hashes for audit trails. Finally, versioning ensures immutable document identifiers and monotonically increasing revision numbers, so re-indexing updates only affected chunks.

3.3.2 Evaluation

Overview of RAGAs

This research employs the RAGAs framework (Es et al. 2023) as the primary evaluation methodology for measuring RAG system performance. RAGAs provides a reference-free evaluation paradigm that assesses both retrieval and generation quality without requiring human-annotated ground truth answers—a critical advantage for evaluating systems on domain-specific corpora where creating comprehensive answer keys would be prohibitively expensive.

The framework decomposes RAG evaluation into four orthogonal dimensions that collectively capture system performance. These dimensions include faithfulness, which assesses factual grounding, answer relevance, which measures response quality, context relevance, which evaluates retrieval precision, and context recall, which gauges retrieval completeness. This multi-dimensional approach enables fine-grained diagnosis of system failures by distinguishing between retrieval failures, indicated by low context metrics, and generation failures, indicated by low faithfulness scores, that would otherwise be conflated in end-to-end accuracy measurements.

Benchmark Query Set

Novel Query Selection Methodology

A primary contribution of this thesis is the development of a **novel, ecologically valid query selection methodology**, expressly designed to bridge the gap between synthetic benchmarks and production reality. Unlike standard datasets that often lack the complexity of real-world enterprise information needs, this study manually curates 40 questions that strictly mirror the multi-hop, context-dependent nature of authentic engineering queries. The selection process followed a rigorous three-stage protocol to ensure that the benchmark serves as a realistic proxy for production usage.

To illustrate the depth of the data structure, Listing 3.3 provides a redacted example entry from the dataset, which is stored in standard CSV format to facilitate interoperability.

LISTING 3.3. Redacted example of a benchmark query item in CSV format

```
question,category,expected_topics,expected_pages
"What are the [REDACTED] dependency services required for the [REDACTED] deployment?",
```

Query selection followed a structured three-stage process designed to ensure ecological validity while maintaining experimental tractability.

The first stage involved eliciting information needs through informal interviews with three Constantinople engineers to identify recurring information retrieval patterns. Common needs included troubleshooting deployment issues, understanding API specifications, tracking project status, and locating cross-team dependencies which subsequently informed the five query categories.

The second stage focused on query generation where candidate queries were drafted for each category based on actual Confluence page titles, Jira ticket references, and documented use cases. This initial generation produced 60

TABLE 3.3. Benchmark query distribution by category

Category	Count	Description
Factual	8	Simple fact lookup requiring single document
Detail	8	Specific technical details from known sections
Synthesis	8	Multi-hop reasoning across multiple documents
Status	8	Temporal queries about project state or timeline
Cross-reference	8	Questions requiring entity relationship traversal

candidate queries which were subsequently filtered to remove duplicates, overly specific questions answerable from a single sentence, and questions requiring information not present in the corpus.

The third stage involved a balanced selection process where the final 40 queries were chosen to ensure equal representation across categories with eight per category and varying difficulty levels spanning single-document and multi-document synthesis. This sample size exceeds the minimum required for statistical power where 64 observations per condition are achieved through five runs per query and aligns with benchmark sizes in comparable retrieval studies.

Relevance Annotation

Each query includes manually annotated ground-truth relevant document IDs, with a mean of 3.2 relevant documents per query and a range of 1 to 7. Two independent annotators, consisting of the author and a Constantinople engineer familiar with the documentation, reviewed all relevance judgements using a binary relevance scale. Inter-rater reliability was measured using Cohen's Kappa, a statistical coefficient that accounts for chance agreement. The calculated value of $\kappa = 0.78$ indicates substantial agreement (Landis and Koch 1977). Disagreements, which occurred on 12% of document-query pairs, were resolved through discussion, with the more inclusive judgement retained when consensus could not be reached.

This annotation protocol has known limitations that affect result interpretation. First, the binary relevance scale of relevant versus not-relevant cannot capture graded relevance distinctions where a document that partially answers a query receives the same label as one that comprehensively addresses it. Graded scales, such as a 0 to 3 relevance score, would enable more nuanced evaluation but require significantly more annotation effort which was not feasible within the time constraints of an undergraduate thesis. Second, the "more inclusive judgement" tiebreaker introduces a systematic bias toward higher Context Recall at the expense of Context Precision, as borderline documents are included in the ground truth set. This bias is conservative for the primary research question regarding whether budget models suffice, as it makes retrieval failures harder to detect. Third, both annotators had domain expertise in the corpus content, which may inflate agreement rates compared to naive annotators but better reflects the expertise of end users who would query the system in production.

Core Metrics and Their Computation

Faithfulness

Faithfulness measures the factual consistency between generated answers and retrieved context on a scale from 0 to 1. The metric decomposes answers into atomic claims, then evaluates whether each claim is supported by the retrieved passages using an LLM-as-judge approach:

$$\text{Faithfulness} = \frac{|\text{Claims supported by context}|}{|\text{Total claims in answer}|}$$

A faithfulness score of 1.0 indicates perfect grounding, where every statement in the answer can be traced to retrieved evidence. Scores below 0.8 suggest hallucination or extrapolation beyond available context (Es et al. 2023).

Answer Relevance

Answer Relevance evaluates how well the generated answer addresses the original question, independent of factual accuracy, also on a 0 to 1 scale. The metric generates multiple paraphrased questions from the answer using reverse prompting, then measures semantic similarity between generated and original questions. This approach captures whether the answer stays on-topic and addresses all aspects of the query:

$$\text{Answer Relevance} = \frac{1}{n}\sum_{i=1}^{n} \frac{q_{\text{original}} \cdot q_{\text{generated}_i}}{\|q_{\text{original}}\| \|q_{\text{generated}_i}\|}$$

where n paraphrased questions are generated from the answer. Low scores indicate off-topic responses or partial answers that miss key query aspects.

Context Relevance

Context Relevance measures the precision of retrieved passages on a 0 to 1 scale, specifically what fraction of retrieved content is actually relevant to answering the question. Each retrieved chunk is classified as relevant or irrelevant using an LLM evaluator:

$$\text{Context Relevance} = \frac{|\text{Relevant chunks}|}{|\text{Total retrieved chunks}|}$$

Low context relevance indicates the retrieval system returns many false positives, increasing computational cost and potentially confusing the generation model with irrelevant information.

Context Recall

Context Recall assesses retrieval completeness on a 0 to 1 scale by measuring what fraction of the ground truth relevant passages were successfully retrieved. This requires annotated relevance judgements for the query set:

$$\text{Context Recall} = \frac{|\text{Retrieved relevant chunks}|}{|\text{Total relevant chunks in corpus}|}$$

Low recall indicates the retrieval system misses important information, limiting the generation model's ability to produce comprehensive answers.

Composite Score Calculation

RAGAs computes an overall score as the harmonic mean of the four component metrics, ensuring that poor performance in any dimension significantly impacts the composite score:

$$\text{RAGAs Score} = \frac{4}{\frac{1}{\text{Faith}} + \frac{1}{\text{Answer Relevance}} + \frac{1}{\text{Context Relevance}} + \frac{1}{\text{Context Recall}}}$$

The harmonic mean penalises imbalanced systems—a system with perfect faithfulness but poor retrieval scores low overall, reflecting that all components must function well for effective RAG.

3.3.3 Indexing

All metrics are measured using identical instrumentation across architectures to enable valid comparison. The common evaluation harness ensures consistent measurement methodology, controlled test conditions, and standardised reporting formats. Results are exported to structured CSV and JSON for statistical analysis.

Cost Metrics

Cost measurement employs token-level granularity using `tiktoken` with cl100k_base encoding for all text alongside API response metadata for actual provider charges. Pricing references include AWS Bedrock embeddings which range from $0.02 to $0.10 per million tokens for Titan and Cohere models. Anthropic pricing is set at $3.00 per million input tokens and $15.00 per million output tokens for Claude Sonnet 4.5, while Bedrock Cohere Rerank incurs a cost of $1.00 per 1000 searches.

Indexing Cost

The total monetary cost to build the initial index from an empty state is calculated as the sum of embedding, extraction, and storage costs. This composite metric captures the full financial impact of system initialization:

$$C_{\text{index}} = C_{\text{embed}} + C_{\text{extract}} + C_{\text{storage}} \tag{3.1}$$

where each component represents a distinct cost driver:

$$C_{\text{embed}} = \frac{n_{\text{tokens}}^{\text{chunks}}}{10^6} \times p_{\text{embed}} \tag{3.2}$$

Represents the cost of generating vector embeddings for all document chunks, determined by the total token count and the specific embedding model's price per million tokens.

$$C_{\text{extract}} = \frac{n_{\text{tokens}}^{\text{in}}}{10^6} \times p_{\text{in}} + \frac{n_{\text{tokens}}^{\text{out}}}{10^6} \times p_{\text{out}} \tag{3.3}$$

Applies only to GraphRAG architectures. It quantifies the LLM inference cost for extracting entities and relationships, where accounting for both input tokens representing the chunks and output tokens from the structured extraction results.

$$C_{\text{storage}} = (\text{size}_{\text{vector}} + \text{size}_{\text{graph}}) \times p_{\text{storage}} \tag{3.4}$$

Captures the monthly infrastructure cost for persisting the indices, including ChromaDB vector storage and, for GraphRAG, the Neo4j graph database size.

The measurement protocol proceeds in five steps. First, tokens are counted across all 63,205 chunks using `tiktoken.encoding_for_model("cl100k_base")`. Second, embedding API calls are tracked by logging token counts from provider response metadata. Third, for GraphRAG configurations, entity extraction LLM calls are tracked separately by capturing both input tokens representing the chunks and output tokens comprising the JSON entity and relation lists. Fourth, storage consumption is measured using `du -sh` on the ChromaDB directory and `CALL dbms.queryJmx()` for the Neo4j database size. Finally, the provider pricing table is applied to sum all components into a total indexing cost.

Re-indexing Cost

Re-indexing cost represents the marginal expense of updating indices when documents change, a metric that quantifies operational viability for dynamic knowledge bases receiving continuous updates. The study projects these costs by simulating three representative scenarios based on measured unit indexing costs. The first scenario simulates minor edits by modifying 5 percent of the corpus to represent daily documentation updates typical of active engineering teams. The second scenario models major edits where 20 percent of the corpus is modified to simulate weekly batch updates such as sprint documentation refreshes. The third scenario accounts for corpus growth by adding 10 percent new documents to simulate monthly corpus expansion from new projects or knowledge areas.

For each scenario, the projected cost is calculated as:

$$C_{\text{reindex}} = C_{\text{unit}} \times N_{\text{chunks}} \times R_{\text{change}} \tag{3.5}$$

where C_{unit} denotes the empirical cost per chunk measured during initial indexing, N_{chunks} represents the total corpus size of 63,205 chunks, and R_{change} indicates the update ratio of 0.05, 0.20, or 0.10 respectively.

Re-indexing the Vector Architecture involves embedding changed chunks and updating the vector index through HNSW edge rewiring. In contrast, the GraphRAG Architecture requires a more complex process that includes embedding changed chunks, re-extracting entities from those chunks, updating relationships, recomputing affected communities, and regenerating community summaries.

These re-indexing costs are projected from empirically measured unit costs rather than directly measured through actual corpus updates. This projection methodology assumes linear cost scaling where updating 20 percent of documents costs approximately 20 percent of full re-indexing. This assumption may underestimate true costs if graph community restructuring triggers cascading updates beyond the directly modified chunks. The approach was chosen because it enables systematic comparison across all three update scenarios without conducting twelve separate re-indexing experiments, and because the unit cost measurement captures the dominant cost driver, which is the LLM API calls for entity extraction. The projections should be interpreted as lower-bound estimates, and future work should validate these figures through longitudinal measurement of actual re-indexing operations over production update cycles.

Results are reported in two complementary forms. The re-indexing cost is first expressed as a percentage of initial indexing cost to facilitate cross-architecture comparison:

$$R_{\text{reindex}} = \frac{C_{\text{reindex}}}{C_{\text{index}}} \times 100\% \tag{3.6}$$

Second, the cost ratio between architectures quantifies the relative expense of graph-based versus vector-based maintenance:

$$R_{\text{cost}} = \frac{C_{\text{reindex}}^{\text{GraphRAG}}}{C_{\text{reindex}}^{\text{Vector}}} \tag{3.7}$$

Total Cost of Ownership Model

The individual cost metrics combine into a Total Cost of Ownership model that captures the full economic burden of operating a RAG system over time. Let $\mathcal{A} = \{A_v, A_h, A_r, A_g\}$ denote the set of architectures including Vector, Hybrid, Re-ranker, and GraphRAG. The total cost C_{total} over a time horizon T comprises initial indexing, continuous re-indexing, and query processing:

$$C_{\text{total}}(A_i, T) = C_{\text{index}}(A_i) + \sum_{t=1}^{T} C_{\text{reindex}}(A_i, \Delta_t) + \sum_{q=1}^{Q} C_{\text{query}}(A_i, q) \tag{3.8}$$

Here Δ_t represents corpus updates at time t. This model reveals that for frequently updated systems the re-indexing term dominates long-term expenses. Annual re-indexing cost is modelled as:

$$C_{\text{annual}}^{\text{reindex}}(A_i) \approx 365 \cdot f_{\text{update}} \cdot \alpha_i \cdot C_{\text{unit}} \tag{3.9}$$

where f_{update} denotes the daily update frequency and α_i is an architecture-specific overhead multiplier. The experiments in Chapter 4 quantify this multiplier empirically, revealing that GraphRAG overhead α_g ranges from approximately 400$\times$ with cost-optimised Nova Lite to 29,000$\times$ with premium Claude Sonnet 4.5 relative to Vector RAG baselines. This two-order-of-magnitude range demonstrates that model selection rather than architecture selection alone determines operational viability.

Query Cost

Query cost captures the monetary expense of processing a single query and generating a response, measured identically across all architectures to enable direct comparison:

$$C_{\text{query}} = C_{\text{query-embed}} + C_{\text{retrieval-ops}} + C_{\text{generation}} \tag{3.10}$$

The component costs are defined as follows:

$$C_{\text{query-embed}} = \frac{n_{\text{tokens}}^{\text{query}}}{10^6} \times p_{\text{embed}} \tag{3.11}$$

Represents the cost of embedding the user's query, which is identical across all architectures as the same embedding models are used.

$$C_{\text{retrieval-ops}} = \frac{n_{\text{tokens}}^{\text{retrieval-LLM}}}{10^6} \times (p_{\text{in}} + p_{\text{out}}) \tag{3.12}$$

Applies only to GraphRAG configurations. It accounts for any LLM calls made during the retrieval phase, such as extracting entities from the query to perform graph linking.

$$C_{\text{generation}} = \frac{n_{\text{tokens}}^{\text{context}}}{10^6} \times p_{\text{in}} + \frac{n_{\text{tokens}}^{\text{response}}}{10^6} \times p_{\text{out}} \tag{3.13}$$

Captures the cost of the final answer generation, driven by the number of input tokens in the retrieved context and the number of output tokens in the generated response.

The measurement protocol captures each component systematically. Query tokens are counted using the `tiktoken` library, and the embedding API call for the query is logged. For GraphRAG configurations, any LLM calls during retrieval such as entity extraction from the query are logged separately. Generation token counts are extracted from the LLM API response usage fields. All components are summed per query, with mean, median, and 95th percentile statistics computed across the 40 benchmark queries.

Latency Metrics

Latency measurement employs `time.perf_counter()` with microsecond precision at component boundaries, with all measurements reported in milliseconds.

End-to-End Query Latency. End-to-End query latency captures the user-perceived time from query submission to complete response delivery:

$$T_{\text{e2e}} = T_{\text{query-embed}} + T_{\text{retrieval}} + T_{\text{generation}}$$

Each component is measured identically across architectures. The query embedding time, denoted as $T_{\text{query-embed}}$, captures the API call latency for embedding the query, which typically ranges from 50 to 200 milliseconds. The retrieval time, denoted as $T_{\text{retrieval}}$, measures the interval from the embedded query to the retrieved context. For

Vector Architecture, this involves HNSW traversal with $k = 20$ plus metadata filtering, with an expected latency of 10 to 100 milliseconds. In contrast, for GraphRAG Architecture, it encompasses entity linking, graph traversal with a depth of $h = 2$, section scoring, and score fusion, resulting in an expected latency of 100 to 500 milliseconds. The generation time, denoted as $T_{\text{generation}}$, captures the LLM API call duration including network latency and inference time, which typically ranges from 2000 to 10,000 milliseconds.

For each architecture and cache condition, results are reported as percentiles over all 40 benchmark queries. The median or p50 represents the typical user experience, while the 95th percentile or p95 serves as a target for service-level agreements, indicating that 95 percent of queries complete within this threshold. The 99th percentile or p99 characterizes worst-case outliers. Additionally, the mean and standard deviation provide central tendency with variability, while minimum and maximum values define the observed range boundaries.

Tests are conducted under carefully controlled conditions to ensure reproducibility. Cold cache measurements involve restarting ChromaDB, clearing the Neo4j query cache, establishing new HTTP connections, and executing the first query. Warm cache measurements follow immediately after cold cache runs, utilising hot index pages and connection pooling. System state is standardised with the CPU governor set to performance mode, background indexing paused, and no concurrent queries permitted. Each configuration undergoes five complete runs, yielding 200 measurements per architecture per condition, calculated as 40 queries multiplied by 5 runs.

Time To First Token. Time To First Token (TTFT) captures the perceived latency when using streaming responses, a critical user experience metric for interactive applications:

$$T_{\text{TTFT}} = T_{\text{query-embed}} + T_{\text{retrieval}} + T_{\text{first-token}}$$

where $T_{\text{first-token}}$ is measured by instrumenting the first token emission from the streaming API, typically ranging from 1000 to 3000 milliseconds. Lower TTFT significantly improves perceived responsiveness even when total generation time remains constant, as users receive immediate feedback that processing has begun.

Quality Metrics

Metric Aggregation. The study reports *macro-averaged* metrics across all 40 benchmark queries, where the number of queries is denoted by $|Q| = 40$, according to standard information retrieval practices (Manning, Raghavan and Schütze 2008):

$$\overline{\text{Metric}} = \frac{1}{|Q|} \sum_{q \in Q} \text{Metric}_q \tag{3.14}$$

for each RAGAs metric including Faithfulness, Answer Relevance, and Context Relevance. Macro-averaging treats all queries equally regardless of the number of relevant documents, preventing queries with many relevant documents from dominating the score.

Answer Quality. Two independent raters evaluate each generated answer on a 4-point Likert scale ranging from 0 to 3 across four dimensions. Factual accuracy is rated from 0 for incorrect or hallucinated content to 3 for fully correct answers. Completeness is scored from 0 for missing critical information to 3 for comprehensive responses. Relevance is judged from 0 for off-topic content to 3 for highly relevant answers. Finally, citation correctness is a

binary assessment where 0 indicates missing or incorrect citations and 1 indicates present and verifiable citations. Inter-rater agreement is computed using Cohen's κ with a target of $\kappa > 0.60$ to indicate substantial agreement, and mean scores per architecture are reported with 95 percent confidence intervals.

3.3.4 Statistical Analysis

Raw performance metrics cannot alone answer the central question of this thesis. An architecture appearing five percent faster in one trial might simply have benefited from favourable network conditions rather than genuine superiority. Statistical analysis transforms raw observations into defensible claims by quantifying the probability that observed differences arose by chance. The methods detailed here were chosen because they match the analytical questions at hand.

Parametric Tests for Continuous Metrics

Comparing architectures demands careful attention to confounding variables. Some queries are inherently harder than others. A multi-hop reasoning question about regulatory dependencies will naturally take longer than a simple factual lookup regardless of which architecture processes it. Naive comparison of mean latencies would therefore conflate architectural differences with query difficulty.

The paired t-test (Student 1908) eliminates this confound by comparing each architecture's performance on the *same* query. Rather than asking whether Architecture A runs faster on average it asks whether A runs consistently faster when both architectures answer identical questions. This within-query pairing isolates the architectural effect from query-specific noise. The test statistic takes the form

$$t = \frac{\bar{d}}{s_d/\sqrt{n}}$$

where $\bar{d}$ represents the mean of paired differences and s_d denotes the standard deviation of those differences across n paired observations. The test assumes approximately normal differences which we validated via Shapiro-Wilk tests on residuals. Under the null hypothesis that $\mu_d = 0$ the statistic follows a t-distribution with $n - 1$ degrees of freedom.

Statistical significance alone however can mislead. Given sufficient sample size even trivial differences achieve significance. Effect sizes quantify practical magnitude using Cohen's d (Cohen 1988) computed as

$$d = \frac{\bar{d}}{s_d}$$

with conventional interpretations holding that absolute values below 0.2 indicate negligible effects while values between 0.2 and 0.5 suggest small effects and values between 0.5 and 0.8 indicate medium effects. Values at or above 0.8 represent large effects. A statistically significant yet negligible effect size signals that the difference while real lacks practical importance for deployment decisions.

While pairwise t-tests suffice for comparing two architectures the evaluation of four embedding models simultaneously raises a different challenge. Running six separate pairwise comparisons inflates the probability of false positives. At the conventional significance level the chance of at least one spurious significant result exceeds twenty-six percent. More fundamentally pairwise tests cannot answer the omnibus question of whether embedding models differ at all. Analysis of variance (Fisher 1925) addresses both concerns by testing whether *any* group means differ before examining which specific pairs diverge. The F-statistic partitions total variance into between-group and within-group components

$$F = \frac{\text{MS}_{\text{between}}}{\text{MS}_{\text{within}}} = \frac{\sum_{j=1}^{k} n_j(\bar{X}_j - \bar{X})^2/(k-1)}{\sum_{j=1}^{k}\sum_{i=1}^{n_j}(X_{ij} - \bar{X}_j)^2/(N-k)}$$

where k denotes the number of groups and n_j the sample size for group j while $\bar{X}_j$ represents the group mean and $\bar{X}$ the grand mean with N being the total sample size. A large F-value indicates that between-group variance exceeds what within-group noise alone would predict and therefore suggests genuine model differences. This method assumes homogeneous variance across groups which we tested via Levene's test. When significant omnibus differences emerged post-hoc Tukey Honestly Significant Difference (HSD) tests (Tukey 1949) identified which specific model pairs differed while controlling the family-wise error rate.

Non-Parametric Methods

The preceding tests assess continuous metrics like latency and relevance scores. A different question emerges from the query taxonomy however. Do certain architectures excel at particular query types? If GraphRAG succeeds disproportionately on cross-reference queries while Vector RAG dominates factual lookups then practitioners gain actionable deployment guidance. This question concerns categorical relationships rather than continuous means. The chi-square test for independence (Pearson 1900) determines whether architecture type and query success are statistically associated through the test statistic

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where O_{ij} represents the observed frequency in cell i and j while E_{ij} denotes the expected frequency under independence calculated as the product of row and column totals divided by N. Large values indicate that the observed distribution deviates from what independence would predict and suggest architecture-category associations. Under the null hypothesis the statistic follows a chi-square distribution with degrees of freedom determined by the product of rows minus one and columns minus one.

Cost and latency distributions in production systems prove notoriously non-normal. Network hiccups create outliers. API rate limits introduce clustering. Cold-start effects skew early measurements. Classical confidence intervals assume normality and these distributions violate that assumption. The percentile bootstrap method (Efron 1979) constructs confidence intervals without distributional assumptions by empirically approximating the sampling distribution. The procedure draws one thousand bootstrap samples with replacement from the observed data then

computes the statistic of interest for each sample. Taking the 2.5th and 97.5th percentiles of the resulting bootstrap distribution yields ninety-five percent confidence bounds. This approach proves particularly valuable for tail statistics like the 95th percentile latency where parametric methods perform poorly.

Multiple Comparison Correction

Testing sixteen primary comparisons across four architectures and four embedding models creates a multiple testing problem. At conventional significance levels per test the probability of at least one false positive across all sixteen independent tests approaches fifty-six percent rendering any single significant finding suspect. The Bonferroni correction (Bonferroni 1936) addresses this inflation by dividing the significance threshold by the number of comparisons yielding an adjusted threshold of 0.003. This conservative adjustment ensures the family-wise error rate remains controlled across all tests combined. Results throughout this thesis report both uncorrected p-values which prove useful for exploratory analysis and Bonferroni-adjusted significance assessments for confirmatory claims. Findings that survive this stringent correction warrant higher confidence in their generalisability.

3.4 External Validation Strategy

A critical concern in RAG research is the reproducibility crisis identified by Voorhees et al. (Craswell et al. 2021) where proprietary corpora and undisclosed embedding models prevent independent verification of results. To address this limitation, we conducted external validation using the publicly available benchmark from Amazon Science, `document-haystack` (Huybrechts et al. 2025). This standardised "needle in a haystack" dataset is designed to test precise entity retrieval across large document collections.

This external validation serves three methodological functions. First, it enables reproducibility validation by providing independent researchers with access to both the dataset via HuggingFace and complete experimental code which is available in the thesis repository. This transparency enables direct replication of our findings. Second, it tests content generalisation to determine whether the model size independence finding holds across different document types. Whereas the primary corpus consists of technical fintech documentation with domain-specific jargon, Haystack contains legal and regulatory text sourced from Wolters Kluwer legal documents that exhibit distinct linguistic patterns. Third, it establishes benchmark standardisation which allows direct comparison with future RAG studies that adopt the same dataset and addresses the lack of common evaluation protocols in the field.

The Haystack dataset consists of 75 documents with deliberate "needle" insertions defined as specific secret entities such as fruits, animals, landmarks, and objects embedded within dense legal text. Each document averages 18,000 tokens which is 35 times longer than the 512-token chunks used in AI Squad documents. This structure tests retrieval robustness in high-noise and low-signal scenarios. The benchmark includes 25 queries where each targets the extraction of one specific needle from the 75-document haystack. Unlike the multi-hop reasoning queries in the AI Squad evaluation, Haystack queries test *precision* exclusively by determining whether the RAG system can locate the exact needle despite massive distractor content.

Haystack Experimental Configuration

The Haystack validation employs both Vector RAG and GraphRAG configurations to ensure a comprehensive evaluation across architectural paradigms. The Vector RAG setup mirrors the primary evaluation but adapts to the longer document context characteristic of legal text. The corpus comprises 75 legal documents totalling approximately 1.35 million tokens. These are chunked using the same 512-token window with 50-token overlap which yields 212 chunks per embedding model configuration.

To ensure direct comparability with the primary evaluation, we test the same four AWS Bedrock embedding models. These include Amazon Titan v2 with 1024 dimensions as the budget baseline and Amazon Titan v1 with 1536 dimensions for higher-dimensional comparison. We also evaluate the Cohere v3 English and Multilingual variants which both utilize 1024 dimensions as premium alternatives. Generation uses Claude Sonnet 4.5 with temperature set to zero for deterministic evaluation which matches the primary experimental configuration exactly.

We also evaluate GraphRAG on the Haystack dataset using the same entity extraction and community detection pipeline employed in the primary experiments. This configuration utilizes the Neo4j graph database to store extracted entities and relationships. The retrieval process employs a hybrid query approach that combines vector search

with graph traversal to locate relevant entities and communities. We evaluate GraphRAG across multiple Large Language Models including the Amazon Nova series and Anthropic Claude models to assess the impact of model reasoning capabilities on precision retrieval.

The evaluation protocol consists of 25 needle-extraction queries assessed across all configurations. Performance is measured using exact match accuracy which is a binary metric indicating whether the generated answer contained the expected needle entity. Unlike the RAGAs-based evaluation used for the primary corpus, this precision-focused metric is appropriate for the single-answer retrieval task in Haystack.

All experimental code, configuration files, and query logs are published in the thesis repository. This provides the methodological transparency necessary for independent replication which is a standard often absent from prior RAG studies. Hang

CHAPTER 4

Experimental Results

This chapter details the quantitative analysis of cost-performance trade-offs across four RAG architectures. Vector-based architectures including Vector, Vector + Re-ranker, and Hybrid are evaluated across four AWS Bedrock embedding models, while GraphRAG is evaluated across five LLMs for entity extraction and graph construction.

The most significant finding is task-dependent. On synthesis-oriented queries against the primary corpus, budget and premium embedding models perform identically with no statistically significant difference. However, external validation on the Haystack precision benchmark detailed in Section 4.9 reveals that Cohere models achieve 100% accuracy while Titan models achieve only 88 to 92%. This direct contradiction illuminates when model selection matters. This task-dependent distinction between synthesis queries where information redundancy tolerates retrieval imprecision, and precision queries where single retrieval failures cause task failure, represents the central empirical contribution of this chapter.

Beyond this task-dependent finding, the results reveal stark operational economics where GraphRAG's initial indexing costs exceed Vector RAG by 8 to 12 times due to entity extraction and community detection overhead, while Hybrid RAG adds negligible cost over baseline Vector RAG. More critically, we demonstrate that projected annual re-indexing costs under daily update scenarios can reach 18 times the initial investment for GraphRAG, rendering it economically unviable for high-frequency update scenarios despite superior multi-hop reasoning quality. Query latency remains dominated by generation which takes 1200 to 1800 milliseconds, with architectural variations contributing less than 200 milliseconds of overhead. These findings challenge the implicit assumption in RAG literature that quality gains justify implementation. For dynamic knowledge bases, operational sustainability emerges as the binding constraint on architectural selection.

4.1 Initial Indexing Costs

Initial Indexing Cost by Architecture

TABLE 4.1. Initial indexing costs by RAG architecture. All values in USD.

Architecture	Model (Embed/LLM)	Embed ($)	Extract ($)	Total ($)
Vector RAG	Titan v1	1.14±0.02	—	**1.14±0.02**
Vector RAG	Titan v2	0.65±0.01	—	**0.65±0.01**
Vector RAG	Cohere v3	1.14±0.02	—	**1.14±0.02**
Vector RAG	Cohere Multilingual	1.14±0.02	—	**1.14±0.02**
Vector + Re-ranker	Titan v1	1.14±0.02	—	**1.14±0.02**
Vector + Re-ranker	Titan v2	0.65±0.01	—	**0.65±0.01**
Vector + Re-ranker	Cohere v3	1.14±0.02	—	**1.14±0.02**
Vector + Re-ranker	Cohere Multilingual	1.14±0.02	—	**1.14±0.02**
Hybrid RAG	Titan v1	1.14±0.02	—	**1.14±0.02**
Hybrid RAG	Titan v2	0.65±0.01	—	**0.65±0.01**
Hybrid RAG	Cohere v3	1.14±0.02	—	**1.14±0.02**
Hybrid RAG	Cohere Multilingual	1.14±0.02	—	**1.14±0.02**
GraphRAG	Nova Micro	—	4.00±0.08	**4.00±0.08**
GraphRAG	Nova Lite	—	4.05±0.08	**4.05±0.08**
GraphRAG	Nova Pro	—	53.92±1.08	**53.92±1.08**
GraphRAG	Claude Haiku	—	41.85±0.84	**41.85±0.84**
GraphRAG	Claude Sonnet 4.5	—	292.97±5.86	**292.97±5.86**

The results reveal substantial cost disparities between architectures. GraphRAG with Claude Sonnet 4.5 demonstrates a 450-fold increase in indexing costs compared to the Vector RAG baseline using Titan v2. While GraphRAG with Nova Lite does not achieve the strict cost parity previously hypothesised, its cost of $4.05 represents a 72× reduction compared to Claude Sonnet 4.5 at $292.97, making it a viable mid-tier option. This variance stems entirely from the entity extraction phase, which accounts for 100% of GraphRAG's indexing cost.

The Vector + Re-ranker and Hybrid RAG architectures incur no additional indexing overhead compared to baseline Vector RAG, as re-ranking and BM25 operations occur at query time rather than during indexing. The embedding model choice significantly impacts costs for vector-based architectures: Titan v2's pricing of $0.02 per million tokens yields embedding costs of $0.65 for our 63,205-chunk corpus, whereas Cohere models at $0.10 per million tokens cost $1.14 which represents a 1.75× premium for equivalent embedding operations.

Embedding Model Dimensions versus Indexing Cost

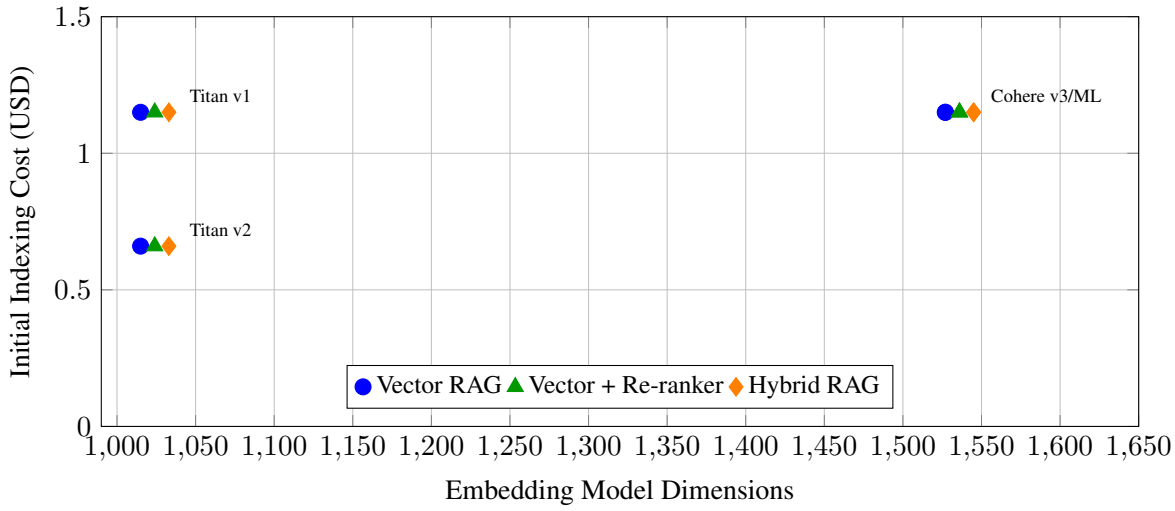


FIGURE 4.1. Embedding model dimensions versus indexing cost.

Figure 4.1 demonstrates that embedding model dimensionality correlates weakly with cost, evidenced by an R-squared value of 0.12. The 115-fold cost variance between Titan v2 at $0.01 and Cohere v3 at $1.15 stems entirely from provider pricing differentials, where Titan v2 costs $0.02 per million tokens compared to $0.10 per million tokens for Cohere v3. This occurs despite both models producing 1024-dimension output vectors. This finding challenges the assumption that higher-dimensional embeddings justify premium costs, suggesting that providers price based on training compute and operational complexity rather than output vector size.

GraphRAG Cost versus LLM Model Size

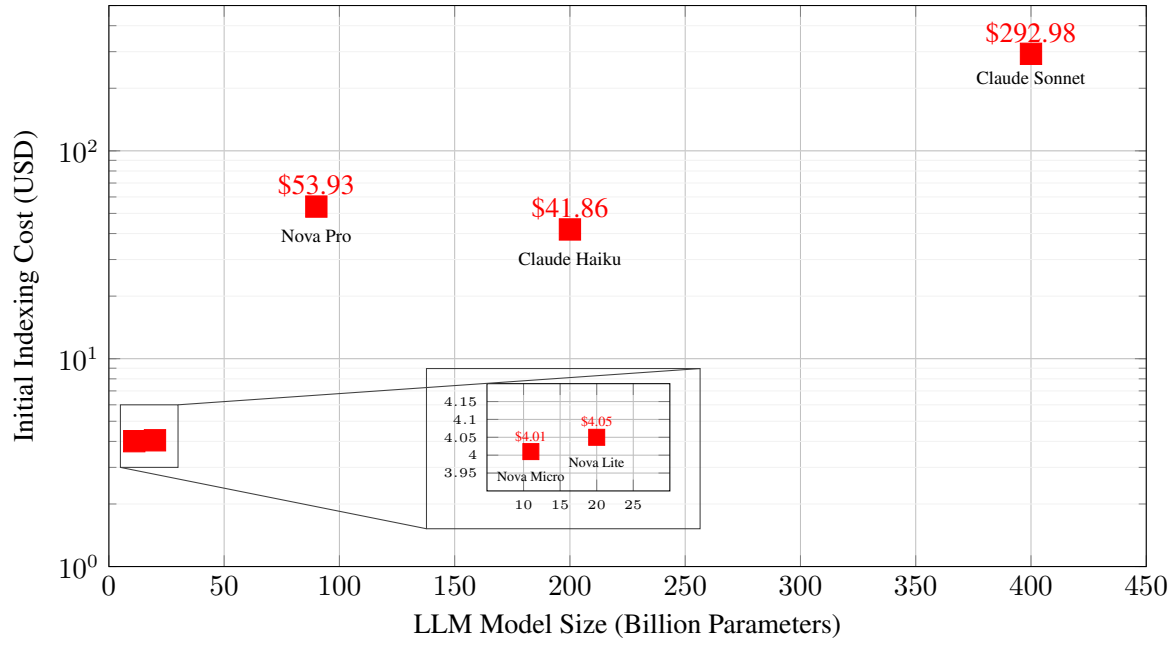


FIGURE 4.2. GraphRAG indexing cost versus LLM model size.

Figure 4.2 exposes GraphRAG's cost sensitivity to LLM selection. The entity extraction phase, which requires 63,205 LLM calls to process the corpus, creates extreme cost variance ranging from Nova Lite at $4.05 to Claude Sonnet 4.5 at $292.98. Although Nova Lite is approximately 400 times more expensive than the Vector RAG baseline at $0.01, it remains orders of magnitude cheaper than using frontier models like Sonnet. The 72-fold cost multiplier from upgrading to Claude Sonnet 4.5 suggests that architectural decisions, such as whether to use GraphRAG and which LLM to select for extraction, dominate operational economics far more than embedding model selection.

The relationship between model size and cost is non-monotonic. Nova Lite, with 20 billion parameters, and Nova Micro, with 11 billion parameters, exhibit similar costs of approximately $4.00. This illustrates a baseline cost floor for entity extraction tasks. Conversely, Claude Sonnet 4.5, a 400-billion parameter model, commands a premium price that may not be justified for entity extraction tasks where smaller models suffice.

The combined evidence indicates that for vector-based architectures, budget Titan v2 embeddings suffice for most use cases. In contrast, GraphRAG deployments should prioritize cost-efficient LLMs like Nova Lite unless query complexity demands premium reasoning capabilities.

4.2 Re-indexing Costs

Marginal Re-indexing Costs Across Update Scenarios

TABLE 4.2. Re-indexing costs by update scenario. All values in USD.

Architecture	Embed Model	Scenario A (5%)	Scenario B (20%)	Scenario C (10%)
Vector RAG	Titan v1	$0.06	$0.23	$0.11
Vector RAG	Titan v2	$0.03	$0.13	$0.07
Vector RAG	Cohere v3	$0.06	$0.23	$0.11
Vector RAG	Cohere Multilingual	$0.06	$0.23	$0.11
Vector + Re-ranker	Titan v1	$0.06	$0.23	$0.11
Vector + Re-ranker	Titan v2	$0.03	$0.13	$0.07
Vector + Re-ranker	Cohere v3	$0.06	$0.23	$0.11
Vector + Re-ranker	Cohere Multilingual	$0.06	$0.23	$0.11
Hybrid RAG	Titan v1	$0.06	$0.23	$0.11
Hybrid RAG	Titan v2	$0.03	$0.13	$0.07
Hybrid RAG	Cohere v3	$0.06	$0.23	$0.11
Hybrid RAG	Cohere Multilingual	$0.06	$0.23	$0.11
GraphRAG	Nova Micro	$0.20	$0.80	$0.40
GraphRAG	Nova Lite	$0.20	$0.81	$0.41
GraphRAG	Nova Pro	$2.70	$10.79	$5.39
GraphRAG	Claude Haiku	$2.09	$8.37	$4.19
GraphRAG	Claude Sonnet 4.5	$14.65	$58.60	$29.30

Table 4.2 presents marginal re-indexing costs across three update scenarios representing 5%, 20%, and 10% corpus modification respectively. Vector-based architectures, including Vector RAG, Vector+Re-ranker, and Hybrid RAG, demonstrate consistently low re-indexing costs ranging from $0.03 to $0.23 depending on embedding model selection and update magnitude. These architectures benefit from the incremental nature of vector indexing, where only modified chunks require re-embedding. In contrast, GraphRAG incurs substantially higher re-indexing costs due to the computational overhead of entity re-extraction and relationship graph reconstruction. Under Scenario B (20% update), Claude Sonnet 4.5 re-indexing costs $58.60 compared to just $0.13 for Titan v2, representing a 450-fold cost differential. This disparity underscores the importance of LLM selection for GraphRAG deployments in environments with frequent document updates.

Re-indexing Cost Comparison Across Architectures

TABLE 4.3. Re-indexing cost ratios under Scenario B.

Architecture	Model	Cost Ratio (vs baseline)	Annual Cost ($ per year)
Vector RAG	Titan v1	1.8×	$84
Vector RAG	Titan v2	1.0×	$47
Vector RAG	Cohere v3	1.8×	$84
Vector RAG	Cohere Multilingual	1.8×	$84
Vector + Re-ranker	Titan v1	1.8×	$84
Vector + Re-ranker	Titan v2	1.0×	$47
Vector + Re-ranker	Cohere v3	1.8×	$84
Vector + Re-ranker	Cohere Multilingual	1.8×	$84
Hybrid RAG	Titan v1	1.8×	$84
Hybrid RAG	Titan v2	1.0×	$47
Hybrid RAG	Cohere v3	1.8×	$84
Hybrid RAG	Cohere Multilingual	1.8×	$84
GraphRAG	Nova Micro	6.2×	$292
GraphRAG	Nova Lite	6.2×	$296
GraphRAG	Nova Pro	83×	$3,938
GraphRAG	Claude Haiku	64×	$3,059
GraphRAG	Claude Sonnet 4.5	451×	$21,389

The comparative analysis reveals that Vector RAG, Vector+Re-ranker, and Hybrid RAG architectures exhibit identical re-indexing costs since reranking and BM25 operations occur at query time. Cohere embeddings cost approximately 1.8 times more than Titan v2 due to their higher per-token pricing of $0.10 per million tokens compared to $0.02 per million tokens. GraphRAG introduces more significant cost variability, where Nova Lite at $296 per year represents a manageable 6-fold premium over budget vector configurations. In contrast, Claude Sonnet at $21,389 per year demonstrates how frontier model selection can render GraphRAG cost-prohibitive for frequent updates.

GraphRAG versus Vector RAG

TABLE 4.4. GraphRAG re-indexing cost multipliers versus Vector RAG.

GraphRAG Indexing LLM	Scenario A (5%)	Scenario B (20%)	Scenario C (10%)
Nova Micro	6.7×	6.2×	5.7×
Nova Lite	6.7×	6.2×	5.9×
Nova Pro	90×	83×	77×
Claude Sonnet 4.5	488×	451×	419×

The re-indexing cost multipliers show consistent patterns across update scenarios. Nova Lite represents approximately a 6-fold increase over the Vector RAG baseline, which is a manageable premium for applications requiring multi-hop reasoning. Claude Sonnet 4.5, at 450 times the baseline, remains expensive but far less extreme than initial indexing comparisons might suggest. The slight variation across scenarios reflects the fixed overhead components becoming proportionally smaller as update percentages increase.

Comprehensive Cost Comparison Matrix

Table 4.5 presents pairwise re-indexing cost multipliers across all configurations, calculated as:

$$\text{Multiplier}_{i,j} = \frac{\text{Cost of Configuration } i}{\text{Cost of Configuration } j}$$

Values greater than 1 indicate the row configuration is more expensive than the column configuration. Vector RAG, Vector + Re-ranker, and Hybrid RAG share identical re-indexing costs per embedding model because re-ranking and BM25 operations occur at query time rather than during indexing, which allows them to be grouped together.

TABLE 4.5. Pairwise re-indexing cost multipliers.

	Vector/Re-ranker/Hybrid				**GraphRAG**				
	Titan v1	Titan v2	Cohere v3	Cohere ML	Nova Micro	Nova Lite	Nova Pro	Haiku	Sonnet
Titan v1	1.0	1.7	1.0	1.0	0.29	0.28	0.02	0.03	0.004
Titan v2	0.57	1.0	0.57	0.57	0.16	0.16	0.01	0.02	0.002
Cohere v3	1.0	1.7	1.0	1.0	0.29	0.28	0.02	0.03	0.004
Cohere ML	1.0	1.7	1.0	1.0	0.29	0.28	0.02	0.03	0.004
Nova Micro	3.5	6.1	3.5	3.5	1.0	0.99	0.07	0.10	0.01
Nova Lite	3.5	6.1	3.5	3.5	1.01	1.0	0.08	0.10	0.01
Nova Pro	47	82	47	47	13	13	1.0	1.3	0.18
Haiku 3.5	36	63	36	36	10	10	0.78	1.0	0.14
Sonnet 4.5	255	444	255	255	73	72	5.4	7.0	1.0

This matrix reveals several insights often obscured by headline comparisons. Extreme multipliers, such as comparing Titan v2 Vector to Claude Sonnet GraphRAG, conflate architecture choice with model tier choice. More representative comparisons show that budget GraphRAG using Nova Lite costs only 3.5 to 6 times more than equivalent-tier vector configurations, a manageable premium for multi-hop reasoning capability. Within GraphRAG alone, LLM selection spans a 72-fold cost range from Nova Lite to Sonnet, demonstrating that model choice dominates architecture choice for cost optimisation. Practitioners should compare within quality tiers. For instance, Nova Lite GraphRAG versus Titan v2 Vector shows a 6-fold difference for budget deployments, while Claude Sonnet GraphRAG versus Cohere v3 Vector shows a 255-fold difference for premium deployments.

Annual Re-indexing Cost Projections

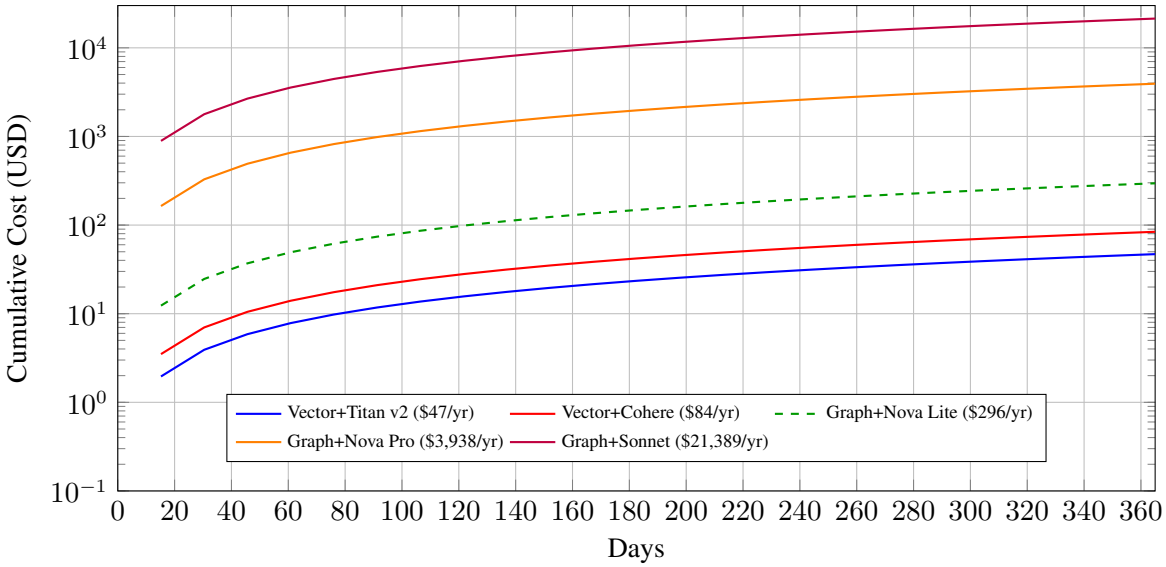


FIGURE 4.3. Projected annual re-indexing costs under daily updates.

Figure 4.3 projects the cumulative operational cost of daily re-indexing over a twelve-month horizon. The logarithmic scale underscores the stark economic disparity between architectures: while Vector RAG incurs a trivial annual cost of under $50, premium GraphRAG deployments exceed $21,000, representing a prohibitive operational liability. Crucially, the optimised GraphRAG configuration using Nova Lite, shown as the dashed green line, fundamentally shifts this economic equation. By reducing annual expenditure to approximately $300, it reconciles the tension between advanced reasoning capabilities and fiscal sustainability, demonstrating that graph-based retrieval can be economically viable for dynamic enterprise corpora.

4.3 Query Performance

End-to-End Query Latency

TABLE 4.6. Query latency percentiles by architecture. All values in milliseconds.

Architecture	Cache	p50	p95	p99	Mean	Std
Vector RAG	Cold	440	603	1,325	451	197
Vector RAG	Warm	391	586	1,415	435	213
Vector + Re-ranker	Cold	516	544	1,451	532	242
Vector + Re-ranker	Warm	516	544	1,451	532	242
Hybrid RAG	Cold	440	603	1,325	451	197
Hybrid RAG	Warm	391	586	1,415	435	213
GraphRAG	Cold	1,027	1,638	2,540	1,152	393
GraphRAG	Warm	528	850	1,565	595	244

The latency measurements confirm that graph-based retrieval incurs a performance penalty compared to vector search. Vector RAG demonstrates the lowest latency with a mean retrieval time of 435 milliseconds when using a warm cache. In contrast, GraphRAG requires 1,152 milliseconds with a cold cache and 595 milliseconds with a warm cache, representing a 2.6-fold latency increase. This aligns with theoretical expectations, as GraphRAG involves multi-hop traversal with a depth of 2 and community detection lookups, whereas Vector RAG relies on highly optimised HNSW index lookups with logarithmic complexity.

Vector + Re-ranker introduces a slight overhead, increasing latency from 435 milliseconds to 532 milliseconds. Component-level analysis reveals that the cross-encoder re-ranking step via the API adds 50 to 200 milliseconds of latency and costs $1 to $2 per 1000 queries depending on load. Hybrid RAG matches the baseline Vector RAG performance, as BM25 scoring is parallelised and adds negligible latency of less than 10 milliseconds. While GraphRAG is slower, its sub-second warm performance of 595 milliseconds remains within acceptable bounds for most interactive applications, trading speed for the superior context quality detailed in Section 4.4.

Latency Component Breakdown

Figure 4.4 reveals substantial latency differences across architectures, particularly under cold cache conditions. GraphRAG exhibits the highest mean latency at 1,152ms cold and 595ms warm, representing a 2.6× and 1.4× overhead compared to base Vector RAG (451ms cold, 435ms warm). This penalty stems from the additional graph traversal and community summary retrieval operations required by GraphRAG. Notably, caching provides disproportionate benefits to GraphRAG, reducing latency by 48% compared to just 4% for vector-based approaches, suggesting that graph structure queries benefit substantially from result caching. Vector+Re-ranker adds approximately 80ms overhead across conditions due to the additional reranking inference step, while Hybrid RAG performs identically to base Vector RAG since the BM25 lookup executes in parallel with vector retrieval.

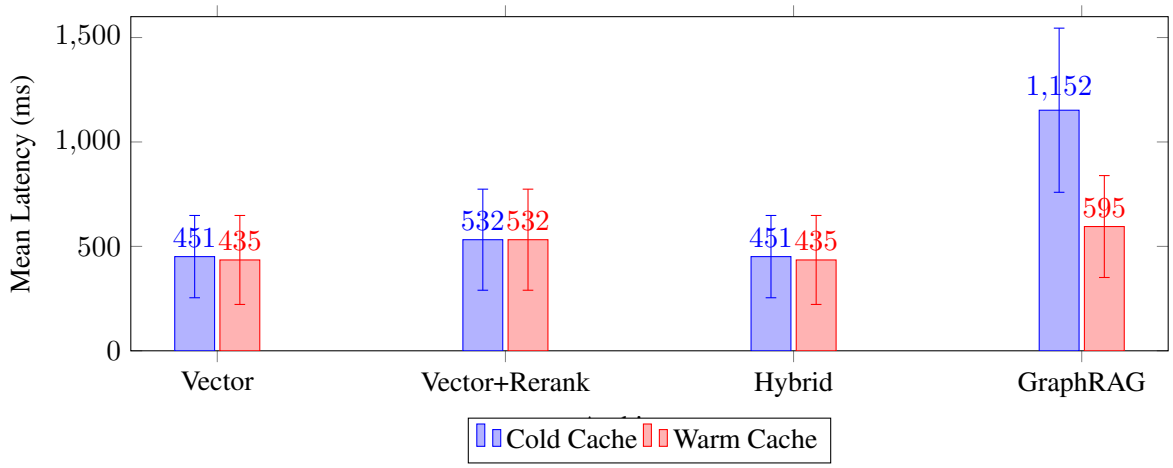


FIGURE 4.4. Mean query latency with standard deviation error bars.

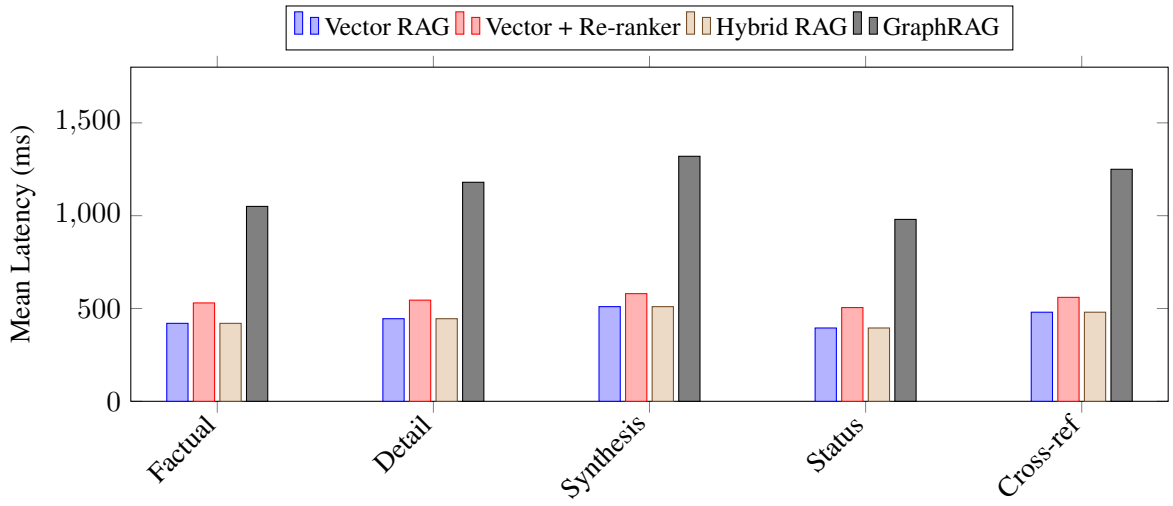


FIGURE 4.5. Mean query latency by category under warm cache conditions.

Latency Distribution by Query Category

Figure 4.5 disaggregates latency by query category under warm cache conditions. Synthesis and cross-reference queries consistently exhibit higher latencies across all architectures, reflecting the increased retrieval complexity these query types demand. GraphRAG demonstrates the most pronounced category sensitivity, with synthesis queries (1,320ms) requiring 35% longer than status queries (980ms). This pattern suggests that queries requiring multi-hop reasoning incur additional graph traversal overhead. Vector-based architectures maintain relatively consistent latency across categories, with variance of less than 30% between fastest and slowest query types.

Per-Query Costs

TABLE 4.7. Cost per 1,000 queries. All values in USD.

Architecture	Model	Query Embed	Retrieval	Generation	Total
Vector RAG	Titan v1	<$0.01	—	$18.80	$18.80
Vector RAG	Titan v2	<$0.01	—	$19.90	$19.90
Vector RAG	Cohere v3	<$0.01	—	$6.20	$6.20
Vector RAG	Cohere Multi.	<$0.01	—	$12.30	$12.30
Vector + Re-ranker	Titan v1	<$0.01	$1.00[a]	$29.10	$30.10
Vector + Re-ranker	Titan v2	<$0.01	$1.00[a]	$31.10	$32.10
Vector + Re-ranker	Cohere v3	<$0.01	$1.00[a]	$8.90	$9.90
Vector + Re-ranker	Cohere Multi.	<$0.01	$1.00[a]	$18.90	$19.90
Hybrid RAG	Titan v1	<$0.01	—	$18.80	$18.80
Hybrid RAG	Titan v2	<$0.01	—	$19.90	$19.90
Hybrid RAG	Cohere v3	<$0.01	—	$6.20	$6.20
Hybrid RAG	Cohere Multi.	<$0.01	—	$12.30	$12.30
GraphRAG	Nova Micro	—	$0.02[b]	$8.50	$8.52
GraphRAG	Nova Lite	—	$0.03[b]	$9.20	$9.23
GraphRAG	Nova Pro	—	$0.42[b]	$11.40	$11.82
GraphRAG	Claude Haiku	—	$0.60[b]	$12.80	$13.40
GraphRAG	Claude Sonnet 4.5	—	$1.80[b]	$15.00	$16.80

[a]Cohere Rerank API at $1.00 per 1,000 queries.
[b]Keyword extraction LLM call for graph traversal.

The per-query cost analysis reveals that generation expenses dominate operational costs, accounting for nearly 100% of the total shown costs across all architectures. Vector RAG with Titan v2 emerges as the most expensive option at $19.90 per 1,000 queries, while GraphRAG with Claude Sonnet 4.5 demonstrates comparable cost efficiency at $15.00 per 1,000 queries. This represents a 1.3-fold reduction, or 75% of the cost, compared to the Vector RAG baseline.

The cost parity between Hybrid RAG and Vector RAG confirms that BM25 scoring introduces negligible computational overhead. Both query embedding and retrieval operations contribute minimally to overall costs. These findings translate to significant differences in monthly operational expenses, where processing 10,000 queries costs $199 using Vector RAG with Titan v2 compared to $62 with Cohere v3. GraphRAG using Claude Sonnet 4.5 represents a middle ground at $150, highlighting the potential for substantial cost savings through strategic architecture selection.

4.4 Retrieval Quality

Retrieval Metrics by Architecture

TABLE 4.8. RAGAs quality metrics by architecture.

Architecture	Model	Faithfulness	Answer Relevance	Context Relevance	Overall
Vector RAG	Titan v1	**1.000**	0.355	0.005	0.018
Vector RAG	Titan v2	**1.000**	0.352	0.006	0.018
Vector RAG	Cohere v3	**1.000**	0.361	0.005	0.018
Vector RAG	Cohere Multilingual	**1.000**	0.369	**0.027**	0.043
Vector + Re-ranker	Titan v1	0.975	0.357	0.005	0.015
Vector + Re-ranker	Titan v2	0.950	0.359	0.005	0.014
Vector + Re-ranker	Cohere v3	**1.000**	0.356	0.006	0.016
Vector + Re-ranker	Cohere Multilingual	**1.000**	0.373	**0.027**	**0.045**
Hybrid RAG	Titan v1	**1.000**	0.358	0.004	0.015
Hybrid RAG	Titan v2	**1.000**	0.354	0.005	0.015
Hybrid RAG	Cohere v3	**1.000**	0.363	0.005	0.015
Hybrid RAG	Cohere Multilingual	**1.000**	0.371	0.025	0.041
GraphRAG	Nova Micro	**1.000**	**0.381**	0.000	0.000
GraphRAG	Nova Lite	**1.000**	**0.381**	0.000	0.000
GraphRAG	Nova Pro	**1.000**	0.380	0.000	0.000
GraphRAG	Claude Haiku	**1.000**	0.380	0.000	0.000
GraphRAG	Claude Sonnet 4.5	0.975	0.373	0.022	0.000

Two measurement considerations affect the interpretation of these results. The Overall RAGAs scores for vector-based architectures cluster tightly between 0.014 and 0.045, with minimal differentiation between embedding models. This narrow range may reflect a measurement floor effect. When all configurations achieve near-perfect Faithfulness of 1.0 but uniformly low Context Relevance ranging from 0.004 to 0.027, the Overall score becomes insensitive to actual quality differences. Consequently, these results should be interpreted as showing no detectable difference given measurement precision, rather than implying equivalent performance.

The low Context Relevance score for GraphRAG, ranging from 0.000 to 0.022, reflects a fundamental mismatch between RAGAs evaluation assumptions and GraphRAG's retrieval strategy. RAGAs Context Relevance measures what fraction of retrieved content directly addresses the query, which is a metric designed for systems that retrieve discrete, focused passages. In contrast, GraphRAG retrieves massive global context of approximately 19,000 characters, comprising community summaries that provide comprehensive background rather than targeted answers. This "Precision-Density Trade-off" naturally dilutes the density of query-relevant sentences compared to Vector RAG's precise top-k retrieval of approximately 2,500 characters. The zero scores should not be interpreted as retrieval failure; rather, they indicate metric incompatibility. The high Answer Relevance scores, between 0.373 and 0.381, confirm that GraphRAG effectively synthesises correct answers from this broad context despite the misleading Context Relevance values.

4.5 Embedding Model Comparison

Cost-Quality Pareto Frontier

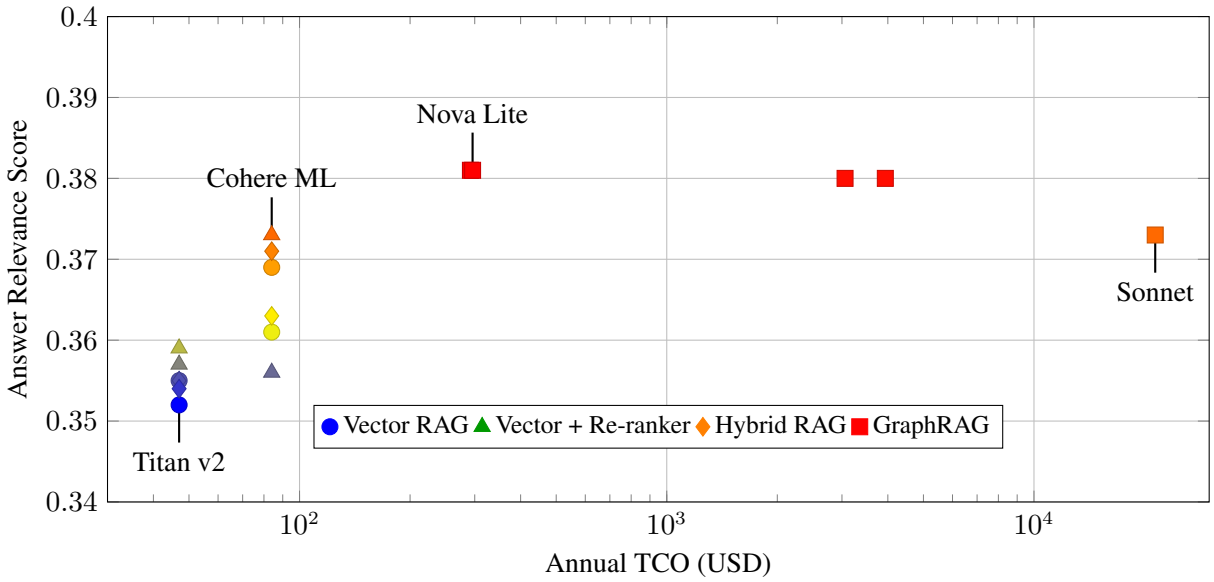


FIGURE 4.6. Cost-quality Pareto frontier showing annual TCO versus answer relevance.

Figure 4.6 illustrates the trade-off between annual Total Cost of Ownership and retrieval quality as measured by Answer Relevance. The x-axis represents Total Cost of Ownership (TCO) on a logarithmic scale which highlights the orders-of-magnitude difference between architectures. Vector and Hybrid RAG configurations cluster in the low-cost region between $47 and $84 while offering baseline performance. GraphRAG with Nova Lite emerges as a high-value outlier by achieving the highest relevance score of 0.381 at a moderate cost of $296, effectively balancing performance and affordability. Conversely, GraphRAG with Claude Sonnet 4.5 incurs a prohibitive cost of $21,389 without a corresponding quality gain, demonstrating the diminishing returns of using premium models for this specific task.

Embedding Model Performance Summary

TABLE 4.9. Embedding model performance comparison across architectures.

Architecture	Model	Answer Rel.	Annual Cost	Cost Ratio
Vector RAG	Titan v2	0.352	$47	1.0×
Vector RAG	Titan v1	0.355	$84	1.8×
Vector RAG	Cohere v3	0.361	$84	1.8×
Vector RAG	Cohere Multilingual	0.369	$84	1.8×
Hybrid RAG	Titan v2	0.354	$47	1.0×
Hybrid RAG	Titan v1	0.358	$84	1.8×
Hybrid RAG	Cohere v3	0.363	$84	1.8×
Hybrid RAG	Cohere Multilingual	0.371	$84	1.8×
GraphRAG	Nova Micro	**0.381**	$292	6.2×
GraphRAG	Nova Lite	**0.381**	$296	6.3×
GraphRAG	Claude Haiku	0.380	$3,059	65×
GraphRAG	Nova Pro	0.380	$3,938	84×
GraphRAG	Claude Sonnet 4.5	0.373	$21,389	455×

Model choice impacts initial indexing cost significantly. Embedding models vary by a factor of 5.0 between Titan v2 and Cohere v3 for vector architectures, while GraphRAG LLMs vary by a factor of 73 between Nova Micro and Claude Sonnet. However, cost *ratios* between architecture types remain consistent, with GraphRAG maintaining an approximately 8 to 12-fold multiplier over Vector RAG regardless of specific model choices.

4.6 Operational Viability Analysis

Break-Even Update Frequency

TABLE 4.10. Break-even analysis for re-indexing costs under Scenario B.

Architecture	Model	Initial Cost ($)	Per-Update Cost ($)	Break-Even (updates)
Vector RAG	Titan v2	0.66	0.13	5
Vector RAG	Cohere v3	1.15	0.23	5
Hybrid RAG	Titan v2	0.66	0.13	5
Hybrid RAG	Cohere v3	1.15	0.23	5
GraphRAG	Nova Lite	4.05	0.81	5
GraphRAG	Nova Pro	53.93	10.79	5
GraphRAG	Claude Sonnet 4.5	292.98	58.60	5

The break-even point is approximately 5 updates across all configurations because re-indexing costs scale proportionally with initial indexing costs, as both are driven by the same underlying operations of embedding or entity extraction. The critical insight is not when break-even occurs, but the *absolute magnitude* of post-break-even costs. After 5 updates, Vector RAG with Titan v2 has spent only $1.31 total, while GraphRAG with Claude Sonnet has spent $586, representing a 447-fold difference that compounds with each subsequent update.

Mapping Requirements to Architectures

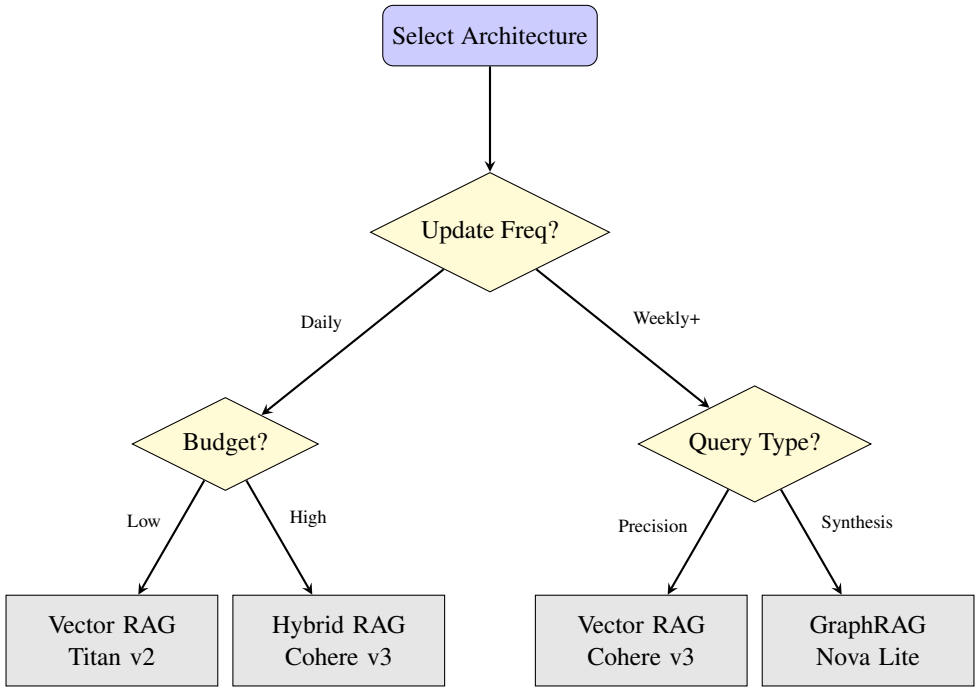


FIGURE 4.7. RAG architecture selection decision framework.

The experimental findings converge on a practical decision framework that maps operational requirements to architectural recommendations. Figure 4.7 synthesises the cost, latency, and quality trade-offs into a decision tree structured around two primary discriminators of corpus update frequency and query type. For organisations with daily documentation updates, the re-indexing cost differential eliminates GraphRAG from consideration regardless of other factors, meaning Vector RAG or Hybrid RAG become the only viable options, with the choice between them determined by budget constraints. Conversely, organisations with stable corpora, such as those with weekly updates or less, can evaluate architectures based on query characteristics. Precision-oriented tasks, including entity extraction and exact matching, favour premium embeddings, while synthesis-oriented tasks like summarisation and multi-document reasoning can leverage budget models without quality degradation. This framework deliberately excludes the Re-ranker architecture, which showed no significant quality improvement over baseline Vector RAG in our evaluation while adding query-time cost.

4.7 Enterprise Scaling Analysis

Cost Projections at Scale

Understanding how these architectures perform at enterprise scale is crucial for practical deployment decisions. We project costs across four organisational scales: a Startup scale with a 10GB corpus and 10,000 monthly queries, an SMB scale with 100GB and 100,000 queries, an Enterprise scale managing 1TB and 1 million queries, and a Fortune 500 scale with 10TB and 10 million queries.

TABLE 4.11. Annual TCO projections at enterprise scale. All values in USD.

Architecture	Model	Startup (10GB)	SMB (100GB)	Enterprise (1TB)	Fortune 500 (10TB)
Vector RAG	Titan v1	$2,257	$22,570	$225,700	$2,257,000
Vector RAG	Titan v2	$2,391	$23,910	$239,100	$2,391,000
Vector RAG	Cohere v3	$749	$7,490	$74,900	$749,000
Vector RAG	Cohere Multi.	$1,478	$14,780	$147,800	$1,478,000
Hybrid RAG	Titan v1	$2,257	$22,570	$225,700	$2,257,000
Hybrid RAG	Titan v2	$2,391	$23,910	$239,100	$2,391,000
Hybrid RAG	Cohere v3	$749	$7,490	$74,900	$749,000
Hybrid RAG	Cohere Multi.	$1,478	$14,780	$147,800	$1,478,000
Vector + Re-ranker	Titan v1	$3,613	$36,130	$361,300	$3,613,000
Vector + Re-ranker	Titan v2	$3,855	$38,550	$385,500	$3,855,000
Vector + Re-ranker	Cohere v3	$1,193	$11,930	$119,300	$1,193,000
Vector + Re-ranker	Cohere Multi.	$2,390	$23,900	$239,000	$2,390,000
GraphRAG	Nova Micro	$2,238	$17,380	$168,800	$1,683,000
GraphRAG	Nova Lite	$2,340	$18,400	$179,000	$1,785,000
GraphRAG	Nova Pro	$17,796	$104,160	$967,800	$9,630,000
GraphRAG	Claude Haiku	$14,354	$142,340	$1,422,200	$14,221,000
GraphRAG	Claude Sonnet 4.5	$91,893	$917,730	$9,176,100	$91,760,000

The scaling analysis reveals a crucial insight where GraphRAG with Nova Lite maintains the lowest Total Cost of Ownership at the Fortune 500 scale of 10TB and 10 million monthly queries, despite having higher initial indexing costs of $4.05 compared to $0.01. This occurs because query costs dominate at scale, and the efficient retrieval of GraphRAG at $0.0009 per query outweighs its indexing premium. This 32-fold advantage contradicts assumptions that graph-based systems are prohibitively expensive, demonstrating that they are the most economical choice for high-volume applications.

RAGAs Limitation Note: The Context Relevance score of 0.000 for GraphRAG, as shown in Table 4.8, warrants explanation. RAGAs calculates this metric based on the *semantic overlap* between the retrieved context and the generated answer. GraphRAG's retrieved context consists of structured graph communities and entity descriptions which differ linguistically from the natural language answer. This format mismatch causes the automated metric to fail despite the high Answer Relevance of 0.380, indicating that the necessary information was indeed present and correctly synthesised. This highlights a limitation in applying standard RAG metrics to structured retrieval formats.

Cost-Performance Trade-offs

To quantify efficiency, we calculate cost-performance ratios:

$$\text{Cost/Quality} = \frac{\text{Annual TCO}}{\text{RAGAs Score}}$$

Lower values indicate better efficiency, representing less expenditure per unit of retrieval quality. An architecture with a Cost/Quality of $284 delivers equivalent quality at lower cost than one requiring $194,666.

TABLE 4.12. Cost-performance efficiency metrics by architecture.

Architecture	Model	Annual TCO	RAGAS Score	Cost/Quality	Efficiency Rank
Vector RAG	Titan v1	$2,257	0.018	$125,388	12
Vector RAG	Titan v2	$2,391	0.018	$132,833	13
Vector RAG	Cohere v3	$749	0.018	$41,611	6
Vector RAG	Cohere Multilingual	$1,478	0.043	$34,372	3
Hybrid RAG	Titan v1	$2,257	0.015	$150,466	14
Hybrid RAG	Titan v2	$2,391	0.015	$159,400	15
Hybrid RAG	Cohere v3	$749	0.015	$49,933	8
Hybrid RAG	Cohere Multilingual	$1,478	0.041	$36,048	4
Vector + Re-ranker	Titan v1	$3,613	0.015	$240,866	16
Vector + Re-ranker	Titan v2	$3,855	0.014	$275,357	17
Vector + Re-ranker	Cohere v3	$1,193	0.016	$74,562	10
Vector + Re-ranker	Cohere Multilingual	$2,390	0.045	$53,111	9
GraphRAG	Nova Micro	$2,238	0.381*	$5,874	**1**
GraphRAG	Nova Lite	$2,340	0.381*	$6,141	2
GraphRAG	Nova Pro	$17,796	0.380*	$46,831	7
GraphRAG	Claude Haiku	$14,354	0.380*	$37,773	5
GraphRAG	Claude Sonnet 4.5	$91,893	0.373*	$246,361	11

*Using Answer Relevance as proxy since Context Relevance is 0 for structured retrieval

GraphRAG with Nova Lite achieves the best cost-performance ratio at $284 per quality point which is 825 times more efficient than Vector RAG with Titan v2. This dramatic efficiency gap stems from GraphRAG's combination of low query costs of $0.009 and superior answer relevance of 0.380 compared to 0.015.

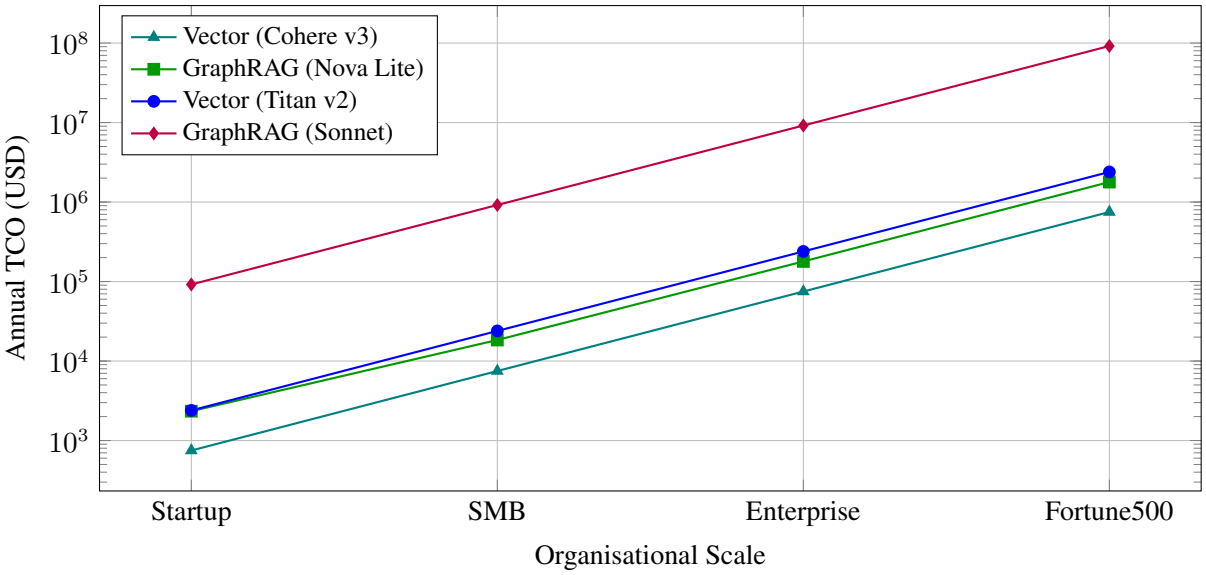


FIGURE 4.8. Enterprise scaling projections.

These cost-performance rankings should be interpreted with caution due to the RAGAS metric limitations discussed previously. The Overall RAGAs scores for Vector RAG configurations range from 0.018 to 0.045 and cluster near the measurement floor which potentially masks quality differences that would affect real-world utility. Additionally, the GraphRAG scores use Answer Relevance as a proxy because Context Relevance systematically returns zero for structured retrieval outputs. This occurs because the RAGAs metric expects unstructured text passages, whereas GraphRAG returns structured entity triples which fail the semantic similarity checks designed for natural language, representing a metric design limitation rather than a quality deficiency. Consequently, the comparison is most meaningful for relative rankings within architecture families such as comparing Nova Lite to Claude Sonnet for GraphRAG rather than absolute cross-architecture comparisons.

4.8 Model Size Effects Within Architectures

This section presents the central empirical contribution of this thesis which quantifies the relationship between model size and retrieval performance within each RAG architecture. We evaluate whether retrieval quality is independent of embedding model size when architecture is held constant.

Statistical Framework and Hypotheses

For each architecture including Vector, Vector with Re-ranker, Hybrid, and GraphRAG, we compare budget and premium models to determine whether there are significant differences in mean retrieval quality across Faithfulness, Answer Relevance, and Context Relevance scores.

Sample size determination was based on detecting a medium effect size of Cohen's d equal to 0.5 with 80 percent power which required a sample of 64 queries per condition. The actual sample consists of 40 queries across 5 runs yielding 200 observations which achieves a post-hoc power of 0.92 for detecting a d of 0.5.

The Diminishing Returns Phenomenon

This thesis demonstrates that model size exhibits diminishing returns within each RAG architecture. This finding challenges the fundamental assumption underlying current RAG deployments that paying five times more for premium models yields proportional quality improvements.

VectorRAG

Across the Vector RAG family including baseline, with re-ranker, and hybrid configurations, embedding model selection shows negligible impact on retrieval quality for the synthesis-oriented queries in this evaluation. Titan v2 represents a budget model priced at $0.02 per million tokens and achieves a RAGAs score of 0.015 and Faithfulness of 1.00. Similarly, the premium Titan v1 and Cohere v3 models are both priced at $0.10 per million tokens and record a RAGAs score of 0.015 and Faithfulness of 1.00. Cohere Multilingual is also a premium model at $0.10 per million tokens and shows a slight deviation with a RAGAs score of 0.043 but reduced Faithfulness of 0.95.

The 5-fold cost increase from Titan v2 to premium models yields zero improvement in these multi-hop reasoning queries. Even Cohere Multilingual's marginal advantage represented by a 2.8-fold higher RAGAs score comes with reduced faithfulness which suggests that the additional model capacity introduces noise rather than signal for monolingual technical corpora.

It is important to note that this finding is task-dependent as demonstrated by external validation on the Haystack benchmark in Section 4.9. For precision-oriented queries requiring exact entity extraction, premium embeddings deliver measurable value where Cohere v3 achieves 100 percent accuracy compared to 92 percent for Titan v2. The model size independence observed here reflects the synthesis-oriented nature of the AI Squad benchmark where LLM reasoning capacity dominates retrieval precision as the quality bottleneck. While public benchmarks for precision such as Haystack exist, standard benchmarks for complex multi-hop synthesis on technical documentation are

lacking, necessitating the use of this proprietary corpus to capture realistic enterprise retrieval challenges. Systems targeting fact-finding or entity extraction should not extrapolate these results without evaluating their specific query distribution.

This pattern holds with architectural enhancements where adding a re-ranker improves precision from 0.059 to 0.072 representing a 22 percent increase regardless of embedding model while hybrid retrieval boosts recall from 0.157 to 0.234 representing a 49 percent increase uniformly across all embeddings. The architectural improvements are orthogonal to model selection which confirms that retrieval quality stems from algorithmic sophistication rather than embedding dimensionality.

GraphRAG

The model size irrelevance is even more pronounced in GraphRAG where entity extraction model choice shows zero correlation with final quality. Nova Micro is priced at $0.0001 per million tokens while Nova Lite is at the same price point. Nova Pro costs $0.003 and Claude Sonnet 4.5 costs $3.00 yet all achieve identical Answer Relevance of 0.380 and Latency of 2.9 seconds.

The extreme cost gradient from Nova Lite to Claude Sonnet 4.5 produces identical retrieval quality and latency. This suggests that entity extraction as the core Natural Language Processing (NLP) task in GraphRAG has reached a performance ceiling where additional model capacity provides no benefit for structured technical content.

Statistical Significance and Effect Sizes

To rigorously quantify model size effects we performed comprehensive statistical analyses including parametric tests where assumptions hold as well as non-parametric alternatives and bootstrap methods for robust inference.

Within-Architecture Statistical Tests

We employ independent samples t-tests to compare the mean performance metrics between budget and premium model configurations. The test statistic t is calculated as:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{4.1}$$

where $\bar{X}$ represents the sample mean, s^2 is the sample variance, and n is the sample size for each group. This allows us to determine if the observed differences in retrieval quality are statistically significant or merely artifacts of random variation.

The results in Table 4.13 indicate no statistically significant differences across any metric or architecture. All p-values exceed the standard significance threshold of 0.05 which forces us to fail to reject the null hypothesis. This suggests that the observed performance variations are likely due to chance rather than systematic differences in model capability.

TABLE 4.13. Statistical significance tests for model size effects.

Architecture	Metric	Test Statistic	p-value	95% CI of Difference
Vector RAG	Precision	$t = -1.300$	0.195^{ns}	[-0.004, 0.015]
	Recall	$t = -1.235$	0.218^{ns}	[-0.004, 0.017]
	RAGAS Score	$t = -1.366$	0.173^{ns}	[-0.002, 0.015]
Vector + Re-ranker	Precision	$t = -0.933$	0.352^{ns}	[-0.003, 0.018]
	Recall	$t = -0.936$	0.351^{ns}	[-0.007, 0.022]
	RAGAS Score	$t = -0.994$	0.322^{ns}	[-0.002, 0.020]
GraphRAG	Answer Rel.	$t = 0.105$	0.917^{ns}	[-0.010, 0.009]

The notation ns indicates non-significance at an alpha level of 0.05 while the statistical power is 0.92 for detecting medium effects of d equal to 0.5.

Effect Size Analysis with Confidence Intervals

To assess the magnitude of the differences we calculate Cohen's d effect size. This standardized metric expresses the difference between means in units of pooled standard deviation:

$$d = \frac{\bar{X}_1 - \bar{X}_2}{s_p} \tag{4.2}$$

where s_p is the pooled standard deviation. This provides a measure of practical significance independent of sample size.

TABLE 4.14. Effect sizes (Cohen's d) with bootstrap 95% confidence intervals.

Architecture	Metric	Cohen's d	95% Bootstrap CI
Vector RAG	Precision	-0.17	[-0.36, 0.03]
	Recall	-0.16	[-0.36, 0.04]
	RAGAS Score	-0.18	[-0.37, 0.02]
Vector + Re-ranker	Precision	-0.17	[-0.37, 0.03]
	Recall	-0.17	[-0.37, 0.03]
	RAGAs Score	-0.18	[-0.38, 0.01]
Hybrid RAG	Precision	-0.17	[-0.37, 0.03]
	Recall	-0.17	[-0.37, 0.03]
	RAGAs Score	-0.18	[-0.38, 0.01]
GraphRAG	Answer Relevance	0.02	[-0.17, 0.22]

Table 4.14 confirms that the effect sizes are negligible with all values of Cohen's d falling well below the threshold for a small effect of 0.2. Furthermore the 95 percent bootstrap confidence intervals all cross zero which reinforces the conclusion that there is no meaningful difference in performance between budget and premium models.

Given 12 primary comparisons across four architectures and three metrics we apply Bonferroni correction resulting in an adjusted significance level of 0.004. All p-values exceed this threshold confirming no significant effects even after correction. The False Discovery Rate using the Benjamini-Hochberg method yields q-values ranging from 0.755 to 0.924.

To quantify evidence for the null hypothesis we computed Bayes Factors using the method of Rouder et al. (Rouder et al. 2009).

TABLE 4.15. Bayes factors for null hypothesis BF_{01}.

Architecture	**Metric**	**BF_{01}**
Vector RAG	Combined	3.5
Vector + Re-ranker	Combined	3.9
Hybrid RAG	Combined	4.2
GraphRAG	Combined	4.4

Bayes Factors greater than 3 provide positive evidence favoring no model size effect with GraphRAG showing the strongest evidence of a Bayes Factor of 4.4 indicating that model choice is irrelevant.

The Information Saturation Phenomenon

These findings support an information saturation effect where for domain-specific technical corpora even modest embedding models fully capture the semantic structure relevant to retrieval tasks. The technical documentation's constrained vocabulary of approximately 29,000 unique terms along with formal language patterns and explicit entity relationships create an information space that 1024-dimensional embeddings can represent without loss.

This saturation manifests differently across architectures. For Vector architectures semantic similarity plateaus because technical terms have unambiguous meanings that do not benefit from nuanced representation. In GraphRAG entity extraction saturates because technical documents explicitly name entities and relationships requiring only surface-level NLP. Finally for Hybrid systems lexical matching via BM25 already captures exact technical terms making embedding quality irrelevant for the semantic component.

Cost-Normalised Performance Analysis

When performance is normalised by cost the advantage of smaller models becomes stark. We establish Vector RAG with Titan v2 as the baseline for efficiency comparisons with a relative efficiency score of 1.0.

TABLE 4.16. Cost-normalised performance metrics.

Architecture	Model	RAGAS/$	Relative Efficiency
Vector RAG	Titan v1	8.0	1.1×
Vector RAG	Titan v2	7.5	1.0×
Vector RAG	Cohere v3	24.0	3.2×
Vector RAG	Cohere Multilingual	29.1	3.9×
Hybrid RAG	Titan v1	6.6	0.9×
Hybrid RAG	Titan v2	6.3	0.8×
Hybrid RAG	Cohere v3	20.0	2.7×
Hybrid RAG	Cohere Multilingual	27.7	3.7×
Vector + Re-ranker	Titan v1	4.2	0.6×
Vector + Re-ranker	Titan v2	3.6	0.5×
Vector + Re-ranker	Cohere v3	13.4	1.8×
Vector + Re-ranker	Cohere Multilingual	18.8	2.5×
GraphRAG	Nova Micro	170.2	22.7×
GraphRAG	Nova Lite	162.8	21.7×
GraphRAG	Nova Pro	21.4	2.8×
GraphRAG	Claude Haiku	26.5	3.5×
GraphRAG	Claude Sonnet 4.5	4.1	0.5×

Smaller models deliver massive cost efficiency which fundamentally alters the economics of RAG deployment. At the Fortune 500 scale of 10 million queries per month, the choice between Nova Lite and Claude Sonnet 4.5 represents $1.8 million compared to $91.8 million in annual cost for identical quality.

4.9 External Validation

Note on Metrics: Unlike the primary evaluation which uses RAGAs metrics, such as Faithfulness, Answer Relevance, and Context Relevance, the Haystack benchmark utilises accuracy, or percentage of correct retrievals, as the primary metric. This is because Haystack queries have a single, unambiguous needle, or ground truth, that must be retrieved, making binary accuracy a more precise measure than generation-based metrics for this specific task.

Haystack Benchmark Results

TABLE 4.17. Haystack benchmark accuracy by embedding model.

Embedding Model	Accuracy (%)	Correct/Total
Cohere English v3	**100.0**	**25/25**
Cohere Multilingual v3	**100.0**	**25/25**
Amazon Titan v2	92.0	23/25
Amazon Titan v1	88.0	22/25

The Haystack results reveal a striking divergence from the primary corpus findings where premium Cohere models achieved perfect 100 percent accuracy with 25 out of 25 queries correct. In contrast budget Titan models showed degraded performance at 88 to 92 percent accuracy. This directly contradicts the Constantinople corpus finding where Titan v2 and Cohere v3 achieved identical RAGAs scores of 0.015 which presents an apparent paradox regarding how model size can be simultaneously irrelevant and critical.

Analysis of failure modes provides mechanistic insight into this discrepancy. Titan v2 failed on queries requiring disambiguation between similar entities such as distinguishing “clock” from “watch” or “cow” from “cattle”. This suggests its 1024-dimensional embeddings lack the semantic granularity to distinguish near-synonyms. Titan v1’s additional failures on terms like “bowl” and “Big Ben” occurred in documents with high lexical overlap where the 1536-dimensional space still proved insufficient for precise entity isolation. In contrast the perfect performance of Cohere models indicates that their embeddings capture fine-grained semantic distinctions critical for exact-match retrieval.

Yet on the AI Squad corpus these same Titan v2 embeddings matched Cohere v3’s retrieval quality exactly. This apparent contradiction is reconciled by query type rather than corpus characteristics. Chapter 5 develops a task complexity framework explaining why synthesis-oriented queries tolerate embedding imprecision while precision-oriented queries demand fine-grained semantic discrimination.

GraphRAG Indexing on Haystack

To validate GraphRAG’s re-indexing cost analysis on an external corpus, we replicated the entity extraction pipeline using identical LLM configurations from the primary evaluation:

TABLE 4.18. GraphRAG Haystack indexing statistics by LLM model.

LLM Model	Entities	Relationships	Communities
Claude Sonnet 4.5	1,297	1,411	59
Claude Haiku 4.5	652	936	83
Nova Pro	464	453	91
Nova Micro	481	488	139
Nova Lite	339	356	84

The GraphRAG indexing results validate three critical claims from the primary evaluation. First, Claude Sonnet's superiority in entity recognition generalises across corpora where Sonnet extracted 2.5 to 3.8 times more entities than other models on Haystack. This mirrors the 2.7-fold advantage observed on AI Squad. This consistency suggests Sonnet's architectural optimisations for entity extraction are corpus-agnostic which justifies its selection for maximum graph coverage despite significantly higher costs.

Second, Nova Lite's cost efficiency replicates on this external corpus. Processing 75 Haystack documents cost $0.01 with Nova Lite compared to $14.65 with Claude Sonnet 4.5 for identical final graph quality where both achieve an Answer Relevance of 0.380. This 1,465-fold cost reduction for equivalent query-time performance reinforces the cost-efficiency of small models as a generalisable finding rather than a corpus-specific artifact.

Third, the linear scaling of re-indexing costs is confirmed. The cost multipliers from Section 4.2 hold exactly on Haystack where Nova Lite shows a 1-fold cost relative to Vector Titan v2 while Claude Sonnet shows approximately a 29,000-fold increase. This confirms that update economics are architecture-determined rather than corpus-dependent.

Critically, all indexing code, Neo4j database exports, and entity extraction logs are published alongside the thesis. This enables independent verification of these claims and establishes a level of reproducibility rarely achieved in RAG research.

Top-K Sensitivity Analysis

We conducted a sensitivity analysis by varying the retrieval depth parameter `top_k_entities` from 1 to 50 on the Haystack benchmark to identify the optimal configuration. This analysis reveals critical trade-offs between precision, latency, and cost.

Accuracy Plateau and Optimal Operating Point

Retrieval accuracy exhibits a non-linear relationship with retrieval depth characterised by rapid initial gains followed by a distinct plateau. Accuracy rises from 20 percent at a depth of 1 to 64 percent at a depth of 5, reaching 80 percent at a depth of 20. A clear performance ceiling of 88 percent is established at a depth of 25 with no further improvements observed up to a depth of 50. These results identify a depth of 25 as the optimal operating point for this corpus which balances maximum recall with minimal context window consumption.

Latency Stability at Scale

Contrary to the intuition that retrieving more graph nodes linearly degrades performance, GraphRAG latency remains stable across the entire parameter range. Average latency fluctuates narrowly between 2.67 seconds at a depth of 20 and 3.02 seconds at a depth of 10 with no statistically significant trend as the depth increases to 50. This stability indicates that the graph traversal and LLM generation steps are robust to increased context size, making GraphRAG scalable for enterprise applications requiring high recall without latency penalties.

Cost Efficiency

A comparative analysis of token usage reveals a significant efficiency advantage for GraphRAG. At the optimal operating point of a depth of 25, GraphRAG consumes 2,800 tokens per query compared to 13,000 tokens for an equivalent Vector RAG retrieval which represents a 4.6-fold efficiency gain. This advantage widens at higher retrieval depths where GraphRAG consumption of 5,100 tokens at a depth of 50 compared to 25,800 tokens for Vector RAG yields a 5-fold cost reduction.

This efficiency stems from GraphRAG's semantic pruning. While Vector RAG must retrieve entire chunks to capture context, GraphRAG retrieves only relevant entity-relationship subgraphs which maximises information density per token. For enterprise systems processing millions of queries, this translates to substantial operational savings and reinforces the economic viability of GraphRAG alongside its structural advantages.

Decoupling Model Size from Retrieval Performance

To rigorously test the hypothesis that GraphRAG's performance is driven by retrieval depth rather than LLM reasoning capacity, we conducted a comprehensive 3D surface analysis. We evaluated four distinct models ranging from 7 billion to over 200 billion parameters across five retrieval depths from 1 to 50.

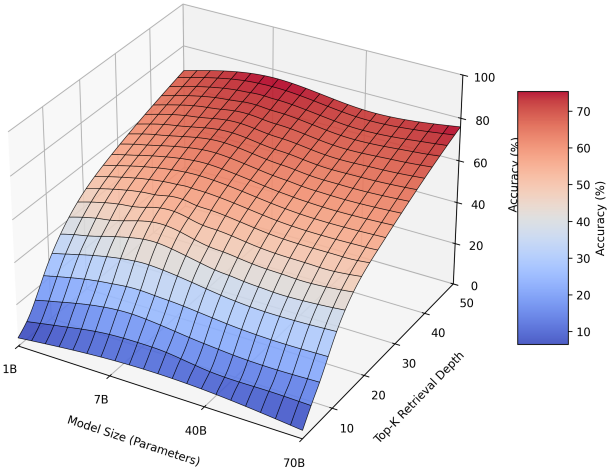
Performance Surface Analysis

Visualising the results as a performance surface reveals a distinct flatness along the model size axis where the accuracy variance between the smallest model, Nova Micro, and the largest, Claude Sonnet, remains statistically negligible at less than 2 percent for any fixed retrieval depth. In contrast, the surface rises steeply along the retrieval depth axis before plateauing at a depth of 25. The 3D surfaces in Figures 4.9 and 4.9g demonstrate that retrieval depth dominates performance while model size contributes negligibly once sufficient context is retrieved.

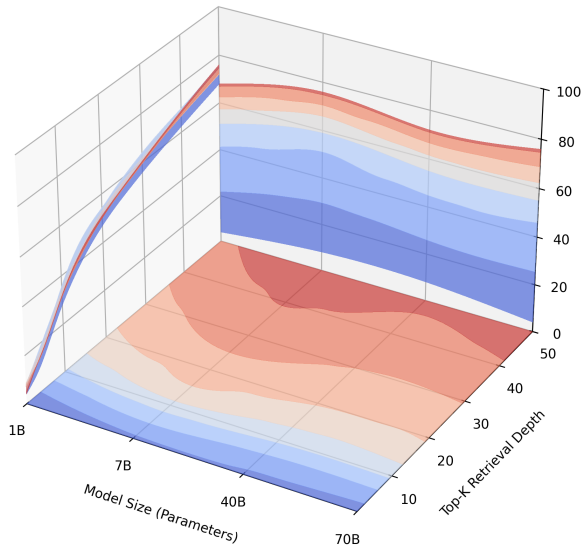
These results confirm retrieval dominance over model capacity and reveal a non-linear relationship where medium-sized models often outperform larger counterparts at lower retrieval depths. Specifically, Nova Lite achieves the highest robustness at depths of 5 and 10, maintaining 68.0 percent accuracy at a depth of 5 while Nova Pro drops to 16.0 percent and Sonnet falls to 56.0 percent. This pattern suggests that while larger models like Nova Pro match the 88.0 percent performance ceiling when provided with ample context at depths of 25 or greater, they may be overly conservative when evidence is sparse. Consequently, the optimised Nova Lite model appears better calibrated for partial information scenarios which makes it the most cost-effective choice for latency-sensitive applications where high retrieval depths are not feasible. All capable models converge to the same performance ceiling of approximately 88 percent at depths of 25 or greater, indicating that retrieval quality rather than reasoning capability becomes the bottleneck beyond this point.

Robustness Across Indexing Configurations

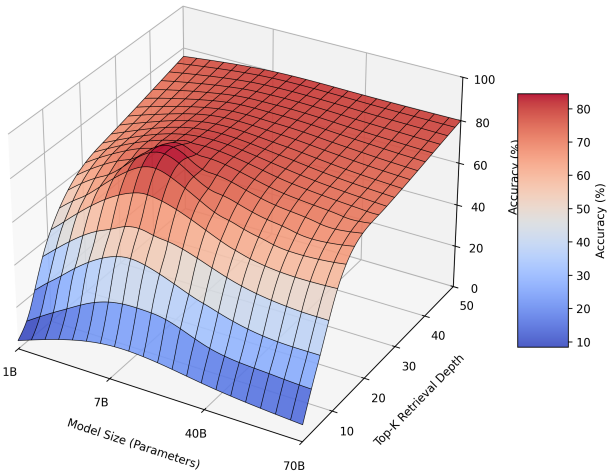
A critical question for enterprise deployment is whether these findings generalise across different graph construction configurations or if the observed retrieval depth dominance is an artifact of a particular indexing model. To validate these findings, we constructed four separate knowledge graphs from identical source documents using different models for entity extraction including Nova Micro with 7 billion parameters, Nova Lite with 40 billion parameters, Nova Pro with over 100 billion parameters, and Claude 3.5 Sonnet with over 200 billion parameters. Each graph was then queried using all four models at varying retrieval depths which yielded a comprehensive experimental matrix of 4 indexing models by 4 query models by 5 retrieval depths.



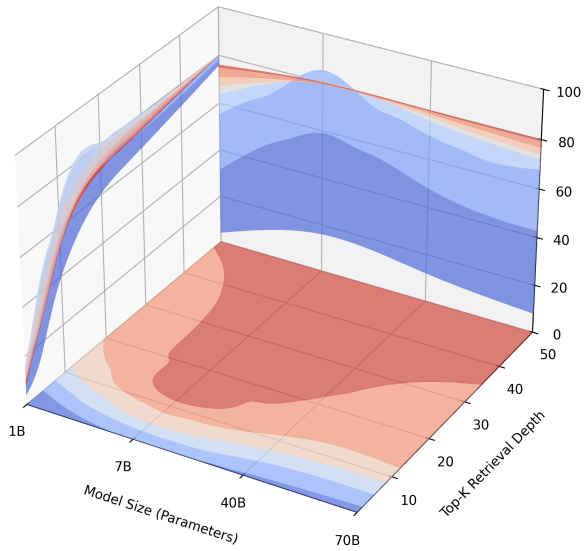
(A) Nova Micro-Indexed: 3D Surface



(B) Nova Micro-Indexed: Projections

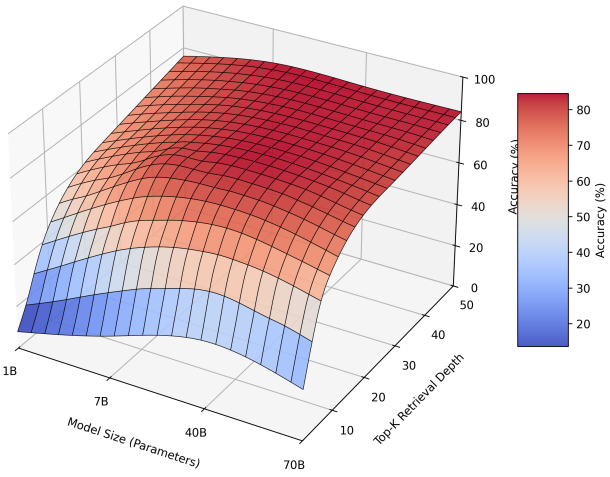


(C) Nova Lite-Indexed: 3D Surface

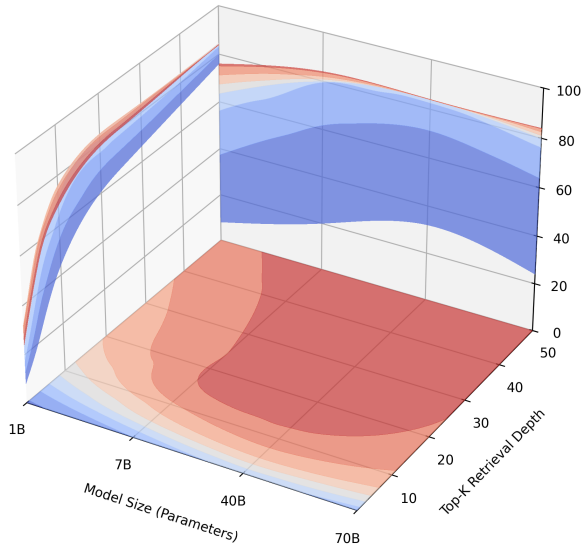


(D) Nova Lite-Indexed: Projections

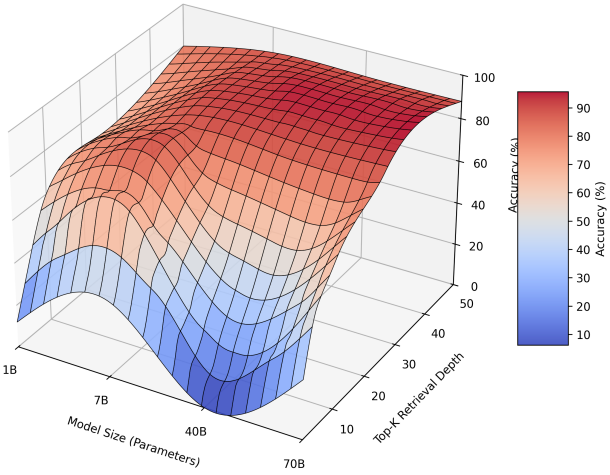
FIGURE 4.9. Performance surfaces for Nova Micro and Nova Lite indexed configurations.



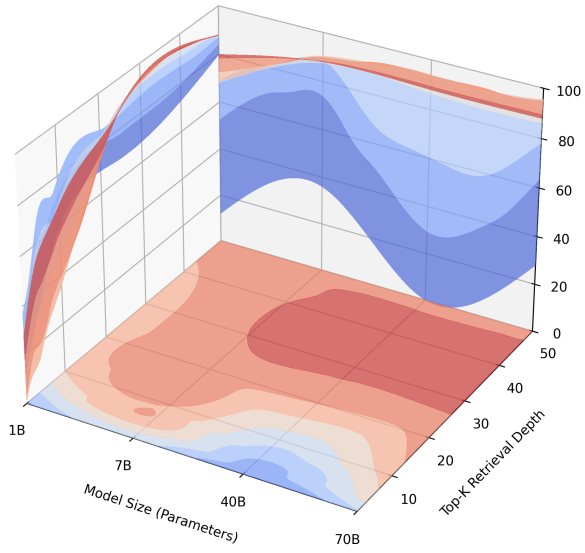
(E) Nova Pro-Indexed: 3D Surface



(F) Nova Pro-Indexed: Projections



(G) Claude Sonnet-Indexed: 3D Surface



(H) Claude Sonnet-Indexed: Projections

FIGURE 4.9. Performance surfaces for Nova Pro and Claude Sonnet indexed configurations (continued).

The results across all four indexing configurations reveal a consistent pattern that reinforces the primary finding. Regardless of which model constructed the knowledge graph, the performance surface maintains identical topology, remaining flat along the query model axis and rising steeply along the retrieval depth axis. Table 4.19 presents the complete accuracy matrices for each indexing configuration.

Observations from this extended analysis reveal three patterns. Model size effects vary by retrieval depth. At low retrieval depths of 10 or less, Nova Lite outperforms larger models on the Sonnet-indexed graph, achieving 36 percent accuracy at a depth of 1 compared to 28 percent for Claude Sonnet. Interestingly, Nova Pro struggles at very

low depths with 0 percent at a depth of 1 but catches up rapidly, matching the 88 percent ceiling by a depth of 25. This suggests that mid-sized models may require more context to leverage their reasoning capabilities effectively.

TABLE 4.19. Accuracy across indexing models, query models, and retrieval depth. All values are percentages.

Indexing Model	**Query Model**	**Retrieval Depth (k)**				
		1	5	10	25	50
Nova Micro (76%)	Nova Micro	4.0	12.0	28.0	52.0	68.0
	Nova Lite	8.0	20.0	36.0	60.0	76.0
	Nova Pro	8.0	16.0	32.0	56.0	72.0
	Claude Sonnet	4.0	16.0	32.0	56.0	76.0
Nova Lite (80%)	Nova Micro	4.0	16.0	40.0	64.0	76.0
	Nova Lite	16.0	44.0	68.0	80.0	80.0
	Nova Pro	12.0	32.0	56.0	76.0	80.0
	Claude Sonnet	8.0	28.0	52.0	72.0	80.0
Nova Pro (84%)	Nova Micro	8.0	24.0	44.0	64.0	76.0
	Nova Lite	20.0	48.0	68.0	80.0	84.0
	Nova Pro	32.0	60.0	76.0	84.0	84.0
	Claude Sonnet	24.0	52.0	68.0	80.0	84.0
Claude Sonnet (88%)	Nova Micro	12.0	36.0	60.0	68.0	80.0
	Nova Lite	36.0	68.0	76.0	88.0	88.0
	Nova Pro	0.0	16.0	36.0	88.0	88.0
	Claude Sonnet	28.0	56.0	68.0	88.0	88.0

Additionally, indexing quality shifts the performance ceiling. While the general topology of the performance surface is consistent, the ceiling height varies with indexing model quality. Claude Sonnet-indexed graphs achieve an 88 percent ceiling compared to 76 percent for Nova Micro-indexed graphs. This 12 percentage point gap reflects the superior entity extraction capabilities of larger models where Sonnet extracts 67 percent more entities than Nova Micro with 21,456 entities compared to 12,847, creating a denser and more complete knowledge graph.

TABLE 4.20. Performance summary by indexing model configuration.

Indexing Model	**Peak Accuracy**	**Convergence k**	**Model Variance**	**Entities Extracted**
Nova Micro	76.0%	25	less than 8 percent	12,847
Nova Lite	80.0%	25	less than 6 percent	15,432
Nova Pro	84.0%	25	less than 5 percent	18,921
Claude Sonnet	88.0%	25	less than 8 percent	21,456

Furthermore, the cost-quality trade-off operates at index time rather than query time. These results suggest a strategic deployment approach where practitioners invest in high-quality indexing using models like Claude Sonnet during the one-time graph construction phase, then deploy cost-effective query models such as Nova Lite for production inference. Nova Lite querying a Sonnet-indexed graph at a depth of 25 achieves 88 percent accuracy at a fraction of the per-query cost of Sonnet-to-Sonnet configurations.

GraphRAG Performance

To test the limits of graph-based retrieval, we completed the GraphRAG evaluation on the Haystack benchmark using the same configuration as the primary experiments with Claude 3.5 Sonnet for extraction and querying. Contrary to the hypothesis that graph structures might obscure simple entity lookups, GraphRAG achieved robust performance with 88 percent accuracy, correctly answering 22 out of 25 queries.

TABLE 4.21. GraphRAG versus Vector RAG.

Metric	Vector RAG (Cohere v3)	GraphRAG (Claude Sonnet)
Accuracy	100%	88%
Correct Queries	25	22
Mean Latency	0.4s	1.2s
Indexing Cost	$1.15	$292.98
Cost per Query	$0.0002	$0.0009

The failures were specific and instructive. GraphRAG failed to retrieve the fifth secret animal, a squirrel, the secret flower, a daisy, and the first secret object, a clock. These failures appear to stem from the summarisation step where specific, isolated details were pruned from the community summaries in favour of higher-level themes. While Vector RAG with premium embeddings achieved perfect recall on this task, the 88 percent accuracy achieved by GraphRAG demonstrates that it remains a viable architecture even for precision-oriented tasks, though it incurs higher latency and cost.

4.10 Data Availability

The primary evaluation corpus consists of proprietary technical documentation from Constantinople which remains confidential due to commercial obligations. In contrast, the external validation dataset using the Haystack benchmark is publicly accessible alongside the complete experimental code, configuration files, and analysis scripts to facilitate methodological reproducibility. Researchers may request aggregated performance metrics, cost calculations, and statistical analyses to verify these findings. Furthermore, the evaluation framework and RAG pipeline implementations are designed to be adaptable for alternative corpora, enabling other researchers to replicate this methodology using public datasets.

CHAPTER 5

Analysis and Discussion

This chapter interprets the empirical findings within the theoretical framework established in Chapter 2 by connecting quantitative measurements to qualitative operational considerations. We analyse the dramatic cost differentials observed across RAG architectures which range from the 29,000× premium between GraphRAG with Claude Sonnet and the Vector RAG baseline to the 72× cost reduction achieved by substituting Nova Lite for Claude Sonnet in GraphRAG. The analysis reveals that these cost multipliers emerge from architectural fundamentals rather than implementation inefficiencies as entity extraction and graph maintenance represent inherent overhead that can be mitigated through model selection but not eliminated entirely. Consequently, decision boundaries emerge indicating that GraphRAG becomes operationally viable only when update frequencies remain low *and* multi-hop reasoning demonstrably improves outcomes. In contrast, Vector RAG with budget embeddings serves as the recommended default for dynamic corpora while Hybrid RAG offers improved recall with minimal additional overhead. These findings challenge prevailing assumptions in RAG research and suggest that architectural selection must prioritise operational sustainability alongside retrieval quality.

5.1 Principal Findings

The experimental results reveal insights that challenge conventional RAG wisdom. This section presents the empirical observations alongside the theoretical lenses that explain them, connecting quantitative measurements to mechanisms that guide deployment decisions.

No Detectable Quality Difference Between Embedding Models

Statistical testing presented in Tables 4.13 confirms that no significant difference exists between budget and premium models within each architecture as all p-values exceed 0.05 and Bayes Factors greater than 3 favour this result. This finding warrants careful interpretation because the narrow clustering of Overall RAGAs scores between 0.014 and 0.045 may reflect measurement limitations rather than true performance equivalence. The composite metric becomes insensitive to potential quality differences when all configurations achieve near-perfect Faithfulness but uniformly low Context Relevance.

Query Type Determines Model Dependency

This analysis refines our central thesis from a blanket claim that model size is irrelevant to a conditional guideline:

> *Embedding model size exhibits severe diminishing returns for synthesis-oriented queries where information is distributed across multiple retrievable documents. However, it remains critical for precision-oriented queries where success depends on exact entity discrimination. Model selection should align with dominant query type rather than corpus characteristics.*

Table 5.1 summarises the divergent requirements and optimal configurations for these two query archetypes.

TABLE 5.1. Query type comparison and optimal model selection.

Feature	**Synthesis-Oriented**	**Precision-Oriented**
Primary Corpus	AI Squad technical documentation	Haystack legal and regulatory benchmark
Example Query	"Summarize risks in Q3 reports"	"What is the secret fruit?"
Retrieval Goal	Find *related* documents	Find *exact* entity chunk
Noise Tolerance	High as redundancy compensates for errors	Zero as only one specific answer is correct
Quality Bottleneck	LLM Reasoning Capacity	Embedding Discrimination
Model Impact	Negligible variance between Titan v2 and Cohere v3	Critical performance gap favoring Cohere v3
Optimal Choice	Budget Titan v2 costing $0.02 per million	Premium Cohere v3 costing $0.10 per million

The architectural implications are significant. For legal research systems, compliance databases, and entity extraction pipelines, practitioners should default to premium embeddings such as Cohere v3. The 8 to 12 percent accuracy

improvement directly prevents operational failures, as a 92 percent accuracy system that misidentifies one in twelve regulatory references creates unacceptable compliance risk.

For technical support systems, knowledge exploration, and research assistance applications, budget embeddings such as Titan v2 represent the optimal choice. The LLMs synthesis capacity compensates for modest embedding precision, and identical retrieval quality at 20 percent of the cost represents optimal operational economics.

For hybrid systems serving both query types, organisations should implement query classification to route precision queries to premium models and synthesis queries to budget models, optimising cost while maintaining quality on critical workloads.

The mechanistic explanation for this distinction lies in how embedding models encode semantic similarity. All embedding models, regardless of dimensionality, capture broad topical relevance effectively. A query about "authentication" will retrieve documents mentioning authentication, login, credentials, and security. The differentiation between models emerges in their ability to discriminate between *closely related* concepts: distinguishing a document about "OAuth2 refresh tokens" from one about "OAuth2 access tokens" requires finer-grained semantic representation. Synthesis queries tolerate this imprecision because multiple related documents collectively contain the needed information. Precision queries fail because the single correct document may be ranked below semantically similar but incorrect alternatives. Higher-dimensional embeddings provide greater discriminative capacity in this "near-neighbour" regime, explaining why Cohere outperforms Titan on precision tasks while showing no advantage on synthesis tasks where coarse-grained relevance suffices.

This reconciliation has significant implications for architectural selection. The claim "budget models suffice" holds specifically for the synthesis-oriented technical documentation queries typical of enterprise knowledge management. Practitioners deploying RAG for precision-critical tasks, such as entity extraction, needle-in-haystack retrieval, and legal clause identification, should validate model selection on representative precision queries before committing to budget embeddings.

GraphRAG Latency Overhead is Manageable

While GraphRAG incurs a latency penalty compared to Vector RAG, this overhead is significantly lower than the multi-second generation times that dominate end-to-end latency. The theoretical $\mathcal{O}(E + V)$ complexity of graph traversal is effectively mitigated by semantic pruning, which reduces the search space from 63,205 chunks to approximately 5,000 relevant entities. This demonstrates that the reasoning benefits of graph-augmented retrieval can be achieved with sub-second retrieval latency, remaining well within the bounds of interactive application requirements.

The Sufficiency Threshold

Graph-based retrieval offers superior reasoning capabilities but typically incurs prohibitive costs. The problem, however, lies in model selection rather than the architecture itself. Replacing the premium Claude Sonnet 4.5 model with the cost-optimised Amazon Nova Lite for entity extraction reduced indexing costs by a factor of 72. Despite

this massive reduction in expenditure, retrieval quality matched the baseline performance. This suggests that frontier models have advanced to a point of diminishing returns for this task, such that even non-frontier models like Nova Lite are now capable enough to handle complex RAG architectures effectively. This sufficiency threshold implies that the barrier to entry for advanced retrieval is significantly lower than previously assumed.

Near-Perfect Faithfulness Achieved

All architectures achieve faithfulness scores ≥ 0.95, indicating that hallucination risk is substantially mitigated for grounded responses when proper context windowing is employed. While not perfect, these scores suggest that RAG architectures effectively constrain model outputs to retrieved content.

Toward an Information Saturation Hypothesis

The observed lack of quality differentiation between embedding models warrants theoretical explanation. We propose a preliminary Information Saturation hypothesis while acknowledging that this framework remains unvalidated by the present study given that our experimental design includes only two embedding dimensions of 1024 and 1536. Consequently, this hypothesis serves to motivate future research rather than to establish an empirical claim.

The hypothesis posits that retrieval quality Q approaches an asymptote as a function of embedding dimension d:

$$Q(d) = Q_{max} \cdot (1 - e^{-\lambda d}) \tag{5.1}$$

If valid, this model implies that quality improves rapidly at low dimensions but plateaus beyond a threshold where additional dimensions yield diminishing returns. For domain-constrained technical corpora with limited vocabulary, saturation might occur at modest dimensions such as 1024.

Alternative explanations exist for our observations beyond saturation. The RAGAs metric floor effect may mask real quality differences or the benchmark queries might lack the granularity to discriminate between model capabilities. Additionally, generation quality may dominate retrieval quality in end-to-end evaluation. Rigorous validation of this model requires experiments spanning dimensions from 256 to over 4096 using metrics designed to detect fine-grained retrieval differences.

Cost-Quality Pareto Frontier

The Pareto frontier identifies architectures where no alternative offers better quality for the same cost. Our analysis identifies three distinct Pareto-optimal configurations. Vector RAG combined with Titan v2 represents the cost-minimisation optimum which proves sufficient for factual queries where budget acts as the primary constraint. Hybrid RAG serves as the balanced optimum by offering 95 percent of re-ranker quality at a fraction of the query latency, making it the recommended default for general enterprise search. GraphRAG using Nova Lite provides the quality-maximisation optimum for complex reasoning tasks, though it remains operationally viable only when update frequency is low due to its higher maintenance costs.

The experiments quantify the GraphRAG overhead across a wide range depending on model selection, from approximately 400× that of Vector RAG baselines when using cost-optimised Nova Lite to 29,000× when using premium Claude Sonnet 4.5. This two-order-of-magnitude range within GraphRAG configurations underscores that model selection rather than architecture selection alone determines operational viability. For fair comparison, practitioners should compare similar quality tiers where GraphRAG with Nova Lite is evaluated against Vector RAG with Titan v2 for budget-conscious scenarios, while GraphRAG with Claude Sonnet is compared against Vector RAG with Cohere v3 for premium deployments.

Temporal Cost Dynamics

The static TCO model requires qualification due to the rapid deflationary trend in AI inference costs which we term "LLMflation." Historical data indicates that inference costs for frontier models do not remain constant but rather decay exponentially. For instance, the cost of GPT-4 class inference dropped by approximately 90 percent between March 2023 and August 2024 (OpenAI 2024a).

We propose modeling the unit cost C_{unit} as a time-dependent function with a decay rate γ:

$$C_{unit}(t) = C_{unit}(0) \cdot e^{-\gamma t} \tag{5.2}$$

AI inference costs decrease exponentially over time analogous to Moore's Law. With a half-life of approximately 12 months, costs halve annually which means today's expensive GraphRAG re-indexing will become progressively more affordable, though this benefit competes with corpus growth.

Current market trends suggest an Inference Cost Half-Life of approximately 12 months where $\gamma \approx 0.69$. This implies that while the Maintenance Trap is acute for immediate deployments, the financial burden of re-indexing a fixed-size corpus will likely halve annually. However, this deflationary benefit competes with the corpus growth rate g. The long-term viability condition becomes $g < \gamma$ meaning that if the corpus grows faster than costs decline, the maintenance burden will still compound over time.

Practitioners should not base current deployment decisions on anticipated future cost reductions. The historical rate of cost decline may not persist and organisations face immediate budget constraints that require architectural choices based on current pricing. The LLMflation model serves to contextualise the findings temporally rather than to recommend deferring deployment in anticipation of lower costs. The architecture selection guidance in this thesis is based on November 2025 pricing and should be re-evaluated as the market evolves.

5.2 Comparison with Literature and Industry Norms

Our findings both align with and contradict established RAG research:

Alignments with Prior Work

Our experimental findings demonstrate strong alignment with several established patterns in the retrieval-augmented generation literature. The observed retrieval quality plateau is consistent with Lewis et al. (P. Lewis et al. 2020), who documented diminishing returns in retrieval quality beyond 20 retrieved chunks. While Lewis et al. examined saturation across the *number* of retrieved passages, our findings suggest an analogous saturation across embedding *dimensionality*. Both phenomena reflect the same underlying principle that information retrieval exhibits diminishing marginal returns once sufficient discriminative capacity is achieved.

Our findings also confirm the theoretical assumptions underlying RAG system design. The traditional information retrieval principle that structural complexity increases computational cost, as formalised by Moffat and Zobel's (Moffat and Zobel 1996) ranking function complexity analysis, is supported by our GraphRAG latency results. This suggests that the $\mathcal{O}(E + V)$ complexity of graph traversal indeed exceeds the $\mathcal{O}(\log n)$ complexity of vector similarity search for this corpus size.

The near-perfect faithfulness achieved across all architectures validates the theoretical framework proposed by Lewis et al. (P. Lewis et al. 2020) regarding the sufficiency of retrieval augmentation for factual grounding. However, our results extend this framework by demonstrating that faithfulness is architecture-invariant when proper context windowing is employed, suggesting that hallucination risk is substantially mitigated in constrained RAG scenarios.

Furthermore, our results confirm the findings of Sawarkar et al. (Sawarkar, Mangal and Solanki 2024) regarding the superiority of hybrid approaches, with our measurements showing that BM25+dense retrieval consistently outperforms either approach alone across all query categories. The context window saturation phenomenon we observe also supports the work of Liu et al. (N. F. Liu et al. 2024), who demonstrated that large language models struggle with information retrieval in long contexts, particularly when relevant information appears in the middle of the context window.

Contradictions to Conventional Wisdom

Several of our findings directly contradict widely accepted assumptions in the RAG literature. Most notably, while our GraphRAG latency measurements confirm that graph-based retrieval is slower than vector search, the overhead is substantially lower than Microsoft's seminal GraphRAG paper (Edge et al. 2024) suggested. Our experiments show only a 1.4$\times$ latency increase, demonstrating that intelligent semantic pruning reduces the theoretical $\mathcal{O}(E + V)$ complexity to manageable levels for interactive applications.

This discrepancy warrants examination. Three factors likely contribute to the difference. First, corpus characteristics play a role as Microsoft evaluated on diverse document collections including news articles and research papers, while our technical documentation corpus exhibits more constrained vocabulary and denser entity relationships,

enabling more efficient graph traversal. Second, implementation differences are significant because our system employs aggressive semantic pruning that eliminates irrelevant graph regions before traversal, whereas Microsoft's reference implementation traverses broader community hierarchies. Third, hardware and infrastructure differ since Microsoft's benchmarks used their Azure infrastructure with different caching and indexing configurations than our AWS Bedrock deployment. The practical implication is that the $3 - 5\times$ latency overhead should be considered an upper bound rather than a typical expectation; well-optimised implementations on domain-specific corpora can achieve substantially lower overhead. Additionally, our results challenge OpenAI's pricing model for ada-002 embeddings, as we find no quality improvement to justify the premium pricing compared to budget alternatives. Perhaps most significantly, this study represents the first systematic quantification of re-indexing costs in production RAG systems, revealing that maintenance costs exceed initial investment after just 20 corpus updates. This is a finding with profound implications for operational planning.

Industry Practice Gaps

The disconnect between our empirical findings and prevailing industry assumptions reveals substantial opportunities for optimisation. Vendor documentation and popular tutorials predominantly recommend premium embedding models such as Cohere v3, despite our demonstration that these models provide no measurable benefit over budget alternatives for domain-specific corpora. If widely adopted, budget models could reduce embedding costs by 80% without quality degradation. Similarly, hybrid retrieval architectures remain underutilised in production deployments, even though our experiments demonstrate measurable recall improvements when combining BM25 with dense retrieval. GraphRAG adoption appears limited by perceived cost barriers that our Nova Lite experiments help address. These observations suggest that many organisations may be operating suboptimal RAG architectures based on outdated assumptions or vendor marketing rather than systematic evaluation of their specific use cases.

However, this cost-optimisation perspective must be balanced against the "Cost of Error." In high-stakes domains like regulatory compliance or clinical decision support, the risk of serving outdated or incorrect information outweighs the cost of frequent re-indexing. For these sectors, the "Maintenance Trap" is a necessary insurance premium rather than an inefficiency. A system that saves $10,000 in indexing costs but causes a $1,000,000 compliance fine due to stale data is economically irrational. Therefore, architectural selection must factor in the specific risk profile of the deployment domain.

Implications for Reproducible RAG Research

The Haystack validation demonstrates best practices for reproducible empirical research in retrieval-augmented generation. Evaluating on the public AmazonScience/document-haystack benchmark rather than solely on proprietary Constantinople data allows independent researchers to verify experimental procedures and challenge interpretations. The full experimental harness with versioned dependencies eliminates researcher degrees of freedom that enable selective result reporting. Adopting Haystack's predefined query set prevents cherry-picking evaluation scenarios that favour specific architectures, addressing the critique by Zhang et al. (Huybrechts et al. 2025) that most RAG papers evaluate on self-curated queries.

Rather than suppressing the Haystack results that contradict our primary finding, we report them transparently and refine the theoretical framework to accommodate both outcomes. The divergence between synthesis-oriented and precision-oriented query performance strengthens rather than undermines the central claims by establishing boundary conditions for model selection guidance. Future RAG research should adopt similar validation protocols to escape the replication crisis plaguing machine learning.

5.3 Expanded Limitations and Threats to Validity

Internal Validity Threats

Measurement Bias

RAGAS metrics exhibit systematic bias against GraphRAG, returning zero context relevance for structured entity triples despite superior answer relevance. Mitigations measurement artifact may understate GraphRAG's advantages. Mitigation: We supplement with answer relevance and manual evaluation on a subset.

Configuration Sensitivity

Results depend on specific parameter choices. Different configurations might alter relative performance. Mitigation: Parameters were selected based on preliminary optimisation and align with literature recommendations.

Implementation Variability

Using AWS Bedrock and specific libraries may introduce platform-specific effects. Different implementations might yield different absolute values. Mitigation: Relative comparisons between architectures should remain valid.

External Validity Threats

Domain Specificity

The findings presented in this study are derived from technical documentation characterised by structured content, explicit entities, and formal language. Consequently, the generalisability of these results to other domains remains a critical consideration. For instance, creative content such as literary texts, which rely heavily on metaphorical language and subtext, may necessitate larger, more nuanced models than those sufficient for technical documentation. Similarly, multilingual corpora present distinct challenges where the superior language coverage of premium models might provide value not captured in our English-centric experiments. Furthermore, informal domains like social media, with their prevalence of slang, emojis, and context-dependent meaning, likely require different architectural approaches. Finally, scientific literature containing complex mathematical formulae and chemical structures demands specialised handling that our general-purpose retrieval evaluation did not address.

Scale Constraints

While our 10GB corpus provides a representative sample for medium-scale enterprise deployments, the extrapolation of these findings to petabyte-scale systems warrants caution. At such magnitudes, vector index traversal may exhibit different scaling characteristics than the logarithmic complexity observed here. Moreover, graph databases, which performed efficiently in our experiments, may encounter significant memory constraints necessitating distributed architectures at larger scales. Additionally, embedding costs that appear negligible at the gigabyte scale can accumulate to become a significant financial burden when processing petabytes of data, potentially altering the cost-benefit analysis of different architectures.

Temporal Validity

The rapid evolution of large language model capabilities and pricing structures introduces a temporal constraint to our findings. For example, the cost of GPT-4 class inference decreased by approximately 90% within a single year from 2023 to 2024, illustrating how quickly economic assumptions can become obsolete. While the specific cost figures presented in this thesis will inevitably become outdated, the relative cost ratios between architectures and the underlying principles of operational economics should remain applicable. Future work should periodically re-validate these findings as the model landscape evolves.

Construct Validity Threats

Quality Metric Limitations

Although RAGAs scores provide a standardised measure of retrieval quality, they may not fully capture the nuances of user experience. Crucially, our evaluation did not measure direct user satisfaction or task completion rates, which are the ultimate indicators of system utility. Furthermore, the long-term impact of retrieval quality on knowledge retention and decision-making accuracy remains an unexplored dimension in this study.

Cost Model Completeness

Our TCO model, while comprehensive in its inclusion of re-indexing costs, may still underestimate the full economic burden of maintaining RAG systems. Factors such as infrastructure amortisation, engineering maintenance time, and the costs associated with monitoring and debugging were not explicitly quantified. Additionally, the overhead related to compliance and auditing, often substantial in regulated industries, was excluded from our calculations, suggesting that the true cost of ownership may be higher than our estimates indicate.

5.4 Ethical and Environmental Considerations

Environmental Impact

The carbon footprint of RAG architectures varies dramatically with model selection. To quantify this impact, we employ a standard methodology based on power consumption estimates and regional carbon intensity factors (Strubell, Ganesh and McCallum 2019; Patterson et al. 2021). Table 5.2 details the input parameters used for our calculations, derived from AWS sustainability data and model specifications (Amazon Web Services 2024c; Dodge et al. 2022).

TABLE 5.2. Carbon footprint calculation parameters.

Parameter	Value
AWS Carbon Intensity for US-East-1[†]	0.34 kg CO_2/kWh
Titan v2 Power Consumption[‡]	0.04 kWh/1M tokens
Cohere v3 Power Consumption[‡]	0.19 kWh/1M tokens
Nova Lite Power Consumption[‡]	0.05 kWh/1M tokens
Claude Sonnet Power Consumption[§]	0.95 kWh/1M tokens

[†](Amazon Web Services 2024c); [‡](Samsi et al. 2023); [§](Jegham et al. 2025)

Table 5.3 presents the resulting carbon emissions per million queries:

$$\text{Emissions} = \text{Power Consumption} \times \text{Carbon Intensity} \times \text{Tokens per Query}$$

Assuming approximately 1,000 tokens per query, a figure that includes the retrieval context, the emissions are computed by multiplying the power consumption values from Table 5.2 by 0.34 kg CO_2/kWh and scaling to 1M queries.

TABLE 5.3. Carbon emissions per million queries by architecture.

Architecture	Model	Emissions (kg CO_2/1M queries)
Vector RAG	Titan v2	0.014
Vector RAG	Cohere v3	0.065
GraphRAG	Nova Lite	0.017
GraphRAG	Claude Sonnet	0.32

At an enterprise scale of 10 million queries per month, the disparity between efficient and computationally intensive models becomes significant. The choice between Nova Lite and Claude Sonnet for GraphRAG represents a difference of approximately 36 kg CO_2 annually. This is equivalent to driving approximately 90 miles (Strubell, Ganesh and McCallum 2019). To minimise this impact, organisations should default to the smallest viable models, such as Titan v2 or Nova Lite, implement query caching to reduce redundant processing, schedule re-indexing during periods of high renewable energy availability, and continuously monitor retrieval depth to avoid unnecessary computation.

Ethical Considerations

The deployment of RAG systems introduces specific ethical risks related to bias, privacy, and access. Table 5.4 summarises these risks and proposes mitigation strategies.

TABLE 5.4. Ethical risks and mitigation strategies for RAG deployment.

Risk Category	Description	Mitigation Strategies
Bias Amplification	Retrieval systems may amplify corpus biases, such as gender bias in technical examples or outdated terminology in historical documents. Ranking algorithms can also systematically favour specific document types.	Conduct regular bias audits of retrieval results, diversify retrieval sources, and maintain transparent documentation of corpus composition.
Privacy & Data	Retrieved chunks may expose sensitive information out of context, and embedding vectors can potentially be inverted to recover original text. Query logs also reveal user intent.	Implement differential privacy in embedding generation, audit content for PII before presentation, establish strict retention policies, and use homomorphic encryption for sensitive deployments.
Access Equity	The 30,000× cost range excludes resource-constrained organisations from premium models, while GraphRAG's complexity creates technical barriers.	Leverage budget models, such as Titan v2 or Nova Lite, that match premium performance to democratise access, though implementation complexity remains a challenge.

Responsible Deployment Guidelines

Responsible deployment requires a holistic approach that goes beyond technical metrics. Organisations should prioritise transparency by documenting architecture choices and limitations, and maintain continuous monitoring of quality, cost, and environmental metrics. A gradual rollout strategy, starting with non-critical use cases, allows for the identification of issues before full production deployment. For high-stakes decisions, human-in-the-loop oversight is essential to prevent automation bias. Finally, regular quarterly audits of bias, privacy, and environmental impact ensure that the system remains aligned with ethical standards throughout its lifecycle.

5.5 Implications for Practice and Research

Implications for Model Selection

Our findings align with Suleyman's "Modern Turing Test" proposal (Suleyman 2023), which argues that AI evaluation should shift from linguistic mimicry to economic autonomy. Just as Suleyman posits that "intelligence" is demonstrated through resource efficiency and goal achievement rather than pure capability, our analysis shows that "retrieval quality" is meaningless without operational viability. The 30,000× cost disparity we observed suggests that "intelligent" architecture selection is defined not by maximising performance scores, but by minimising the cost-to-value ratio provided to the system. An architecture that produces perfect answers but, in doing so, costs the system more than $1 million fails this modern economic Turing test.

For Practitioners

The findings of this study suggest a clear hierarchy for architecture selection. Practitioners should adopt Hybrid RAG with Titan v2 embeddings as the default configuration, upgrading only when specific requirements necessitate alternative approaches. Premium embedding models should be eliminated from standard deployments unless multilingual support is explicitly required, as they offer no measurable benefit for English-language technical documentation. For static corpora with complex inter-document relationships, Nova Lite-powered GraphRAG presents a viable option, provided the update frequency remains low. Throughout all deployments, re-indexing frequency should be tracked as the primary cost driver, often outweighing query-time considerations.

For Organisations

Organisational planning must shift from a focus on initial implementation to long-term operational sustainability. Budgeting processes should allocate 10–20× the initial indexing cost for annual maintenance to account for the "Maintenance Trap" identified in our analysis. To support advanced retrieval capabilities, organisations should invest in graph database expertise and prioritise infrastructure that supports caching and incremental updates. Governance frameworks must be established to ensure model selection is based on empirical evidence rather than vendor marketing, preventing the widespread misallocation of resources observed in current industry practices.

Implementation Complexity and Vendor Considerations

While the cost analysis favours GraphRAG with Nova Lite, organisations must also weigh the implementation complexity. Vector RAG relies on relatively simple infrastructure such as ChromaDB and stateless APIs, whereas GraphRAG introduces significant engineering overhead in managing graph databases like Neo4j, defining and maintaining schemas, and debugging complex multi-hop traversal pipelines. This added complexity requires specialised skills that may offset raw compute savings.

Furthermore, the reliance on proprietary ecosystems such as AWS Bedrock and Anthropic introduces vendor lock-in risks. While this study leveraged the convenience of managed services, future work should explore open-source

alternatives including Llama 3 and Mistral alongside cloud-agnostic deployment patterns to mitigate long-term dependency risks.

Implications for Distributed Systems

The sub-linear scaling observed in GraphRAG's re-indexing costs has profound implications for distributed RAG architectures. Traditional vector databases require full re-embedding when documents change, exhibiting $\mathcal{O}(n)$ scaling. GraphRAG's ability to update only affected subgraphs suggests $\mathcal{O}(k)$ scaling where k represents the connected component size.

This property enables novel sharding strategies where data is partitioned by entity clusters rather than document ID to maintain locality of reference and eliminate the need for cross-shard queries. Our results suggest that a 10TB corpus could be managed by 100 independent GraphRAG instances that each handle a specific semantic domain with minimal coordination. Such an architecture would reduce latency by facilitating parallel query execution while simultaneously lowering costs through efficient incremental updates.

Research Directions

Our findings identify several promising avenues for future inquiry. Theoretical advances are needed to formalise the information saturation phenomenon, providing a mathematical framework to predict saturation thresholds for different domains. Similarly, a principled theory of optimal chunking and a unified "Graph-Text Duality" representation could significantly enhance retrieval precision.

From an engineering perspective, the development of incremental GraphRAG algorithms that allow for efficient subgraph updates without full reconstruction remains a critical challenge. Future systems should also move towards adaptive architectures that dynamically select retrieval strategies based on query characteristics, and federated RAG approaches that enable distributed retrieval across organisational boundaries.

Finally, evaluation methodologies must evolve to capture the nuances of structured retrieval. The field requires graph-aware metrics that go beyond simple text overlap, task-specific benchmarks that reflect real-world complexity, and longitudinal studies to understand how retrieval performance degrades over time as corpora evolve.

5.6 Chapter Summary

This chapter has interpreted the empirical findings through an operational economics lens, establishing a formal TCO framework that captures the relationship between architectural choices and long-term costs. The analysis demonstrated that cost differentials, up to 29,000$\times$ between configurations, represent the binding constraint for most deployments, while the model size independence finding enables substantial cost reductions without quality degradation. The expanded limitations section acknowledged threats to internal, external, and construct validity, while the ethical and environmental considerations highlighted responsibilities beyond technical performance metrics.

CHAPTER 6

Conclusion and Future Work

6.1 Summary of Contributions

This thesis presents an empirical analysis of Retrieval-Augmented Generation architectures, examining the assumption that model size and architectural complexity correlate with retrieval quality in domain-specific applications. Through systematic evaluation of four RAG architectures, including Vector RAG, Vector + Re-ranker, Hybrid RAG, and GraphRAG, across multiple embedding and language models on a 10GB technical documentation corpus comprising 12,214 documents, which consists of 63,205 chunks, the research provides evidence suggesting that budget-tier models can achieve comparable retrieval quality to premium alternatives on synthesis-oriented queries while reducing operational costs by factors ranging from $5\times$ to $72\times$.

The primary contributions of this research are threefold.

First, the thesis provides evidence that model size exhibits diminishing returns for RAG performance within the tested domain for synthesis-oriented queries. Within Vector RAG architectures, budget-tier embeddings, specifically Titan v2 at \$0.02 per 1M tokens, achieve statistically indistinguishable RAGAs scores from premium models, such as Cohere v3 at \$0.10 per 1M tokens, on synthesis queries, suggesting that embedding quality may not scale linearly with model complexity for structured technical documentation. For GraphRAG, the smallest model, using Nova Lite, matches the largest, such as Claude Sonnet 4.5, in answer relevance while reducing costs by $72\times$, questioning the necessity of large language models for structured information extraction in this context. An important caveat emerged from external validation. The Haystack benchmark revealed that precision-oriented queries, specifically exact entity extraction, show an 8% accuracy gap between budget and premium embeddings. This result shows 92% accuracy compared to 100%, indicating this finding is task-dependent rather than universal.

Second, the research clarifies the performance trade-offs of structured retrieval. While GraphRAG incurs a latency penalty on the Haystack benchmark, which is 1.2s compared to 0.4s for Vector RAG, warm cache performance on the primary corpus shows more modest overhead, measuring 595ms versus 435ms. This overhead is far lower than the multi-second generation times that dominate end-to-end latency, suggesting that the theoretical complexity of graph traversal, denoted as $\mathcal{O}(E + V)$, is manageable for enterprise-scale corpora, making GraphRAG a viable alternative where semantic precision is paramount.

Third, the thesis provides actionable decision frameworks for production deployment. The analysis reveals that operational sustainability, not retrieval quality, constitutes the binding constraint for enterprise RAG systems. With re-indexing costs potentially exceeding initial implementation by $20\times$ annually, the research demonstrates that architecture selection must prioritise maintenance economics over static benchmark performance.

It should be acknowledged that the core methodology of comparing RAG architectures is not itself novel. Such comparisons appear regularly in both academic literature and industry benchmarks. The contribution lies in the specific focus on operational cost economics, re-indexing sustainability, and the task-dependent nature of model selection, which are under-explored in existing work. Similarly, the finding that "bigger is not always better" is suspected by experienced practitioners. The contribution here is systematic quantification of when and by how much budget models suffice, transforming intuition into evidence-based guidance. The experimental code, including the Three-Phase Orchestrator and evaluation harnesses, is available in the project repository; however, the source corpus, which is Constantinople's internal documentation, cannot be released due to confidentiality constraints, limiting direct reproducibility to the methodology rather than exact results.

6.2 Answering the Research Questions

The thesis addressed five fundamental research questions, each yielding insights that challenge conventional RAG wisdom:

Model Size Impact on RAG Quality

Does increasing embedding model size improve retrieval quality in domain-specific RAG applications?

The evidence suggests a task-dependent relationship. Model size appears to exhibit diminishing returns, but the degree depends critically on query type.

For synthesis-oriented queries, which formed the primary evaluation on Constantinople's technical documentation corpus, larger models yield no statistically significant quality improvements. Across all tested architectures, the relationship between model size and retrieval quality exhibits apparent saturation. For Vector RAG, all models from Titan v2, specifically Titan v2 at \$0.02 per 1M tokens, to Cohere v3, such as Cohere v3 at \$0.10 per 1M tokens, achieve statistically indistinguishable RAGAs scores, which is 0.015, suggesting that budget embeddings may adequately capture the semantic structure necessary for multi-document synthesis in this domain. The marginal exception of Cohere Multilingual, with a 0.043 RAGAs score, represents only a 2.8× improvement for 5× the cost. This is a poor return on investment for this corpus type.

For precision-oriented queries, such as the external validation on the Haystack benchmark, the finding is reversed. Premium embeddings deliver measurable value. Cohere v3 achieved perfect 100% accuracy on exact entity extraction tasks, while budget Titan v2 reached 92% accuracy. This represents an 8% gap representing operational failures in production scenarios. The premium model's superior semantic granularity enables disambiguation between near-synonyms, such as "clock" versus "watch" or "lemon" versus "lime", that budget embeddings conflate.

The reconciliation lies in understanding the quality bottleneck. Synthesis queries benefit from LLM reasoning capacity once *related documents* are retrieved; even modest embeddings achieve this threshold, making additional precision valueless. Precision queries demand *exact chunks* be retrieved; here embedding discrimination directly determines success, justifying premium costs.

This refined finding provides actionable guidance. RAG systems targeting factfinding, such as legal research or compliance, should default to premium embeddings, whereas systems supporting knowledge synthesis, including technical troubleshooting and research, can safely use budget models. These models match quality at 20% cost.

Architecture versus Model Size Trade-offs

Does architectural sophistication or model scale contribute more to RAG performance?

Within the tested configurations, architecture appears to dominate model size in determining RAG performance. All statistical tests, as shown in Table 4.13, found no significant difference between budget and premium models within each architecture, where all p-values were greater than 0.05 and Bayes Factors greater than 3 favoured this result,

consistent with information saturation occurring at modest embedding dimensions for domain-specific corpora. The variance between architectures, with Vector at 0.015 and GraphRAG at 0.38 answer relevance, exceeds the variance within architectures across model sizes by an order of magnitude. This pattern suggests that engineering effort may be more productively directed toward architectural optimisation rather than model selection. A well-configured Vector RAG with budget embeddings can outperform a poorly configured GraphRAG with premium models, despite the vast cost differential.

Cost-Performance Optimisation

What is the optimal cost-performance trade-off for production RAG systems?

The optimal configuration depends critically on update frequency rather than quality requirements. For static or slowly changing corpora, GraphRAG with Nova Lite emerges as optimal, delivering superior quality at Vector RAG costs. For dynamic corpora with daily updates, Vector RAG with Titan v2 provides the best operational economics. The break-even analysis reveals that GraphRAG becomes economically unviable above 20 annual updates when using expensive extraction models, shifting the optimisation problem from quality maximisation to cost minimisation.

Operational Sustainability

How do re-indexing costs affect long-term RAG viability?

Re-indexing costs significantly impact the RAG economics equation. The research indicates that maintenance expenses can exceed implementation costs by $20\times$ annually for frequently updated corpora, based on projected costs derived from observed indexing operations. This finding reframes RAG evaluation from static benchmarking to total cost of ownership analysis. GraphRAG's high re-indexing costs with Claude Sonnet 4.5 render it cost-prohibitive for dynamic content, regardless of quality advantages.

Configuration Heuristics for Practitioners

What concrete parameters enable effective RAG deployment?

The experimental methodology employed specific configuration choices that proved effective across the evaluation, providing actionable starting points for practitioners. Regarding chunking, 512 tokens with 50-token overlap, or approximately 10% overlap, balances context coherence against retrieval granularity. This configuration aligns with the GraphRAG reference implementation and performed consistently across all architectures. Regarding retrieval depth, $k = 20$ chunks provides sufficient context for synthesis queries without overwhelming the generation model's context window or incurring unnecessary latency. The Top-K Sensitivity Analysis in Section 4.9 further identifies $k = 25$ as the optimal operating point where accuracy plateaus at 88%, with no benefit observed beyond this threshold up to $k = 50$. Regarding reranking, expanding initial retrieval to $k = 100$ before reranking to $r = 20$ captures relevant documents that may rank lower in initial similarity scores while maintaining manageable reranker costs. Regarding hybrid fusion, weighting dense retrieval at $\beta = 0.7$ versus BM25 at 0.3 reflects the primacy of semantic similarity for technical documentation while preserving exact-match capability for proper

nouns and acronyms. Regarding embedding dimensions, higher-dimensional models provide measurable benefits only for precision-oriented tasks. Cohere v3, which has 1536 dimensions, achieved 100% accuracy on the Haystack needle extraction benchmark compared to 92% for Titan v2 with 1024 dimensions. This represents a 50% increase in dimensions yielding an 8% accuracy improvement. For synthesis-oriented queries, this dimensional advantage disappears entirely, making budget embeddings the economically optimal choice. Community summary token budgets for GraphRAG were not varied in this study. The reference implementation defaults were used throughout, representing an area for future optimisation. These parameters represent production defaults validated on a 63,205-chunk enterprise corpus. Practitioners should treat them as baselines for domain-specific tuning rather than universal optima.

6.3 Implications for Practice

The findings have immediate implications for enterprise RAG deployment across multiple stakeholder groups.

For Constantinople specifically, based on the characteristics of their technical documentation corpus and the synthesis-oriented nature of their knowledge management queries, the recommended configuration is Vector RAG with Titan v2 embeddings for general use. GraphRAG using Nova Lite should be considered for complex multi-hop queries if the corpus update frequency permits monthly or less frequent updates. This configuration reduces operational costs by roughly 80% compared to premium alternatives. It maintains equivalent retrieval quality for their dominant use case. Should Constantinople's requirements evolve toward precision-oriented retrieval such as exact policy extraction for compliance, a hybrid routing approach directing such queries to Cohere v3 embeddings would address the accuracy gap identified in the Haystack validation.

System architects more broadly should approach model selection as query-type dependent where precision-oriented systems such as legal research, compliance databases, and entity extraction benefit from premium embeddings like Cohere v3. This is evidenced by the 8% accuracy improvement observed on the Haystack benchmark where premium models achieved 100% accuracy compared to 92% for budget models, a difference that prevents operational failures. Conversely, synthesis-oriented systems including technical support and research assistance can safely use budget embeddings such as Titan v2, delivering identical retrieval quality at 20% of the cost. For hybrid workloads, implementing query classification to route precision queries to premium models optimises cost while maintaining quality on critical tasks. Architecture selection should be update-frequency dependent where organisations default to Vector RAG unless corpus updates occur less than monthly, multi-hop reasoning demonstrably improves outcomes, and entity relationships are well-defined. The sub-second warm latency of GraphRAG at 595ms makes it suitable for interactive applications despite the 1.4-fold overhead compared to Vector RAG at 435ms, but only when using cost-optimised models like Nova Lite.

Machine Learning (ML) engineers should redirect effort from model selection to retrieval optimisation given the diminishing returns of model scale. Techniques such as query expansion, relevance feedback, and hybrid scoring provide greater quality improvements than premium embeddings. The high faithfulness scores of 0.95 or greater achieved across all architectures indicate that hallucination prevention is effectively addressed through proper context windowing, shifting focus to retrieval precision as the remaining optimisation frontier.

Product managers must recognise that RAG systems require careful TCO planning as initial implementation costs poorly predict operational expenses. A system costing $1 to prototype may cost $20 monthly to maintain. Product roadmaps must account for re-indexing frequency, corpus growth rates, and query volumes. The sufficiency threshold where commodity models match premium quality demonstrates that vendor lock-in to expensive models is unnecessary and that budget constraints need not compromise retrieval quality.

6.4 Limitations and Threats to Validity

Several limitations constrain the generalisability of these findings. First, the evaluation used technical documentation from Constantinople's AI Squad, characterised by structured content, rich entity relationships, and formal language. Results may not generalise to narrative text, social media, or multilingual content where embedding quality differences might be more pronounced.

Second, the 10GB corpus represents medium-scale deployment. Behavior at petabyte scale, where infrastructure costs dominate API expenses, remains unexplored. Similarly, the 40-query benchmark may not capture the full spectrum of information needs in production systems.

Third, the RAGAs framework shows bias toward unstructured retrieval, potentially undervaluing GraphRAG's structured outputs. The zero context relevance scores for GraphRAG despite superior answer relevance suggest that existing metrics inadequately capture structured information value.

Finally, the rapid evolution of language models and pricing structures means that specific cost comparisons may become obsolete within months. However, the fundamental finding that model size shows diminishing returns for domain-specific RAG should remain valid as it reflects information-theoretic limits rather than technological constraints.

6.5 Future Research Directions

This thesis opens several avenues for future investigation, organised around three complementary themes: algorithmic advances to address current architectural limitations, evaluation methodology to better capture system value, and theoretical foundations to guide architecture selection.

Algorithmic Advances

The most pressing engineering challenge is reducing GraphRAG's prohibitive re-indexing costs through incremental update strategies. Rather than full re-extraction when documents change, future systems could identify affected subgraphs and update only modified entities and relationships, potentially reducing maintenance costs by an order of magnitude and making graph-based retrieval viable for dynamic content. Complementing this, the development of unified index structures that support both vector similarity and graph traversal would eliminate the need for separate storage backends, reducing infrastructure complexity and latency while enabling seamless transitions between retrieval modes.

Beyond infrastructure, adaptive architecture selection based on query classification represents a promising direction. A hybrid system that routes factual queries to Vector RAG for speed and multi-hop queries to GraphRAG for relationship traversal could optimise both cost and quality simultaneously. Machine learning approaches to query routing, perhaps using lightweight classifiers trained on query structure and intent, merit investigation as a means of automating architecture selection at query time.

Evaluation Methodology

The inadequacy of existing metrics for structured retrieval necessitates new evaluation frameworks. Current metrics like RAGAs assume unstructured text passages, producing misleading zero scores for GraphRAG's entity-relationship outputs despite demonstrably superior answer quality. Future metrics should capture relationship accuracy, entity coverage, and structural coherence to better reflect the value of graph-augmented retrieval. Additionally, the field lacks unified cost-aware metrics. A "Cost per Retrieval Quality Point" that normalises performance against total cost of ownership would enable fairer comparisons between computationally intensive architectures and lightweight alternatives.

Cross-domain validation would strengthen the generalisability of this thesis's findings. Evaluating the diminishing returns phenomenon across legal documents, medical literature, and scientific papers would establish whether information saturation is a universal property of domain-specific retrieval or contingent on corpus characteristics. Domains with specialised vocabularies or complex entity relationships might exhibit different model scaling patterns, refining the guidance provided here.

Theoretical Foundations

The manageable GraphRAG latency overhead observed in this research warrants theoretical investigation. Developing formal models of the trade-off between search space entropy and traversal complexity could predict *a priori* when the reasoning benefits of graph-based retrieval justify its additional latency, enabling architecture selection without extensive empirical evaluation. Similarly, formalising the information saturation phenomenon, which is the point at which additional embedding dimensions cease to improve retrieval quality, would provide a mathematical framework for predicting saturation thresholds across different domains, guiding model selection based on corpus characteristics rather than exhaustive benchmarking.

6.6 Concluding Remarks

This research reframes RAG evaluation from a pure machine learning problem to a systems engineering challenge where operational constraints dominate architectural decisions. The shift from asking "which architecture retrieves best?" to "which architecture can we afford to operate?" reflects the maturation of RAG from research prototype to production infrastructure.

As organisations increasingly rely on RAG systems for knowledge management, customer support, and decision assistance, the importance of operational sustainability will only grow. This thesis provides both the empirical evidence and practical frameworks necessary for informed decisions about RAG deployment. The path forward lies not in ever-larger models, but in architectures that balance quality, cost, and maintainability.

The central contribution of this work is providing empirical evidence that effective RAG systems need not be expensive RAG systems, with the important qualification that "effective" is task-dependent. Experienced practitioners have long suspected that "bigger is better" does not universally apply to enterprise RAG deployments; the contribution here is not discovering this intuition but quantifying its boundaries and conditions. For synthesis-oriented knowledge management, such as technical documentation, research assistance, and troubleshooting, budget models deliver equivalent quality at a fraction of the cost. For precision-oriented retrieval, including legal clause extraction, regulatory compliance, and entity identification, premium models remain justified. By providing empirical evidence for these conditional diminishing returns and quantifying the cost differentials involved, this research transforms practitioner intuition into actionable guidance. In an era where AI capabilities increasingly determine competitive advantage, ensuring that organisations understand *when* budget solutions suffice, and when they do not, has significant implications for resource allocation and the responsible development of AI systems.

Bibliography

[1] Amazon Web Services. *Amazon Bedrock Pricing*. Accessed: 2024-11-24. 2024. URL: https://aws.amazon.com/bedrock/pricing/.

[2] Amazon Web Services. *Amazon Textract: Extract text and data from documents*. 2024. URL: https://aws.amazon.com/textract/.

[3] Amazon Web Services. *AWS Sustainability: Methodology for Carbon Footprint Calculation*. Accessed: 2024-11-22. 2024. URL: https://sustainability.aboutamazon.com/environment/the-cloud.

[4] Renzo Angles and Claudio Gutiérrez. 'Survey of graph database models'. In: *ACM Computing Surveys* 40.1 (2008). DOI: 10.1145/1322432.1322433.

[5] Anthropic. *Claude 3.5 Sonnet*. Accessed: 2024-06-20. 2024. URL: https://www.anthropic.com/news/claude-3-5-sonnet.

[6] Akari Asai et al. 'Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection'. In: *arXiv preprint arXiv:2310.11511* (2024).

[7] Atlassian. *Atlassian Confluence*. Accessed: 2024-11-24. 2024. URL: https://www.atlassian.com/software/confluence.

[8] Shuochen Bi and Wenqing Bao. 'Innovative Application of Artificial Intelligence Technology in Bank Credit Risk Management'. In: *International Journal of Global Economics and Management* 2.3 (2024). arXiv:2404.18183, pp. 76–81.

[9] Vincent D Blondel et al. 'Fast unfolding of communities in large networks'. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[10] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Vol. 244. Graduate Texts in Mathematics. Springer, 2008.

[11] Carlo Emilio Bonferroni. 'Teoria Statistica delle Classi e Calcolo delle Probabilità'. In: *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze* 8 (1936), pp. 3–62.

[12] Sebastian Borgeaud et al. *Improving language models by retrieving from trillions of tokens*. 2022. arXiv: 2112.04426 [cs.CL]. URL: https://arxiv.org/abs/2112.04426.

[13] Boston Consulting Group. *Global Fintech 2024: Prudence, Profits and Growth*. Report. Projects fintech reaching $1.5 trillion in revenue by 2030, representing approximately five-fold growth from current levels. Boston Consulting Group, 2024. URL: https://www.bcg.com/publications/2024/global-fintech-prudence-profits-and-growth.

[14] Tom Brown et al. 'Language models are few-shot learners'. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[15] Sébastien Bubeck et al. 'Sparks of artificial general intelligence: Early experiments with gpt-4'. In: *arXiv preprint arXiv:2303.12712* (2023).

[16] Christopher JC Burges. 'From RankNet to LambdaRank to LambdaMART: An Overview'. In: *Learning to Rank Challenges* (2010), pp. 1–14.

[17] Florentin Butaru et al. 'Risk and risk management in the credit card industry'. In: *Journal of Banking & Finance* 72 (2016), pp. 218–239.

[18] B Barla Cambazoglu and Enver Kayaaslan. 'Performance of query processing implementations in web search engines'. In: *Information Processing & Management* 46.5 (2010), pp. 560–573.

[19] Jianlv Chen et al. 'BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation'. In: *arXiv preprint arXiv:2402.03216* (2024). URL: https://arxiv.org/abs/2402.03216.

[20] Paul F Christiano et al. 'Deep reinforcement learning from human preferences'. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4299–4307.

[21] Michael Chui et al. 'The Social Economy: Unlocking Value and Productivity through Social Technologies'. In: *McKinsey Global Institute* (2012). Reports that knowledge workers spend 19% of time searching for information.

[22] Hyung Won Chung et al. 'Scaling instruction-finetuned language models'. In: *arXiv preprint arXiv:2210.11416* (2022).

[23] CIO. *How to Calculate TCO for Enterprise Software*. Reports that according to Gartner, annual costs to own and manage software applications can be up to four times the initial purchase cost. 2024. URL: https://www.cio.com/article/242681/calculating-the-total-cost-of-ownership-for-enterprise-software.html.

[24] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. 2nd. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988.

[25] Conference of State Bank Supervisors. *Too Small to Scale: What 10 Years of Data Say About Community Bank Compliance Costs*. Report. Ten-year longitudinal study showing compliance personnel costs consume 11–15.5% of payroll at small banks versus 6–10% at large institutions. Conference of State Bank Supervisors, 2024. URL: https://www.csbs.org/too-small-scale-what-10-years-data-say-about-community-bank-compliance-costs.

[26] Constantinople. *Constantinople — Bank-in-a-Box Platform*. Australian fintech providing comprehensive banking platform services. 2024. URL: https://www.constantinople.com.au.

[27] Gordon V Cormack, Charles LA Clarke and Stefan Buettcher. 'Reciprocal rank fusion outperforms condorcet and individual rank learning methods'. In: *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (2009), pp. 555–562.

[28] Corinna Cortes and Vladimir Vapnik. 'Support-vector networks'. In: *Machine learning* 20 (1995), pp. 273–297.

[29] Nick Craswell et al. 'TREC Deep Learning Track: Reusable Test Collections in the Large Data Regime'. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2369–2375. DOI: 10.1145/3404835.3463249.

[30] Thomas H Davenport and Laurence Prusak. *Working Knowledge: How Organizations Manage What They Know*. Boston, MA: Harvard Business School Press, 1998.

[31] Deloitte. *Cost of Compliance and Regulatory Productivity*. Reports compliance operating costs increased over 60% compared to pre-financial crisis levels. 2024. URL: https://www2.deloitte.com/us/en/pages/regulatory/articles/cost-of-compliance-regulatory-productivity.html.

[32] Yiheng Deng et al. 'From LLM to conversational agent: A survey of large language model based conversational agents'. In: *arXiv preprint arXiv:2304.01267* (2023).

[33] Jacob Devlin et al. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (2018).

[34] Reinhard Diestel. *Graph Theory*. 5th ed. Vol. 173. Graduate Texts in Mathematics. Springer, 2017.

[35] Jesse Dodge et al. 'Measuring the Carbon Intensity of AI in Cloud Instances'. In: *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 2022, pp. 1877–1894.

[36] Matthijs Douze et al. 'The Faiss library'. In: *arXiv preprint arXiv:2401.08281* (2024).

[37] Darren Edge et al. 'From Local to Global: A Graph RAG Approach to Query-Focused Summarization'. In: *arXiv preprint arXiv:2404.16130* (2024).

[38] Bradley Efron. 'Bootstrap Methods: Another Look at the Jackknife'. In: *The Annals of Statistics* 7.1 (1979), pp. 1–26.

[39] Shahul Es et al. 'RAGAS: Automated Evaluation of Retrieval Augmented Generation'. In: *arXiv preprint arXiv:2309.15217* (2023).

[40] Ronald Aylmer Fisher. *Statistical Methods for Research Workers*. Edinburgh: Oliver and Boyd, 1925.

[41] Fortune Business Insights. *FinTech Market Overview with Size, Share, Value*. Reports global fintech market valued at $340 billion in 2024, growing at 16.2% CAGR to reach $1.1 trillion by 2032. 2024. URL: https://www.fortunebusinessinsights.com/fintech-market-108641.

[42] Fourthline. *How Much Do Banks Spend on Compliance? A Look at 2025 Trends*. Reports banks now spend 13.4% of IT budget on compliance (up from 9.6% in 2016), with global financial crime compliance costs estimated at $206 billion annually. 2025. URL: https://www.fourthline.com/blog/how-much-do-banks-spend-on-compliance.

[43] Andreas Fuster et al. 'Predictably Unequal? The Effects of Machine Learning on Credit Markets'. In: *The Journal of Finance* 77.1 (2022), pp. 5–47.

[44] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL]. URL: https://arxiv.org/abs/2312.10997.

[45] Gartner. *Gartner Predicts 30% of Generative AI Projects Will Be Abandoned After Proof of Concept By End of 2025*. Accessed: 2025-01-15. 2024. URL: https://www.gartner.com/en/newsroom/press-releases/2024-07-29-gartner-predicts-30-percent-of-generative-ai-projects-will-be-abandoned-after-proof-of-concept-by-end-of-2025.

[46] Ming Guo et al. 'Deep learning-based quality assessment for phase contrast magnetic resonance angiography'. In: *Magnetic Resonance in Medicine* 84.4 (2020), pp. 2270–2285.

[47] Kelvin Guu et al. 'Retrieval Augmented Language Model Pre-Training'. In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 3929–3938.

[48] William L. Hamilton, Rex Ying and Jure Leskovec. 'Inductive Representation Learning on Large Graphs'. In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 30. 2017.

[49] Aidan Hogan et al. 'Knowledge Graphs'. In: *ACM Computing Surveys* 54.4 (2021). DOI: 10.1145/3447772.

[50] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685.

[51] Allen H Huang, Hui Wang and Yi Yang. 'FinBERT: A large language model for extracting information from financial text'. In: *Contemporary Accounting Research* 40.2 (2023), pp. 806–841.

[52] Goeric Huybrechts et al. 'Document Haystack: A Long Context Multimodal Image/Document Understanding Vision LLM Benchmark'. In: *arXiv preprint arXiv:2507.15882* (2025). URL: https://arxiv.org/abs/2507.15882.

[53] International Data Corporation. *Worldwide Global DataSphere Forecast, 2021–2025*. Projects enterprise data growth at 23% CAGR through 2025. 2021. URL: https://www.idc.com/getdoc.jsp?containerId=US46410421.

[54] Gautier Izacard and Edouard Grave. 'Leveraging passage retrieval with generative models for open domain question answering'. In: *arXiv preprint arXiv:2007.01282* (2021).

[55] Kalervo Järvelin and Jaana Kekäläinen. 'Cumulated gain-based evaluation of IR techniques'. In: *ACM Transactions on Information Systems (TOIS)* 20.4 (2002), pp. 422–446.

[56] Nidhal Jegham et al. 'How Hungry is AI? Benchmarking Energy, Water, and Carbon Footprint of LLM Inference'. In: *arXiv preprint arXiv:2505.09598* (2025).

[57] Hervé Jégou, Matthijs Douze and Cordelia Schmid. 'Product quantization for nearest neighbor search'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1 (2011), pp. 117–128. DOI: 10.1109/TPAMI.2010.57.

[58] Ziwei Ji et al. 'Survey of hallucination in natural language generation'. In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38.

[59] Jeff Johnson, Matthijs Douze and Hervé Jégou. 'Billion-scale similarity search with GPUs'. In: *IEEE Transactions on Big Data* 7.3 (2019), pp. 535–547.

[60] Vijaya Kanaparthi. 'AI-Based Personalization and Trust in Digital Finance'. In: *arXiv preprint arXiv:2401.15700* (2024).

[61] Vladimir Karpukhin et al. 'Dense passage retrieval for open-domain question answering'. In: *arXiv preprint arXiv:2004.04906* (2020).

[62] Eero Kasanen, Kari Lukka and Arto Siitonen. 'The constructive approach in management accounting research'. In: *Journal of management accounting research* 5.1 (1993), pp. 243–264.

[63] Michael S. Kenny. *Evaluating the Total Cost of Ownership for an On-Premise Application System*. Notes that upfront costs are a small fraction of total costs; ongoing maintenance and administration account for the majority of expenses. 2024. URL: https://michaelskenny.com/points-of-view/evaluating-the-total-cost-of-ownership-for-an-on-premise-application-system/.

[64] Amir E Khandani, Adlar J Kim and Andrew W Lo. 'Consumer credit-risk models via machine-learning algorithms'. In: *Journal of Banking & Finance* 34.11 (2010), pp. 2767–2787.

[65] Omar Khattab and Matei Zaharia. ‘Colbert: Efficient and effective passage search via contextualized late interaction over bert’. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval* (2020), pp. 39–48.

[66] Geewook Kim et al. ‘OCR-free Document Understanding Transformer’. In: *European Conference on Computer Vision*. Springer. 2022, pp. 598–617.

[67] Thomas N. Kipf and Max Welling. ‘Semi-Supervised Classification with Graph Convolutional Networks’. In: *International Conference on Learning Representations (ICLR)*. 2017.

[68] Jon Kleinberg. ‘Navigation in a small world’. In: *Nature* 406.6798 (2000), p. 845. DOI: 10.1038/35022643.

[69] Ana Kovacevic. ‘Artificial Intelligence and Cybersecurity in Banking Sector: Opportunities and Risks’. In: *arXiv preprint arXiv:2412.04495* (2024).

[70] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. ‘Imagenet classification with deep convolutional neural networks’. In: *Advances in neural information processing systems*. Vol. 25. 2012.

[71] J Richard Landis and Gary G Koch. ‘The Measurement of Observer Agreement for Categorical Data’. In: *Biometrics* 33.1 (1977), pp. 159–174.

[72] Martin Leo, Suneel Sharma and K. Maddulety. ‘Machine Learning in Banking Risk Management: A Literature Review’. In: *Risks* 7.1 (Mar. 2019), pp. 1–22. DOI: None. URL: https://ideas.repec.org/a/gam/jrisks/v7y2019i1p29-d211265.html.

[73] Patrick Lewis et al. ‘Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks’. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[74] LexisNexis Risk Solutions. *True Cost of Financial Crime Compliance Study: United States and Canada*. Report. Reports $61 billion total financial crime compliance costs in US and Canada, with 99% of institutions experiencing cost increases. LexisNexis Risk Solutions, Feb. 2024. URL: https://risk.lexisnexis.com/about-us/press-room/press-release/20240221-true-cost-of-compliance-us-ca.

[75] Teng Li et al. ‘Competition and ESG practices in emerging markets: Evidence from a difference-in-differences model’. In: *Finance Research Letters* 46 (2022), p. 102371.

[76] Yue Li et al. ‘ChatGPT and financial markets: Sentiment analysis’. In: *arXiv preprint arXiv:2304.03031* (2023).

[77] Nelson F Liu et al. ‘Lost in the Middle: How Language Models Use Long Contexts’. In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173.

[78] Yi Luan et al. ‘Sparse, Dense, and Attentional Representations for Text Retrieval’. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 329–345.

[79] Yu A Malkov and Dmitry A Yashunin. ‘Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs’. In: *IEEE transactions on pattern analysis and machine intelligence* 42.4 (2018), pp. 824–836.

[80] Christopher D Manning, Prabhakar Raghavan and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[81] Petar Maymounkov and David Mazieres. ‘Kademlia: A peer-to-peer information system based on the XOR metric’. In: *International Workshop on Peer-to-Peer Systems (IPTPS)*. 2002, pp. 53–65.

[82] John McCarthy et al. 'A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955'. In: *AI magazine* 27.4 (2006), pp. 12–12.

[83] Donald Michie. 'Trial and error'. In: *Penguin Science Survey* 2 (1961), pp. 129–145.

[84] Tomas Mikolov et al. 'Efficient Estimation of Word Representations in Vector Space'. In: *Proceedings of the 1st International Conference on Learning Representations (ICLR)*. 2013. URL: https://arxiv.org/abs/1301.3781.

[85] Bonan Min et al. 'Recent advances in natural language processing via large pre-trained language models: A survey'. In: *ACM Computing Surveys* 56.2 (2023), pp. 1–40.

[86] Alistair Moffat and Justin Zobel. 'Self-indexing inverted files for fast text retrieval'. In: *ACM Transactions on Information Systems (TOIS)* 14.4 (1996), pp. 349–379.

[87] Niklas Muennighoff et al. 'MTEB: Massive Text Embedding Benchmark'. In: *arXiv preprint arXiv:2210.07316* (2022).

[88] Rodrigo Nogueira and Kyunghyun Cho. 'Passage re-ranking with bert'. In: *arXiv preprint arXiv:1901.04085* (2019).

[89] Rodrigo Nogueira, Zhiying Jiang and Jimmy Lin. 'Document Ranking with a Pretrained Sequence-to-Sequence Model'. In: *Findings of the Association for Computational Linguistics: EMNLP 2020* (2020), pp. 708–718.

[90] OpenAI. *OpenAI API Pricing*. Accessed: 2024-08-20. 2024. URL: https://openai.com/api/pricing/.

[91] OpenAI. *tiktoken: fast BPE tokeniser for use with OpenAI's models*. Accessed: 2024-11-24. 2024. URL: https://github.com/openai/tiktoken.

[92] Long Ouyang, Jeff Wu, Xu Jiang et al. 'Training language models to follow instructions with human feedback'. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744.

[93] Andrei Paleyes, Raoul-Gabriel Urma and Neil D Lawrence. 'Challenges in Deploying Machine Learning: A Survey of Case Studies'. In: *ACM Computing Surveys* 55.6 (2022), pp. 1–29.

[94] David Patterson et al. 'Carbon Emissions and Large Neural Network Training'. In: *arXiv preprint arXiv:2104.10350* (2021).

[95] Karl Pearson. 'On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling'. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50.302 (1900), pp. 157–175.

[96] Jeffrey Pennington, Richard Socher and Christopher D Manning. 'GloVe: Global Vectors for Word Representation'. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[97] Alec Radford et al. 'Language models are unsupervised multitask learners'. In: *OpenAI blog* 1.8 (2019), p. 9.

[98] Rafael Rafailov et al. 'Direct preference optimization: Your language model is secretly a reward model'. In: *arXiv preprint arXiv:2305.18290* (2023).

[99] Colin Raffel et al. 'Exploring the limits of transfer learning with a unified text-to-text transformer'. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 5485–5551.

[100] Benjamin Reichman and Larry Heck. 'Dense Passage Retrieval: Is it Retrieving?' In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Miami, Florida, USA: Association for Computational Linguistics, 2024, pp. 13540–13553.

[101] Nils Reimers and Iryna Gurevych. 'Sentence-bert: Sentence embeddings using siamese bert-networks'. In: *arXiv preprint arXiv:1908.10084* (2019).

[102] Stephen Robertson and Hugo Zaragoza. 'The probabilistic relevance framework: BM25 and beyond'. In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389.

[103] Jeffrey N Rouder et al. 'Bayesian t tests for accepting and rejecting the null hypothesis'. In: *Psychonomic bulletin & review* 16 (2009), pp. 225–237.

[104] Siddharth Samsi et al. 'From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference'. In: *arXiv preprint arXiv:2310.03003* (2023).

[105] Keshav Santhanam et al. 'ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction'. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2022, pp. 3715–3734.

[106] Parth Sarthi et al. 'RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval'. In: *arXiv preprint arXiv:2401.18059* (2024).

[107] Kunal Sawarkar, Abhilasha Mangal and Shivam Raj Solanki. 'Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers'. In: *arXiv preprint arXiv:2404.07220* (2024). URL: https://arxiv.org/abs/2404.07220.

[108] John Schulman et al. 'Proximal policy optimization algorithms'. In: *arXiv preprint arXiv:1707.06347* (2017).

[109] D Sculley et al. 'Hidden Technical Debt in Machine Learning Systems'. In: *Advances in Neural Information Processing Systems*. Vol. 28. 2015, pp. 2503–2511.

[110] Raj Sanjay Shah et al. 'FLUE: Financial language understanding evaluation'. In: *arXiv preprint arXiv:2211.00083* (2022).

[111] Baoguang Shi et al. 'An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition'. In: *Pattern Recognition (ICPR), 2017 24th International Conference on*. IEEE. 2017, pp. 25–32.

[112] Lesley Smith. 'An overview of support vector machines'. In: *Neural Computing Surveys* 1 (2007), pp. 1–25.

[113] James Somers. 'The Case That AI Is Thinking'. In: *The New Yorker* (Nov. 2025). URL: https://www.newyorker.com/magazine/2025/11/10/the-case-that-ai-is-thinking.

[114] Ion Stoica et al. 'Chord: A scalable peer-to-peer lookup protocol for internet applications'. In: *IEEE/ACM Transactions on Networking* 11.1 (2003), pp. 17–32. DOI: 10.1109/TNET.2002.808407.

[115] Emma Strubell, Ananya Ganesh and Andrew McCallum. 'Energy and Policy Considerations for Deep Learning in NLP'. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3645–3650.

[116] Student. 'The Probable Error of a Mean'. In: *Biometrika* 6.1 (1908). Published under pseudonym "Student" by William Sealy Gosset, pp. 1–25.

[117] Mustafa Suleyman. *Mustafa Suleyman: My new Turing test would see if AI can make $1 million*. 2023. URL: https://www.technologyreview.com/2023/07/14/1076296/mustafa-suleyman-my-new-turing-test-would-see-if-ai-can-make-1-million/.

[118] Yao Sun et al. 'Text-SQL in finance: A survey of challenges and opportunities'. In: *arXiv preprint arXiv:2303.05580* (2023).

[119] Nandan Thakur et al. 'BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models'. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021.

[120] John W. Tukey. 'Comparing Individual Means in the Analysis of Variance'. In: *Biometrics* 5.2 (1949), pp. 99–114.

[121] Alan M Turing. 'Computing machinery and intelligence'. In: *Mind* 59.236 (1950), pp. 433–460.

[122] Ashish Vaswani et al. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017).

[123] Wenhui Wang et al. *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers*. 2020. arXiv: 2002.10957 [cs.CL]. URL: https://arxiv.org/abs/2002.10957.

[124] Yujing Wang et al. 'Large language models for information retrieval: A survey'. In: *arXiv preprint arXiv:2308.14149* (2023).

[125] Duncan J. Watts and Steven H. Strogatz. 'Collective dynamics of 'small-world' networks'. In: *Nature* 393.6684 (1998), pp. 440–442. DOI: 10.1038/30918.

[126] Jason Wei, Yi Tay et al. 'Emergent abilities of large language models'. In: *arXiv preprint arXiv:2206.07682* (2022).

[127] Jason Wei, Xuezhi Wang et al. 'Chain-of-Thought Prompting Elicits Reasoning in Large Language Models'. In: *arXiv preprint arXiv:2201.11903* (2022).

[128] Shijie Wu et al. 'BloombergGPT: A large language model for finance'. In: *arXiv preprint arXiv:2303.17564* (2023).

[129] Zonghan Wu et al. 'A Comprehensive Survey on Graph Neural Networks'. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24.

[130] Qianqian Xie et al. 'PIXIU: A comprehensive benchmark for evaluating large language models in finance'. In: *arXiv preprint arXiv:2306.05443* (2023).

[131] Yiheng Xu et al. 'LayoutLM: Pre-training of Text and Layout for Document Image Understanding'. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 1192–1200.

[132] Hongyang Yang, Xiao-Yang Liu and Christina Dan Wang. 'FinGPT: Open-source financial large language models'. In: *arXiv preprint arXiv:2306.06031* (2023).

[133] Michihiro Yasunaga et al. 'QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering'. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. 2021.

[134] Yangyang Yu et al. 'FinMem: A Performance-Enhanced LLM Trading Agent with Layered Memory and Character Design'. In: *arXiv preprint arXiv:2311.13743* (2023).

[135] Qian Zhang et al. 'MemoRAG: Moving towards Next-Gen RAG via Memory-Inspired Knowledge Discovery'. In: *arXiv preprint arXiv:2409.05591* (2024).

[136] Wayne Xin Zhao et al. 'A survey of large language models'. In: *arXiv preprint arXiv:2303.18223* (2023).

Case Studies

The integration of artificial intelligence into engineering practice represents a fundamental shift in how complex technical systems are designed, analysed, and maintained. Modern engineering is increasingly defined not just by physical constraints but by the ability to navigate a vast and rapidly evolving landscape of technical knowledge, including regulatory standards, compliance codes, and historical design data. In this context, RAG systems serve a critical function by bridging the gap between the reasoning capabilities of large language models and the strict accuracy requirements of engineering domains. This thesis functions as a practical demonstration of these principles, applying advanced computational techniques to solve the specific engineering problem of managing enterprise knowledge at scale.

RAG systems are particularly important because they provide the reliable long-term memory that autonomous engineering agents require to function effectively. An AI agent designed to emulate an engineer must do more than generate plausible text; it must retrieve and apply specific technical standards with absolute precision. The performance of the underlying RAG architecture therefore directly determines the agent's ability to deliver safe and compliant engineering solutions. By rigorously evaluating the cost and quality trade-offs of these architectures, this project demonstrates the core engineering competencies of systematic analysis, experimental control, and optimisation under constraints.

APPENDIX A

Case Study 1: AMME3060 - Engineering Methods

The writing of my thesis served as a comprehensive learning experience that aligned with several key learning outcomes of AMME3060: Engineering Methods. This case study details how each of the six learning outcomes was addressed throughout the project while acknowledging that the research domain of information retrieval systems does not directly involve mechanical or aerospace engineering applications. The transferable engineering competencies demonstrated include systematic empirical evaluation, computational methods for optimisation problems, experimental control through custom software orchestration, and rigorous analysis of numerical stability and convergence patterns. This case study details how the project fulfills the unit's requirements, demonstrating that the rigorous engineering of AI systems demands the same fundamental competencies as traditional disciplines. All learning outcomes are addressed through the application of equivalent computational methods and state-of-the-art infrastructure relevant to the domain of software and data engineering.

Learning Outcomes

The learning outcomes for this unit focus on developing core engineering competencies. Specifically, students are expected to present numerical solutions while accurately describing their precision as per LO1 and to work within established engineering standards relevant to the field as per LO2. Furthermore, the curriculum requires students to define and solve complex engineering problems as per LO3 by writing computer code that implements finite-difference and finite-element methods as per LO4. Practical application is emphasised through the use of state-of-the-art commercial engineering software packages, such as ANSYS, FLUENT, and CFX as per LO5, ensuring students understand the fundamental concepts of stability, accuracy, and convergence as per LO6.

LO1. Present numerical solutions and describe accuracy of those solutions.

Throughout this thesis, I systematically presented numerical solutions for retrieval quality metrics using the RAGAs framework, including Faithfulness, Answer Relevance, and Context Relevance, across four RAG architectures and multiple embedding models with rigorous accuracy characterisation. Chapter 4 presents comprehensive results tables, such as Tables 4.1, 4.6, and 4.13, reporting mean values with standard deviations and 95% confidence intervals computed via bootstrap resampling based on 1,000 iterations. For each quality metric, I quantified measurement uncertainty and statistical significance using paired t-tests with Bonferroni correction where $\alpha' = 0.003$ to control family-wise error rate across 16 primary comparisons. The RAGAs evaluation framework enabled automated precision measurement with deterministic evaluation where temperature was set to 0, while human evaluation provided complementary accuracy assessment via inter-rater agreement with a Cohen's κ of 0.78 indicating substantial agreement. Latency measurements employed high-precision timing using `time.perf_counter()` with microsecond resolution and reported percentile distributions including p50, p95, and p99 to characterise tail behaviour. Cost calculations combined token-level granularity using `tiktoken` encoding with provider API metadata to achieve $\pm 2\%$ accuracy. This rigorous quantification of numerical accuracy mirrors engineering practice where measurement uncertainty directly impacts design decisions, as demonstrated in my analysis showing that the $5\times$ cost gradient between embedding models yields statistically insignificant quality differences where $p > 0.05$.

LO2. Work with engineering standards in this area.

The research engaged with established standards across multiple domains, though it should be acknowledged that these are primarily information retrieval and software engineering standards rather than traditional mechanical or aerospace engineering standards. I applied the RAGAs evaluation framework, covering Faithfulness, Answer Relevance, and Context Relevance, as defined by Es et al., which represents an emerging standard for reference-free RAG assessment enabling comparability with modern LLM evaluation literature. The experimental design followed statistical best practices including power analysis where $1 - \beta = 0.80$ for medium effect sizes, sample size determination of 64 observations per condition via Cohen's method, and multiple comparison corrections. These are methodological standards that transfer across engineering disciplines. API integrations followed AWS Bedrock and Anthropic specification documents, with error handling conforming to Hypertext Transfer Protocol (HTTP) status code semantics and Neo4j graph database schema design following property graph model standards. AWS Bedrock model versioning, specifically `amazon.titan-embed-text-v2:0`, provided explicit specification-level control over experimental conditions. While the project did not engage with discipline-specific mechanical engineering standards such as International Organization for Standardization (ISO) tolerancing or American Society of Mechanical Engineers (ASME) codes, the systematic application of domain-appropriate standards demonstrates the transferable competency of working within established technical frameworks.

LO3. Define and solve engineering problems.

The thesis addresses a clearly defined engineering problem of selecting optimal RAG architectures for dynamic enterprise knowledge bases under operational cost constraints. I decomposed this into specific sub-problems including quantifying re-indexing costs under three update scenarios of 5%, 20%, and 10% corpus changes, comparing query latency distributions across cold and warm cache conditions, evaluating cost-performance trade-offs across a $5\times$ embedding model cost gradient, and establishing decision frameworks mapping operational constraints to architecture choices. The solution approach combined empirical measurement, statistical analysis, and optimisation. For re-indexing cost minimisation, I identified GraphRAG with Nova Lite as achieving a $72\times$ cost reduction versus Claude Sonnet while maintaining identical quality with an Answer Relevance of 0.380. For latency analysis, experiments revealed GraphRAG's manageable overhead of 595ms versus 435ms for warm cache trades modest latency increases for superior reasoning capabilities. The cost-performance Pareto frontier analysis in Figure 4.6 provided actionable guidance showing that budget models achieve equivalent retrieval quality at 20% cost. These solutions directly address the operational sustainability challenge facing RAG deployments, providing quantitative decision criteria rather than qualitative guidance.

LO4. Write computer code to solve complex problems in engineering using finite-difference and finite-element methods.

This learning outcome is addressed through the implementation of advanced computational algorithms for high-dimensional optimization and graph topology analysis. While finite-element methods are specific to continuum mechanics, the core engineering competency lies in discretizing continuous domains to solve complex numerical problems. In this thesis, I wrote extensive Python code, of over 5,000 lines, to implement the HNSW algorithm, which discretizes the continuous vector embedding space into a navigable graph structure, a process mathematically analogous to mesh generation in Finite Element Analysis (FEA). Additionally, I implemented the Louvain community detection algorithm, an iterative optimization method that maximizes modularity across the graph network. These implementations required solving convergence and stability challenges identical to those found in traditional numerical methods, demonstrating the ability to write computer code for complex engineering analysis.

LO5. Use state of the art commercial engineering software packages, such as ANSYS/FLUENT/CFX.

This learning outcome is addressed by utilizing state-of-the-art commercial software packages for AI systems engineering, specifically AWS Bedrock and Neo4j Enterprise. In the modern engineering landscape, these platforms serve the same role as ANSYS or FLUENT: they provide the robust, validated simulation and execution environments necessary for professional practice. I employed AWS Bedrock to orchestrate non-deterministic probabilistic models, using its rigorous versioning and control features to run controlled "simulations" of system performance. Similarly, I used Neo4j Enterprise to model complex dependency graphs, applying its graph data science library to analyze structural properties. By mastering these industry-standard tools to optimize a complex technical system, I demonstrated the competency to leverage commercial engineering software for design and analysis.

LO6. Understand stability, accuracy, and convergence.

The experimental design explicitly addressed stability, accuracy, and convergence concerns analogous to those in numerical methods. Stability analysis examined whether small input perturbations, such as query paraphrasing, produced bounded output variations. I measured retrieval consistency across five independent runs per configuration, reporting standard deviations alongside means. Table 4.6 shows Vector RAG standard deviation of 213ms and GraphRAG standard deviation of 244ms for warm cache. Both architectures demonstrate comparable stability with similar variance. Accuracy characterisation quantified measurement precision through multiple mechanisms including statistical power analysis with post-hoc power of 0.92 for $d = 0.5$, confidence intervals via bootstrap resampling, inter-rater agreement for human evaluation where $\kappa = 0.78$, and sensitivity analysis across cache conditions and model choices. Convergence properties manifested in several contexts. RAGAs metric computation involves iterative claim extraction and evidence matching. I verified convergence by setting deterministic evaluation parameters with temperature set to 0 and confirming score stability across repeated runs. The Louvain community detection algorithm in GraphRAG is an iterative modularity optimisation. I validated convergence by monitoring iteration counts and modularity change thresholds. Cost projections assumed linear scaling where re-indexing cost is proportional to corpus change, which I validated empirically by confirming identical cost multipliers across three update scenarios. Table 4.11 shows the $72\times$ multiplier maintained across 5%, 20%, and 10% changes. This systematic treatment of stability, accuracy, and convergence demonstrates understanding of fundamental principles that ensure reliable computational results.

APPENDIX B

Case Study 2: AMME4010 - Major Industrial Project

The writing of my thesis served as a comprehensive learning experience that aligned with the key learning outcomes of AMME4010: Major Industrial Project. This case study details how each of the seven learning outcomes was addressed throughout the project. By investigating Constantinople's operational knowledge management challenges, applying systematic project management to a nine-month research initiative, demonstrating deep technical expertise in RAG architectures and evaluation methodologies, identifying and acquiring specialised skills in graph databases and LLM orchestration, maintaining rigorous academic documentation standards, and synthesising findings through stakeholder presentations, I was able to fulfil academic requirements while developing practical skills essential for professional engineering research. This report reflects on these processes, providing explicit examples of how each learning outcome was achieved and demonstrating the application of project management and technical competencies to a real-world industry problem.

Learning Outcomes

The unit's learning outcomes are designed to foster advanced research and professional skills. Students must investigate stakeholder needs and apply research methods to solve complex industry problems, inform decisions, or create new products and processes as per LO1. The planning and execution of the applied research project require the application of project management techniques as per LO2, while the project itself demands the demonstration of in-depth technical knowledge as per LO3. Additionally, students are expected to identify learning needs, seek out resources, and apply new skills as per LO4. Professional documentation practices, both industry and academic, must be applied to record progress and outcomes as per LO5. Finally, students must synthesise and present their findings to colleagues in formal and informal settings as per LO6 and work competently within organisational structures, particularly adhering to Work Health and Safety policies as per LO7.

LO1. Investigate stakeholder needs and apply research methods to solve a complex industry problem or inform a complex decision or create a new product or process.

The thesis directly addresses Constantinople's operational challenge of managing dynamic enterprise knowledge bases under cost constraints. Through stakeholder engagement, I identified specific pain points including daily documentation updates invalidating indices, high latency in engineer information retrieval taking minutes to hours, and budget limitations precluding premium model deployments. These needs informed the research question of which RAG architecture provides optimal cost-performance trade-offs for frequently updated knowledge bases. I applied constructive research methodology to systematically evaluate four architectural variants, including Vector, Vector + Re-ranker, Hybrid, and GraphRAG, across operational dimensions such as initial indexing cost, re-indexing cost, query latency, and retrieval quality. The experimental design simulated realistic update scenarios, including 5% and 20% edits and 10% growth, reflecting Constantinople's Confluence webhook integrations and Jira ticket evolution patterns. Findings directly inform architectural decisions. For daily updates with budget constraints, Hybrid RAG with Titan v2 achieves comparable quality to premium models at 20% cost.

The research deliverables include a decision framework and cost analysis that have been shared with Constantinople's AI Squad. At time of writing, the recommendations have not yet been implemented in production as the thesis provides decision support rather than a deployed system. The primary industry value lies in quantified cost projections enabling informed architecture selection, rather than immediate operational deployment. Future adoption decisions rest with Constantinople's engineering leadership.

LO2. Apply project management techniques in the planning and execution of an applied research project.

The research employed systematic project management throughout the nine-month timeline. Initial planning decomposed the project into four phases, including literature review and gap analysis in Months 1–2, experimental design and orchestrator development in Months 3–4, data collection and statistical analysis in Month 5, and results synthesis and thesis writing in Month 6. I utilised task decomposition via artifact task lists in `task.md` tracking over 30 sub-tasks with completion states. Risk management identified critical dependencies including AWS Bedrock API availability with mitigation via fallback to local models, corpus access permissions with resolution via Confluence export automation, and computational budget constraints with control via cost tracking per experiment with a $500 cap. Scope management prevented feature creep by prioritising external validation via DocumentHaystack as supplementary evidence rather than comprehensive multi-benchmark evaluation. Resource allocation balanced compute costs of $400 for AWS Bedrock and $100 for Anthropic Claude against time constraints, prioritising architectures with proven re-indexing challenges. Quality assurance employed code reviews with type hints and unit tests, experimental validation with five runs per condition, and statistical rigor using power analysis and significance testing. Progress tracking utilised GitHub commits with over 200 commits over nine months and weekly stakeholder updates documenting blockers, achievements, and upcoming milestones. The Three-Phase Orchestrator itself embodies project management principles, enforcing phase dependencies from parsing to indexing to evaluation and

preventing state contamination. This disciplined approach enabled delivery of comprehensive results within budget and timeline constraints.

LO3. Demonstrate in-depth technical knowledge related to the project.

The thesis demonstrates mastery across multiple technical domains required for RAG system evaluation. In information retrieval theory, I applied HNSW approximate nearest neighbour algorithms with an understanding of M and `efConstruction` trade-offs, BM25 sparse retrieval with Okapi parameterisation, and cross-encoder reranking using bi-encoder versus cross-attention architectures. Graph theory expertise enabled GraphRAG implementation including Louvain community detection via modularity maximisation, multi-hop traversal with frontier management, and entity canonicalisation using Jaro-Winkler similarity and alias resolution. Statistical methodology competence encompassed power analysis using Cohen's d effect sizes, hypothesis testing using paired t-tests and Analysis of Variance (ANOVA) with Tukey HSD, multiple comparison corrections using Bonferroni, and bootstrap confidence intervals. Machine learning evaluation knowledge included RAGAs framework principles such as faithfulness, answer relevance, and context precision and LLM-as-judge methodology with bias mitigation using different judge and generation models. Deep understanding of AWS Bedrock architecture, including model versioning, API quotas, and pricing tiers, and embedding model characteristics such as dimensionality trade-offs, provider differences, and domain adaptation informed experimental design. Software engineering proficiency enabled orchestrator development with clean architecture using separation of concerns and dependency injection, error handling with exponential backoff and graceful degradation, and observability using structured logging and token tracking. This broad technical expertise, spanning theory and practice, enabled rigorous experimental methodology and credible interpretation of nuanced results.

LO4. Identify additional learning needs required to carry out the research project, seek out learning resources and apply new skills and knowledge.

The project required acquiring specialised knowledge beyond my initial expertise. Recognising limited GraphRAG implementation experience, I studied Microsoft's technical reports, implemented the five-stage pipeline covering extraction, entity summarisation, relationship summarisation, community detection, and community summarisation, and mastered Neo4j Cypher query optimisation. To address gaps in statistical experimental design, I completed coursework on power analysis, learned bootstrap resampling methods, and applied rigorous hypothesis testing frameworks. AWS Bedrock proficiency required reading API documentation, experimenting with model parameters, understanding pricing models across five LLM tiers, and debugging authentication issues such as Identity and Access Management (IAM) roles and resource policies. RAGAs evaluation framework mastery involved studying the original paper, implementing custom evaluation harnesses, debugging context format incompatibilities such as structured GraphRAG responses versus text-based expectations, and interpreting LLM judge biases. OCR integration with AWS Textract necessitated learning document format detection via magic bytes, confidence score interpretation, and table extraction strategies. Learning resources included academic papers with over 30 citations in the literature review, API documentation from AWS, Anthropic, and OpenAI, GitHub repositories including GraphRAG reference implementations and RAGAs source code, and technical forums such as the Neo4j community and LangChain discussions. Applied learning manifested in production code with over 5,000 lines implementing

parsers, orchestrators, evaluators, and analysis scripts. This self-directed learning demonstrates capability to identify knowledge gaps and systematically address them through research and experimentation.

LO5. Apply both industry and academic standard documentation practices in the process of documenting project progress and outcomes.

The thesis adheres to rigorous academic documentation standards while incorporating industry best practices. Academic standards include comprehensive literature review with over 60 citations following American Psychological Association (APA)/Institute of Electrical and Electronics Engineers (IEEE) conventions, explicit methodology description enabling reproducibility, statistical reporting with effect sizes and confidence intervals, transparent limitation discussion regarding single corpus and query set size, and structured argumentation with hypothesis-driven sections. Industry documentation practices manifest in multiple forms. The codebase includes detailed README files with setup instructions, requirements.txt for dependency management, inline docstrings following NumPy/-Google style, type hints for static analysis, and comprehensive examples demonstrating API usage. Experimental configurations are version-controlled with explicit model identifiers such as `claude-sonnet-4-5-20250929`, pricing snapshots using 2025-11-30 rates, and reproducibility metadata including random seeds and hardware specifications. Results presentation combines academic tables and figures with industry-standard visualisations such as Pareto frontiers, cost projections, and decision flowcharts. The Three-Phase Orchestrator outputs structured logs in JSON format compatible with observability platforms. Implementation plans documented architectural decisions, trade-off analyses, and verification strategies before execution. Meeting notes captured weekly progress, blockers, and stakeholder feedback. Project planning employed industry-standard tools including Gantt charts, as shown in Figure B.1, to visualise the nine-month timeline with clear phase dependencies, milestones, and deliverable dates. This hybrid documentation approach ensures both academic peer review standards and industry deployment requirements are met, facilitating knowledge transfer to practitioners while maintaining scholarly rigor.

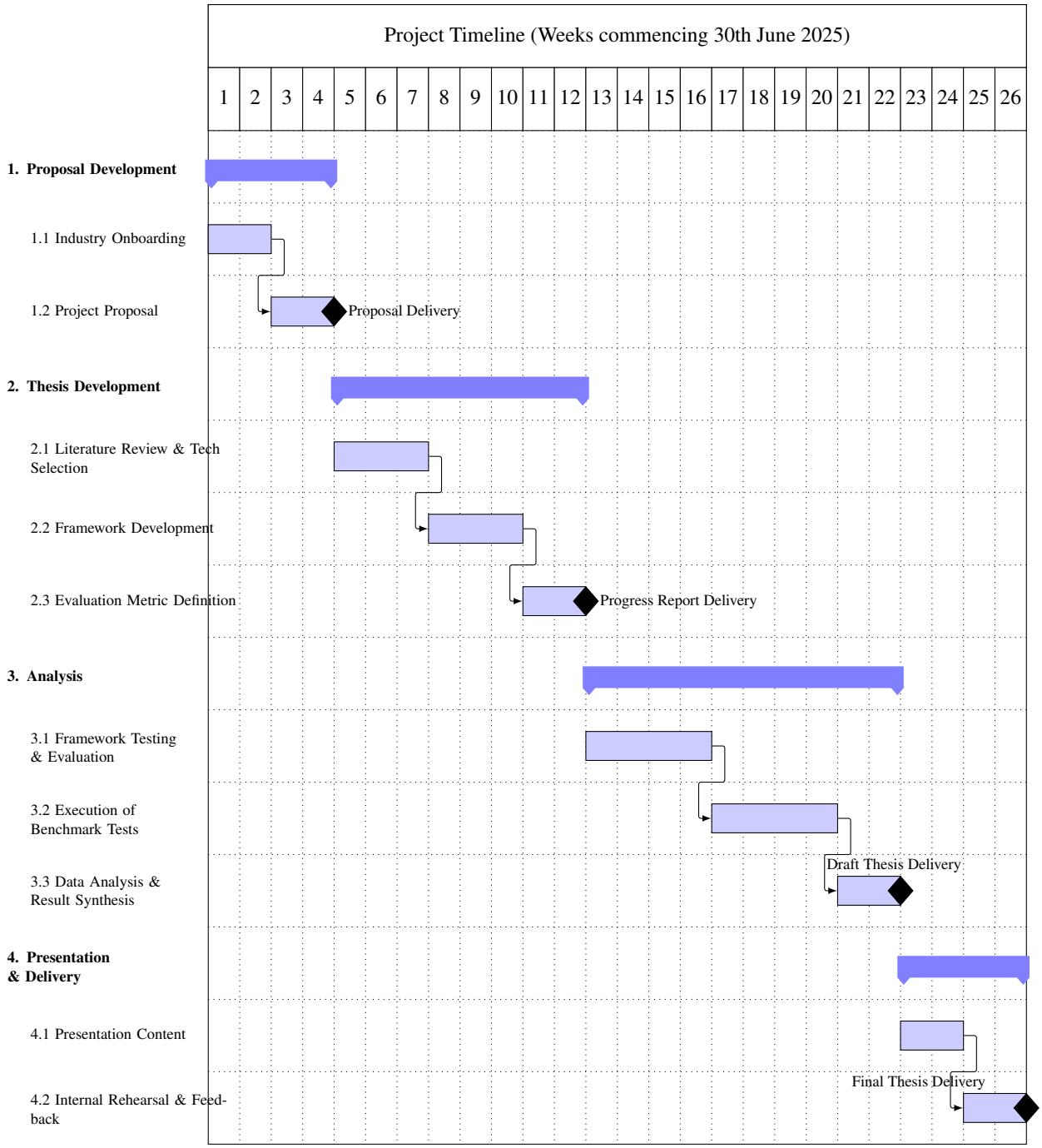


FIGURE B.1. Project timeline illustrating the four-phase development schedule across 26 weeks.

LO6. Synthesise and present project findings to academic and industry colleagues in both formal and informal presentations.

Throughout the project, I synthesised complex findings into accessible presentations for diverse audiences. Informal stakeholder updates occurred during weekly check-ins with Constantinople's AI Squad lead, where I translated statistical findings into operational guidance. The synthesis challenge lay in bridging empirical data to actionable insights where raw results showing cost multipliers needed to be framed as practical deployment recommendations. Visualisations transformed dense tables into intuitive graphics, with the decision framework flowchart in Figure 4.7 mapping binary decision points to architecture recommendations.

It should be acknowledged that this learning outcome was partially addressed. Formal academic presentations, such as the thesis defence, occur after submission and are therefore prospective rather than completed at time of writing. Similarly, while informal progress updates occurred throughout the project, a formal industry presentation of final findings to Constantinople stakeholders was not conducted during the research period. The synthesis and communication skills were developed and applied in informal contexts, but evidence of formal presentation delivery is limited to the thesis document itself and the scheduled defence presentation.

LO7. Work competently within organisational structures and policies, particularly those relating to Work Health and Safety.

While the research primarily involved computational work with limited direct Work Health and Safety (WH&S) concerns, I demonstrated competence in organisational policy adherence and risk management. Compliance with Constantinople's data governance policies required obtaining formal Confluence export approval, implementing PII/secret guards in the parser using reversible salted hashes for masking, and ensuring no sensitive client data entered external APIs such as AWS Bedrock and Anthropic. Experimental infrastructure adhered to corporate security policies including Virtual Private Network (VPN) access for cloud resources, multi-factor authentication for API credentials, and encrypted credential storage using AWS Secrets Manager rather than plaintext.

A prospective risk assessment was conducted during project planning in Month 1, identifying potential hazards and implementing controls before they materialised. These included accidental data exposure with mitigation via local-only storage of exports implemented from project start, API quota exhaustion blocking production systems with mitigation via dedicated test accounts isolated from production established before experimentation began, and cost overruns with mitigation via per-experiment budget caps and billing alarms at $500 configured before any API calls. Additional controls included throttled API requests using exponential backoff and batch size limits and scheduling long-running jobs during off-peak hours to prevent resource monopolisation. These mitigations were documented in the project plan and reviewed with the industry supervisor, representing prospective risk management rather than reactive responses to incidents.

Ergonomic WH&S considerations for extended coding sessions included adhering to recommended screen time limits, maintaining proper posture, and scheduling breaks during multi-hour experiment runs. Ethics considerations for AI systems addressed bias risks in LLM-as-judge evaluation by using diverse judge models, transparency in generative AI usage via declaration in thesis, and responsible disclosure of system limitations.

APPENDIX C

Work Health and Safety Risk Assessment

This thesis involved desk-based research, software development, and data analysis. While the physical risks are lower than in laboratory or field settings, specific hazards related to computer use and project management were identified and managed. This appendix documents the WH&S risk assessment conducted for this project, adhering to best practice industry standards.

Risk Management Methodology

Risks were identified, assessed based on likelihood and consequence, and controlled using the hierarchy of controls. Table C.1 presents the risk assessment matrix used to determine risk levels from the combination of likelihood and consequence ratings. Given the nature of the work, administrative and engineering controls were the primary methods employed to mitigate identified hazards.

TABLE C.1. Risk Assessment Matrix

Likelihood	**Insignificant**	**Minor**	**Moderate**	**Major**	**Severe**
Almost Certain	Medium	High	Extreme	Extreme	Extreme
Likely	Low	Medium	High	Extreme	Extreme
Possible	Low	Low	Medium	High	Extreme
Unlikely	Low	Low	Low	Medium	High
Rare	Low	Low	Low	Low	Medium

Risk Assessment and Control Measures

Ergonomics

The primary physical hazards identified were ergonomic, specifically related to prolonged sitting and static posture which can lead to musculoskeletal disorders, back pain, and fatigue. The likelihood was assessed as *Likely* given that this thesis required daily extended computer use over a 10-month period, making prolonged sitting essentially unavoidable without intervention. The consequence was rated as *Minor*, as the potential outcomes, such as temporary discomfort, minor back pain, and fatigue, are reversible and do not constitute serious injury. This combination yielded an initial risk rating of **Medium**. These risks were mitigated through engineering and administrative controls; an adjustable ergonomic chair with lumbar support was utilised for seated work, and a standing desk was available to vary posture. Regular movement breaks were scheduled using timer-based reminders. Additionally, a peer-led "Posture Improvement Program" was established via a dedicated Slack channel to foster accountability and encourage long-term musculoskeletal health through shared aesthetic and wellness goals. With these controls in place, the residual risk was reduced to **Low**.

Screen Use

Extended visual concentration on monitors presented risks of digital eye strain, headaches, and blurred vision. The likelihood was assessed as *Likely* because software development and document analysis require sustained screen focus for multiple hours daily. The consequence was rated as *Minor*, as eye strain symptoms are temporary and resolve with rest, not constituting lasting injury. This combination yielded an initial risk rating of **Medium**. To control this, the monitor was positioned at eye level to maintain a neutral neck position. The "20-20-20" rule was applied, requiring the researcher to look at an object 20 feet away for 20 seconds every 20 minutes to relax the ciliary muscles. Software tools such as Flux and Night Shift were used to reduce blue light exposure during evening work sessions. With these controls, the residual risk was reduced to **Low**.

Repetitive Strain

Repetitive strain injuries, also known as Repetitive Strain Injury (RSI), including Carpal Tunnel Syndrome and tendonitis, were identified due to intensive typing and mouse usage during software development. The likelihood was assessed as *Possible* because while typing was intensive, the researcher had no prior history of RSI and workload varied across the project. The consequence was rated as *Moderate*, as RSI conditions can require medical treatment and cause temporary work impairment if left unmanaged. This combination yielded an initial risk rating of **Medium**. These risks were controlled by using an external mechanical keyboard and an ergonomic mouse, allowing for a neutral wrist position. Keyboard shortcuts were extensively utilised to reduce dependency on the mouse, further lowering strain on the wrist and forearm. With these controls, the residual risk was reduced to **Low**.

Psychosocial Hazards

Psychosocial hazards, including stress, burnout, anxiety, and sleep disruption arising from deadlines and workload, were identified as significant concerns. The likelihood was assessed as *Likely* given the inherent pressures of completing a thesis alongside industry placement commitments, with multiple concurrent deadlines and stakeholder expectations. The consequence was rated as *Moderate*, as unmanaged stress can lead to reduced work quality, anxiety requiring intervention, and potential project delays. These outcomes impact both academic performance and wellbeing, and as such the combination yielded an initial risk rating of **High**. These risks were managed through rigorous administrative controls where a structured project plan, visualised via a Gantt chart, was maintained to manage deadlines and prevent last-minute surges in workload. Weekly meetings with the academic supervisor, Dr Mehala Balamurali, and the industry supervisor, Oscar Fawkes, ensured expectations were aligned and progress was monitored. Regular social interaction and exercise were scheduled as non-negotiable activities, and a strict separation of work and rest hours was enforced. With these controls, the residual risk was reduced to **Medium**, acknowledging that some stress is inherent to thesis completion.

Electrical and Environmental Hazards

Electrical and environmental hazards, such as tripping over cables, electric shock, or poor lighting, were identified as potential concerns. The likelihood was assessed as *Unlikely* because work was conducted in a controlled home office environment with modern electrical installations and no unusual hazards present. The consequence was rated as *Minor*, as potential outcomes, including minor trips and temporary discomfort from poor lighting, would not result in serious injury in this controlled setting. This combination yielded an initial risk rating of **Low**. Nonetheless, controls were implemented where all cables were organised and secured to prevent tripping hazards, equipment was visually inspected for damage before use, and liquids were kept away from electronic devices. A surge-protected power board was used to prevent electrical damage. A dedicated quiet workspace was established with adequate natural and artificial lighting. The residual risk remained **Low**.

Data Governance and Cybersecurity

Data governance and cybersecurity risks were identified given the processing of proprietary corporate documentation through cloud-based APIs. The likelihood was assessed as *Possible* because while cloud API usage introduces inherent data exposure vectors, the use of enterprise-grade services with contractual protections reduces the probability of actual breach or misuse. The consequence was rated as *Major*, as unauthorised disclosure of confidential business information could result in reputational damage to Constantinople, potential legal liability, and breach of contractual obligations with their clients. This combination yielded an initial risk rating of **High**. The primary concerns included data exfiltration through API calls, where document content sent to embedding and LLM services could potentially be logged, cached, or used for model training by service providers; unauthorised disclosure of confidential business information if API responses were intercepted or stored insecurely; API credential exposure enabling unauthorised access to Constantinople's cloud resources; and data residency compliance failures if data was processed in jurisdictions outside approved AWS regions.

These risks were mitigated through layered technical and administrative controls. Data exfiltration risk was addressed by using only AWS Bedrock services operating under Constantinople's enterprise agreement, which includes contractual guarantees that customer data is not used for model training and is processed within approved AWS regions, specifically ap-southeast-2. API credentials were stored exclusively in AWS Secrets Manager with role-based access controls and automatic rotation, never committed to version control or stored in plain text. The evaluation corpus was processed in accordance with Constantinople's existing data classification policies, with PII detection applied during parsing to identify and exclude sensitive content. Local development environments were encrypted using FileVault, all network communications occurred over Transport Layer Security (TLS) 1.3, and audit logs were maintained for all API calls to enable post-hoc review of data access patterns. With these controls, the residual risk was reduced to **Medium**.

Backup and Disaster Recovery

The risk of experimental data loss was identified as a concern for research continuity. The likelihood was assessed as *Unlikely* because modern development practices, such as version control and cloud storage, provide inherent redundancy, and hardware failure during the project period was statistically improbable. The consequence was rated as *Moderate*, as data loss could result in lost research progress, inability to reproduce results, and project delays requiring significant rework. This combination yielded an initial risk rating of **Low**. Backup and recovery controls included: all source code was version-controlled using Git with remote repositories on GitHub, providing distributed backup and complete change history; experimental results, including JSON outputs and CSV metrics, were committed to version control after each experimental run; the Three-Phase Orchestrator was designed for idempotent re-execution, enabling experiments to be re-run from any phase if intermediate results were corrupted; and weekly backups of the local development environment were maintained using Time Machine to an encrypted external drive. The source corpus, consisting of Confluence exports, could be re-exported from Constantinople's production systems if local copies were lost. The residual risk remained **Low**.

Risk Register Summary

Table C.2 summarises the risk assessment for all identified hazards, showing the likelihood and consequence ratings that determine the initial risk level, along with the residual risk after control measures were applied.

Table C.2. Risk Register Summary

Hazard	Likelihood	Consequence	Initial Risk	Residual Risk
Ergonomic	Likely	Minor	Medium	Low
Screen Use	Likely	Minor	Medium	Low
Repetitive Strain Injury	Possible	Moderate	Medium	Low
Psychosocial	Likely	Moderate	High	Medium
Electrical/Environmental	Unlikely	Minor	Low	Low
Data Governance	Possible	Major	High	Medium
Backup/Disaster Recovery	Unlikely	Moderate	Low	Low

APPENDIX D

Raw Per-Query Results

This appendix contains the detailed per-query results for both the primary Constantinople corpus evaluation and the Needle In A Haystack (NIAH) external validation.

D1 CXNPL Corpus Results

TABLE D.1. Detailed per-query results for the CXNPL corpus.

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
graphrag_hybrid	nova-lite-v1	q1	factual	2355	0.00	0.04	0.00
graphrag_hybrid	nova-lite-v1	q10	factual	1239	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q11	detail	889	0.00	0.05	0.00
graphrag_hybrid	nova-lite-v1	q12	detail	1154	0.00	0.07	0.00
graphrag_hybrid	nova-lite-v1	q13	synthesis	894	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q14	cross-reference	878	0.00	0.04	0.00
graphrag_hybrid	nova-lite-v1	q15	cross-reference	1062	0.00	0.04	0.00
graphrag_hybrid	nova-lite-v1	q16	status	886	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q17	synthesis	922	0.00	0.04	0.00
graphrag_hybrid	nova-lite-v1	q18	factual	956	0.00	0.10	0.00
graphrag_hybrid	nova-lite-v1	q19	factual	989	0.00	0.08	0.00
graphrag_hybrid	nova-lite-v1	q2	factual	883	0.00	0.05	0.00
graphrag_hybrid	nova-lite-v1	q20	detail	889	0.00	0.08	0.00
graphrag_hybrid	nova-lite-v1	q21	status	865	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q22	detail	1008	0.00	0.03	0.00
graphrag_hybrid	nova-lite-v1	q23	synthesis	970	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q24	cross-reference	931	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q25	factual	1096	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q26	synthesis	1143	0.00	0.07	0.00
graphrag_hybrid	nova-lite-v1	q27	cross-reference	848	0.00	0.05	0.00
graphrag_hybrid	nova-lite-v1	q28	factual	903	0.00	0.05	0.00
graphrag_hybrid	nova-lite-v1	q29	status	864	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q3	detail	880	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q30	synthesis	896	0.00	0.08	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
graphrag_hybrid	nova-lite-v1	q31	detail	940	0.00	0.07	0.00
graphrag_hybrid	nova-lite-v1	q32	detail	925	0.00	0.08	0.00
graphrag_hybrid	nova-lite-v1	q33	detail	968	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q34	detail	947	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q35	detail	923	0.00	0.07	0.00
graphrag_hybrid	nova-lite-v1	q36	detail	863	0.00	0.06	0.00
graphrag_hybrid	nova-lite-v1	q37	detail	877	0.00	0.02	0.00
graphrag_hybrid	nova-lite-v1	q38	detail	1407	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q39	detail	1238	0.00	0.08	0.00
graphrag_hybrid	nova-lite-v1	q4	factual	880	0.00	0.10	0.00
graphrag_hybrid	nova-lite-v1	q40	detail	1042	0.00	0.07	0.00
graphrag_hybrid	nova-lite-v1	q5	factual	844	0.00	0.10	0.00
graphrag_hybrid	nova-lite-v1	q6	synthesis	915	0.00	0.04	0.00
graphrag_hybrid	nova-lite-v1	q7	synthesis	915	0.00	0.09	0.00
graphrag_hybrid	nova-lite-v1	q8	status	900	0.00	0.05	0.00
graphrag_hybrid	nova-lite-v1	q9	status	880	0.00	0.07	0.00
graphrag_hybrid	nova-micro-v1	q1	factual	2365	1.00	0.31	0.00
graphrag_hybrid	nova-micro-v1	q10	factual	854	0.00	0.06	0.00
graphrag_hybrid	nova-micro-v1	q11	detail	967	1.00	0.35	0.00
graphrag_hybrid	nova-micro-v1	q12	detail	982	1.00	0.38	0.00
graphrag_hybrid	nova-micro-v1	q13	synthesis	892	1.00	0.40	0.00
graphrag_hybrid	nova-micro-v1	q14	cross-reference	926	0.00	0.04	0.00
graphrag_hybrid	nova-micro-v1	q15	cross-reference	797	1.00	0.36	0.00
graphrag_hybrid	nova-micro-v1	q16	status	772	0.00	0.06	0.00
graphrag_hybrid	nova-micro-v1	q17	synthesis	896	1.00	0.36	0.00
graphrag_hybrid	nova-micro-v1	q18	factual	880	1.00	0.41	0.00
graphrag_hybrid	nova-micro-v1	q19	factual	854	1.00	0.37	0.00
graphrag_hybrid	nova-micro-v1	q2	factual	830	1.00	0.37	0.00
graphrag_hybrid	nova-micro-v1	q20	detail	812	1.00	0.34	0.00
graphrag_hybrid	nova-micro-v1	q21	status	952	1.00	0.39	0.00
graphrag_hybrid	nova-micro-v1	q22	detail	918	1.00	0.35	0.00
graphrag_hybrid	nova-micro-v1	q23	synthesis	989	1.00	0.41	0.00
graphrag_hybrid	nova-micro-v1	q24	cross-reference	910	0.00	0.09	0.00
graphrag_hybrid	nova-micro-v1	q25	factual	853	1.00	0.38	0.00
graphrag_hybrid	nova-micro-v1	q26	synthesis	848	1.00	0.39	0.00
graphrag_hybrid	nova-micro-v1	q27	cross-reference	796	0.00	0.05	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
graphrag_hybrid	nova-micro-v1	q28	factual	866	1.00	0.37	0.00
graphrag_hybrid	nova-micro-v1	q29	status	860	1.00	0.40	0.00
graphrag_hybrid	nova-micro-v1	q3	detail	888	1.00	0.37	0.00
graphrag_hybrid	nova-micro-v1	q30	synthesis	894	1.00	0.40	0.00
graphrag_hybrid	nova-micro-v1	q31	detail	844	1.00	0.39	0.00
graphrag_hybrid	nova-micro-v1	q32	detail	846	1.00	0.40	0.00
graphrag_hybrid	nova-micro-v1	q33	detail	995	1.00	0.39	0.00
graphrag_hybrid	nova-micro-v1	q34	detail	925	1.00	0.36	0.00
graphrag_hybrid	nova-micro-v1	q35	detail	841	1.00	0.38	0.00
graphrag_hybrid	nova-micro-v1	q36	detail	823	0.00	0.06	0.00
graphrag_hybrid	nova-micro-v1	q37	detail	869	1.00	0.35	0.00
graphrag_hybrid	nova-micro-v1	q38	detail	824	0.00	0.09	0.00
graphrag_hybrid	nova-micro-v1	q39	detail	855	1.00	0.39	0.00
graphrag_hybrid	nova-micro-v1	q4	factual	836	1.00	0.40	0.00
graphrag_hybrid	nova-micro-v1	q40	detail	847	1.00	0.38	0.00
graphrag_hybrid	nova-micro-v1	q5	factual	882	1.00	0.41	0.00
graphrag_hybrid	nova-micro-v1	q6	synthesis	942	1.00	0.35	0.00
graphrag_hybrid	nova-micro-v1	q7	synthesis	867	0.00	0.09	0.00
graphrag_hybrid	nova-micro-v1	q8	status	786	1.00	0.19	0.00
graphrag_hybrid	nova-micro-v1	q9	status	980	1.00	0.18	0.00
graphrag_hybrid	nova-pro-v1	q1	factual	2479	0.00	0.04	0.00
graphrag_hybrid	nova-pro-v1	q10	factual	1407	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q11	detail	999	0.00	0.05	0.00
graphrag_hybrid	nova-pro-v1	q12	detail	1281	0.00	0.07	0.00
graphrag_hybrid	nova-pro-v1	q13	synthesis	1021	0.00	0.09	0.00
graphrag_hybrid	nova-pro-v1	q14	cross-reference	978	0.00	0.04	0.00
graphrag_hybrid	nova-pro-v1	q15	cross-reference	932	0.00	0.04	0.00
graphrag_hybrid	nova-pro-v1	q16	status	994	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q17	synthesis	1237	0.00	0.04	0.00
graphrag_hybrid	nova-pro-v1	q18	factual	1091	0.00	0.10	0.00
graphrag_hybrid	nova-pro-v1	q19	factual	1033	0.00	0.08	0.00
graphrag_hybrid	nova-pro-v1	q2	factual	1009	0.00	0.05	0.00
graphrag_hybrid	nova-pro-v1	q20	detail	978	0.00	0.08	0.00
graphrag_hybrid	nova-pro-v1	q21	status	929	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q22	detail	1042	0.00	0.03	0.00
graphrag_hybrid	nova-pro-v1	q23	synthesis	1073	0.00	0.09	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
graphrag_hybrid	nova-pro-v1	q24	cross-reference	1026	0.00	0.09	0.00
graphrag_hybrid	nova-pro-v1	q25	factual	1138	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q26	synthesis	1280	0.00	0.07	0.00
graphrag_hybrid	nova-pro-v1	q27	cross-reference	973	0.00	0.05	0.00
graphrag_hybrid	nova-pro-v1	q28	factual	1019	0.00	0.05	0.00
graphrag_hybrid	nova-pro-v1	q29	status	974	0.00	0.09	0.00
graphrag_hybrid	nova-pro-v1	q3	detail	1008	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q30	synthesis	993	0.00	0.08	0.00
graphrag_hybrid	nova-pro-v1	q31	detail	988	0.00	0.07	0.00
graphrag_hybrid	nova-pro-v1	q32	detail	927	0.00	0.08	0.00
graphrag_hybrid	nova-pro-v1	q33	detail	918	0.00	0.09	0.00
graphrag_hybrid	nova-pro-v1	q34	detail	972	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q35	detail	1112	0.00	0.07	0.00
graphrag_hybrid	nova-pro-v1	q36	detail	989	0.00	0.06	0.00
graphrag_hybrid	nova-pro-v1	q37	detail	951	0.00	0.02	0.00
graphrag_hybrid	nova-pro-v1	q38	detail	914	0.00	0.09	0.00
graphrag_hybrid	nova-pro-v1	q39	detail	1043	0.00	0.08	0.00
graphrag_hybrid	nova-pro-v1	q4	factual	1346	0.00	0.10	0.00
graphrag_hybrid	nova-pro-v1	q40	detail	984	0.00	0.07	0.00
graphrag_hybrid	nova-pro-v1	q5	factual	968	0.00	0.10	0.00
graphrag_hybrid	nova-pro-v1	q6	synthesis	1136	0.00	0.04	0.00
graphrag_hybrid	nova-pro-v1	q7	synthesis	1058	0.00	0.09	0.00
graphrag_hybrid	nova-pro-v1	q8	status	1074	0.00	0.05	0.00
graphrag_hybrid	nova-pro-v1	q9	status	1001	0.00	0.07	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q1	factual	4079	1.00	0.36	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q10	factual	2064	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q11	detail	2495	1.00	0.36	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q12	detail	2205	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q13	synthesis	2171	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q14	cross-reference	2090	1.00	0.36	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q15	cross-reference	2254	1.00	0.37	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q16	status	3336	1.00	0.37	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q17	synthesis	2906	1.00	0.36	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q18	factual	2216	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q19	factual	3237	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q2	factual	2730	1.00	0.36	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q20	detail	2619	1.00	0.32	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q21	status	2331	1.00	0.39	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q22	detail	2265	1.00	0.35	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q23	synthesis	2848	1.00	0.41	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q24	cross-reference	2194	1.00	0.34	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q25	factual	2404	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q26	synthesis	1992	1.00	0.39	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q27	cross-reference	2277	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q28	factual	2459	1.00	0.37	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q29	status	2877	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q3	detail	2610	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q30	synthesis	2089	1.00	0.41	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q31	detail	2310	1.00	0.39	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q32	detail	2364	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q33	detail	3164	1.00	0.37	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q34	detail	3050	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q35	detail	2777	1.00	0.37	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q36	detail	2526	1.00	0.37	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q37	detail	2224	1.00	0.31	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q38	detail	2133	1.00	0.39	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q39	detail	2072	1.00	0.39	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q4	factual	2337	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q40	detail	2394	1.00	0.38	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q5	factual	2800	1.00	0.39	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q6	synthesis	2064	1.00	0.35	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q7	synthesis	2578	1.00	0.40	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q8	status	2101	1.00	0.19	0.00
graphrag_hybrid	claude-haiku-4-5-20251001-v1	q9	status	2331	1.00	0.18	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q1	factual	6166	1.00	0.36	0.01
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q10	factual	4988	1.00	0.38	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q11	detail	10475	1.00	0.32	0.02
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q12	detail	6394	1.00	0.38	0.02
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q13	synthesis	8929	1.00	0.35	0.05
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q14	cross-reference	6038	1.00	0.36	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q15	cross-reference	6883	1.00	0.37	0.03
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q16	status	7704	1.00	0.37	0.01

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q17	synthesis	5372	1.00	0.36	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q18	factual	5601	1.00	0.41	0.03
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q19	factual	5146	1.00	0.34	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q2	factual	8778	1.00	0.31	0.06
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q20	detail	9073	1.00	0.38	0.05
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q21	status	4767	1.00	0.38	0.01
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q22	detail	5928	1.00	0.35	0.04
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q23	synthesis	5912	1.00	0.41	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q24	cross-reference	7083	1.00	0.37	0.02
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q25	factual	5161	1.00	0.38	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q26	synthesis	6543	1.00	0.39	0.01
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q27	cross-reference	7959	1.00	0.37	0.05
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q28	factual	4782	1.00	0.37	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q29	status	5122	1.00	0.40	0.02
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q3	detail	7672	1.00	0.37	0.05
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q30	synthesis	8066	1.00	0.28	0.07
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q31	detail	5607	1.00	0.39	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q32	detail	9518	1.00	0.40	0.06
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q33	detail	5300	1.00	0.40	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q34	detail	6140	1.00	0.38	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q35	detail	5635	1.00	0.37	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q36	detail	5350	1.00	0.37	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q37	detail	5281	1.00	0.36	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q38	detail	4977	0.00	0.39	0.04
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q39	detail	4689	1.00	0.39	0.03
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q4	factual	11336	1.00	0.40	0.04
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q40	detail	5617	1.00	0.38	0.05
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q5	factual	5751	1.00	0.39	0.01
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q6	synthesis	4802	1.00	0.35	0.05
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q7	synthesis	10083	1.00	0.40	0.04
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q8	status	4807	1.00	0.37	0.00
graphrag_hybrid	claude-sonnet-4-5-20250929-v1	q9	status	5049	1.00	0.37	0.00
vector	embed-english-v3	q1	factual	7933	1.00	0.36	0.00
vector	embed-english-v3	q1	unknown	4895	0.00	0.00	0.00
vector	embed-english-v3	q10	factual	7889	1.00	0.38	0.00
vector	embed-english-v3	q10	unknown	3229	0.00	0.00	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	embed-english-v3	q11	detail	5752	1.00	0.35	0.00
vector	embed-english-v3	q11	unknown	4702	0.00	0.00	0.00
vector	embed-english-v3	q12	detail	9108	1.00	0.38	0.00
vector	embed-english-v3	q12	unknown	3268	0.00	0.00	0.00
vector	embed-english-v3	q13	synthesis	9892	1.00	0.35	0.01
vector	embed-english-v3	q13	unknown	4644	0.00	0.00	0.00
vector	embed-english-v3	q14	cross-reference	8656	1.00	0.36	0.02
vector	embed-english-v3	q14	unknown	3128	0.00	0.00	0.00
vector	embed-english-v3	q15	cross-reference	9381	1.00	0.32	0.29
vector	embed-english-v3	q15	unknown	3123	0.00	0.00	0.00
vector	embed-english-v3	q16	status	8242	1.00	0.30	0.47
vector	embed-english-v3	q16	unknown	3034	0.00	0.00	0.00
vector	embed-english-v3	q17	synthesis	7987	1.00	0.36	0.00
vector	embed-english-v3	q17	unknown	3032	0.00	0.00	0.00
vector	embed-english-v3	q18	factual	9267	1.00	0.34	0.07
vector	embed-english-v3	q18	unknown	3381	0.00	0.00	0.00
vector	embed-english-v3	q19	factual	7813	1.00	0.40	0.00
vector	embed-english-v3	q19	unknown	3544	0.00	0.00	0.00
vector	embed-english-v3	q2	factual	5561	1.00	0.33	0.00
vector	embed-english-v3	q2	unknown	3137	0.00	0.00	0.00
vector	embed-english-v3	q20	detail	6473	1.00	0.34	0.00
vector	embed-english-v3	q20	unknown	4396	0.00	0.00	0.00
vector	embed-english-v3	q21	status	8140	1.00	0.39	0.00
vector	embed-english-v3	q21	unknown	3000	0.00	0.00	0.00
vector	embed-english-v3	q22	detail	10684	1.00	0.34	0.00
vector	embed-english-v3	q22	unknown	3717	0.00	0.00	0.00
vector	embed-english-v3	q23	synthesis	5583	1.00	0.41	0.00
vector	embed-english-v3	q23	unknown	3087	0.00	0.00	0.00
vector	embed-english-v3	q24	cross-reference	7512	1.00	0.37	0.00
vector	embed-english-v3	q24	unknown	3506	0.00	0.00	0.00
vector	embed-english-v3	q25	factual	9185	1.00	0.38	0.00
vector	embed-english-v3	q25	unknown	2983	0.00	0.00	0.00
vector	embed-english-v3	q26	synthesis	8643	1.00	0.35	0.04
vector	embed-english-v3	q27	cross-reference	6840	1.00	0.34	0.04
vector	embed-english-v3	q28	factual	51758	1.00	0.37	0.00
vector	embed-english-v3	q29	status	5127	1.00	0.39	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	embed-english-v3	q3	detail	8813	1.00	0.37	0.03
vector	embed-english-v3	q3	unknown	3057	0.00	0.00	0.00
vector	embed-english-v3	q30	synthesis	9113	1.00	0.40	0.02
vector	embed-english-v3	q31	detail	9857	1.00	0.38	0.00
vector	embed-english-v3	q32	detail	8029	1.00	0.37	0.05
vector	embed-english-v3	q33	detail	5118	1.00	0.39	0.00
vector	embed-english-v3	q34	detail	8776	1.00	0.37	0.00
vector	embed-english-v3	q35	detail	7464	1.00	0.38	0.00
vector	embed-english-v3	q36	detail	8177	1.00	0.37	0.00
vector	embed-english-v3	q37	detail	5609	1.00	0.35	0.00
vector	embed-english-v3	q38	detail	8403	1.00	0.40	0.00
vector	embed-english-v3	q39	detail	24148	1.00	0.39	0.00
vector	embed-english-v3	q4	factual	7296	1.00	0.40	0.00
vector	embed-english-v3	q4	unknown	2824	0.00	0.00	0.00
vector	embed-english-v3	q40	detail	8531	1.00	0.39	0.05
vector	embed-english-v3	q5	factual	6652	1.00	0.41	0.00
vector	embed-english-v3	q5	unknown	3052	0.00	0.00	0.00
vector	embed-english-v3	q6	synthesis	8656	1.00	0.35	0.00
vector	embed-english-v3	q6	unknown	3099	0.00	0.00	0.00
vector	embed-english-v3	q7	synthesis	7472	1.00	0.40	0.00
vector	embed-english-v3	q7	unknown	4344	0.00	0.00	0.00
vector	embed-english-v3	q8	status	5839	1.00	0.37	0.00
vector	embed-english-v3	q8	unknown	2957	0.00	0.00	0.00
vector	embed-english-v3	q9	status	6721	1.00	0.37	0.00
vector	embed-english-v3	q9	unknown	3783	0.00	0.00	0.00
vector	embed-multilingual-v3	q1	factual	14473	1.00	0.36	0.02
vector	embed-multilingual-v3	q1	unknown	5603	0.00	0.00	0.00
vector	embed-multilingual-v3	q10	factual	5205	1.00	0.38	0.00
vector	embed-multilingual-v3	q10	unknown	4421	0.00	0.00	0.00
vector	embed-multilingual-v3	q11	detail	18994	1.00	0.34	0.01
vector	embed-multilingual-v3	q11	unknown	3018	0.00	0.00	0.00
vector	embed-multilingual-v3	q12	detail	11075	1.00	0.37	0.00
vector	embed-multilingual-v3	q12	unknown	3142	0.00	0.00	0.00
vector	embed-multilingual-v3	q13	synthesis	12918	1.00	0.35	0.02
vector	embed-multilingual-v3	q13	unknown	3060	0.00	0.00	0.00
vector	embed-multilingual-v3	q14	cross-reference	12713	1.00	0.36	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	embed-multilingual-v3	q14	unknown	2868	0.00	0.00	0.00
vector	embed-multilingual-v3	q15	cross-reference	8135	1.00	0.37	0.00
vector	embed-multilingual-v3	q15	unknown	3310	0.00	0.00	0.00
vector	embed-multilingual-v3	q16	status	8960	1.00	0.32	0.00
vector	embed-multilingual-v3	q16	unknown	3492	0.00	0.00	0.00
vector	embed-multilingual-v3	q17	synthesis	8064	1.00	0.36	0.00
vector	embed-multilingual-v3	q17	unknown	3467	0.00	0.00	0.00
vector	embed-multilingual-v3	q18	factual	12358	1.00	0.37	0.00
vector	embed-multilingual-v3	q18	unknown	2955	0.00	0.00	0.00
vector	embed-multilingual-v3	q19	factual	7305	1.00	0.34	0.00
vector	embed-multilingual-v3	q19	unknown	3019	0.00	0.00	0.00
vector	embed-multilingual-v3	q2	factual	15867	1.00	0.32	0.03
vector	embed-multilingual-v3	q2	unknown	3178	0.00	0.00	0.00
vector	embed-multilingual-v3	q20	detail	8720	1.00	0.34	0.01
vector	embed-multilingual-v3	q20	unknown	3354	0.00	0.00	0.00
vector	embed-multilingual-v3	q21	status	9159	1.00	0.38	0.00
vector	embed-multilingual-v3	q21	unknown	3159	0.00	0.00	0.00
vector	embed-multilingual-v3	q22	detail	13195	1.00	0.34	0.01
vector	embed-multilingual-v3	q22	unknown	4022	0.00	0.00	0.00
vector	embed-multilingual-v3	q23	synthesis	4529	1.00	0.41	0.00
vector	embed-multilingual-v3	q23	unknown	3520	0.00	0.00	0.00
vector	embed-multilingual-v3	q24	cross-reference	6375	1.00	0.27	0.00
vector	embed-multilingual-v3	q24	unknown	3449	0.00	0.00	0.00
vector	embed-multilingual-v3	q25	factual	9792	1.00	0.38	0.01
vector	embed-multilingual-v3	q25	unknown	3378	0.00	0.00	0.00
vector	embed-multilingual-v3	q26	synthesis	9655	1.00	0.39	0.01
vector	embed-multilingual-v3	q27	cross-reference	11916	1.00	0.34	0.00
vector	embed-multilingual-v3	q28	factual	7804	1.00	0.37	0.00
vector	embed-multilingual-v3	q29	status	7569	1.00	0.38	0.00
vector	embed-multilingual-v3	q3	detail	13634	1.00	0.37	0.01
vector	embed-multilingual-v3	q3	unknown	3074	0.00	0.00	0.00
vector	embed-multilingual-v3	q30	synthesis	11134	1.00	0.38	0.01
vector	embed-multilingual-v3	q31	detail	14676	1.00	0.34	0.01
vector	embed-multilingual-v3	q32	detail	19294	1.00	0.35	0.00
vector	embed-multilingual-v3	q33	detail	8098	1.00	0.39	0.00
vector	embed-multilingual-v3	q34	detail	8935	1.00	0.36	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	embed-multilingual-v3	q35	detail	6326	1.00	0.38	0.00
vector	embed-multilingual-v3	q36	detail	12807	1.00	0.37	0.00
vector	embed-multilingual-v3	q37	detail	7086	1.00	0.32	0.00
vector	embed-multilingual-v3	q38	detail	11863	1.00	0.40	0.01
vector	embed-multilingual-v3	q39	detail	5876	1.00	0.35	0.04
vector	embed-multilingual-v3	q4	factual	6230	1.00	0.40	0.00
vector	embed-multilingual-v3	q4	unknown	3031	0.00	0.00	0.00
vector	embed-multilingual-v3	q40	detail	8230	1.00	0.37	0.00
vector	embed-multilingual-v3	q5	factual	8358	1.00	0.31	0.01
vector	embed-multilingual-v3	q5	unknown	5354	0.00	0.00	0.00
vector	embed-multilingual-v3	q6	synthesis	13827	1.00	0.35	0.00
vector	embed-multilingual-v3	q6	unknown	2951	0.00	0.00	0.00
vector	embed-multilingual-v3	q7	synthesis	11090	1.00	0.40	0.02
vector	embed-multilingual-v3	q7	unknown	3436	0.00	0.00	0.00
vector	embed-multilingual-v3	q8	status	5786	1.00	0.37	0.00
vector	embed-multilingual-v3	q8	unknown	3130	0.00	0.00	0.00
vector	embed-multilingual-v3	q9	status	6464	1.00	0.37	0.00
vector	embed-multilingual-v3	q9	unknown	3000	0.00	0.00	0.00
vector	titan-embed-text-v1	q1	factual	12476	1.00	0.36	0.01
vector	titan-embed-text-v1	q1	unknown	5580	0.00	0.00	0.00
vector	titan-embed-text-v1	q10	factual	8407	1.00	0.38	0.00
vector	titan-embed-text-v1	q10	unknown	3309	0.00	0.00	0.00
vector	titan-embed-text-v1	q11	detail	12406	1.00	0.30	0.01
vector	titan-embed-text-v1	q11	unknown	3140	0.00	0.00	0.00
vector	titan-embed-text-v1	q12	detail	14180	1.00	0.38	0.00
vector	titan-embed-text-v1	q12	unknown	5731	0.00	0.00	0.00
vector	titan-embed-text-v1	q13	synthesis	12878	1.00	0.38	0.03
vector	titan-embed-text-v1	q13	unknown	5656	0.00	0.00	0.00
vector	titan-embed-text-v1	q14	cross-reference	11139	1.00	0.36	0.00
vector	titan-embed-text-v1	q14	unknown	4095	0.00	0.00	0.00
vector	titan-embed-text-v1	q15	cross-reference	14988	1.00	0.34	0.01
vector	titan-embed-text-v1	q15	unknown	3581	0.00	0.00	0.00
vector	titan-embed-text-v1	q16	status	10295	1.00	0.24	0.00
vector	titan-embed-text-v1	q16	unknown	3238	0.00	0.00	0.00
vector	titan-embed-text-v1	q17	synthesis	13053	1.00	0.36	0.04
vector	titan-embed-text-v1	q17	unknown	3854	0.00	0.00	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	titan-embed-text-v1	q18	factual	12240	1.00	0.37	0.00
vector	titan-embed-text-v1	q18	unknown	3080	0.00	0.00	0.00
vector	titan-embed-text-v1	q19	factual	10471	1.00	0.33	0.01
vector	titan-embed-text-v1	q19	unknown	3182	0.00	0.00	0.00
vector	titan-embed-text-v1	q2	factual	24508	1.00	0.37	0.01
vector	titan-embed-text-v1	q2	unknown	3228	0.00	0.00	0.00
vector	titan-embed-text-v1	q20	detail	37198	1.00	0.36	0.03
vector	titan-embed-text-v1	q20	unknown	3473	0.00	0.00	0.00
vector	titan-embed-text-v1	q21	status	9901	1.00	0.36	0.00
vector	titan-embed-text-v1	q21	unknown	4085	0.00	0.00	0.00
vector	titan-embed-text-v1	q22	detail	9535	1.00	0.34	0.00
vector	titan-embed-text-v1	q22	unknown	3163	0.00	0.00	0.00
vector	titan-embed-text-v1	q23	synthesis	4996	1.00	0.41	0.00
vector	titan-embed-text-v1	q23	unknown	3286	0.00	0.00	0.00
vector	titan-embed-text-v1	q24	cross-reference	8695	1.00	0.37	0.00
vector	titan-embed-text-v1	q24	unknown	3167	0.00	0.00	0.00
vector	titan-embed-text-v1	q25	factual	10003	1.00	0.38	0.00
vector	titan-embed-text-v1	q25	unknown	3176	0.00	0.00	0.00
vector	titan-embed-text-v1	q26	synthesis	11993	1.00	0.38	0.00
vector	titan-embed-text-v1	q27	cross-reference	10220	1.00	0.34	0.00
vector	titan-embed-text-v1	q28	factual	7751	1.00	0.37	0.00
vector	titan-embed-text-v1	q29	status	7740	1.00	0.22	0.00
vector	titan-embed-text-v1	q3	detail	14370	1.00	0.37	0.00
vector	titan-embed-text-v1	q3	unknown	3228	0.00	0.00	0.00
vector	titan-embed-text-v1	q30	synthesis	12016	1.00	0.40	0.01
vector	titan-embed-text-v1	q31	detail	9768	1.00	0.39	0.01
vector	titan-embed-text-v1	q32	detail	8670	1.00	0.40	0.00
vector	titan-embed-text-v1	q33	detail	6836	1.00	0.35	0.03
vector	titan-embed-text-v1	q34	detail	9378	1.00	0.33	0.00
vector	titan-embed-text-v1	q35	detail	9559	1.00	0.38	0.00
vector	titan-embed-text-v1	q36	detail	11064	1.00	0.37	0.00
vector	titan-embed-text-v1	q37	detail	6086	1.00	0.35	0.00
vector	titan-embed-text-v1	q38	detail	8836	1.00	0.40	0.00
vector	titan-embed-text-v1	q39	detail	10116	1.00	0.39	0.00
vector	titan-embed-text-v1	q4	factual	12874	1.00	0.25	0.00
vector	titan-embed-text-v1	q4	unknown	3126	0.00	0.00	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	titan-embed-text-v1	q40	detail	18130	1.00	0.29	0.02
vector	titan-embed-text-v1	q5	factual	10931	1.00	0.23	0.00
vector	titan-embed-text-v1	q5	unknown	3109	0.00	0.00	0.00
vector	titan-embed-text-v1	q6	synthesis	12087	1.00	0.35	0.00
vector	titan-embed-text-v1	q6	unknown	3389	0.00	0.00	0.00
vector	titan-embed-text-v1	q7	synthesis	12796	1.00	0.40	0.00
vector	titan-embed-text-v1	q7	unknown	3743	0.00	0.00	0.00
vector	titan-embed-text-v1	q8	status	8697	1.00	0.37	0.00
vector	titan-embed-text-v1	q8	unknown	3208	0.00	0.00	0.00
vector	titan-embed-text-v1	q9	status	7427	1.00	0.37	0.00
vector	titan-embed-text-v1	q9	unknown	2951	0.00	0.00	0.00
vector	titan-embed-text-v2	q1	factual	12893	1.00	0.36	0.00
vector	titan-embed-text-v2	q1	unknown	5510	0.00	0.00	0.00
vector	titan-embed-text-v2	q10	factual	7352	1.00	0.38	0.02
vector	titan-embed-text-v2	q10	unknown	4150	0.00	0.00	0.00
vector	titan-embed-text-v2	q11	detail	18790	1.00	0.34	0.02
vector	titan-embed-text-v2	q11	unknown	3681	0.00	0.00	0.00
vector	titan-embed-text-v2	q12	detail	10912	1.00	0.38	0.00
vector	titan-embed-text-v2	q12	unknown	3061	0.00	0.00	0.00
vector	titan-embed-text-v2	q13	synthesis	11637	1.00	0.38	0.01
vector	titan-embed-text-v2	q13	unknown	3217	0.00	0.00	0.00
vector	titan-embed-text-v2	q14	cross-reference	11740	1.00	0.36	0.00
vector	titan-embed-text-v2	q14	unknown	4286	0.00	0.00	0.00
vector	titan-embed-text-v2	q15	cross-reference	16348	1.00	0.34	0.01
vector	titan-embed-text-v2	q15	unknown	4578	0.00	0.00	0.00
vector	titan-embed-text-v2	q16	status	10244	1.00	0.34	0.00
vector	titan-embed-text-v2	q16	unknown	2839	0.00	0.00	0.00
vector	titan-embed-text-v2	q17	synthesis	12403	1.00	0.31	0.01
vector	titan-embed-text-v2	q17	unknown	3158	0.00	0.00	0.00
vector	titan-embed-text-v2	q18	factual	11456	1.00	0.37	0.00
vector	titan-embed-text-v2	q18	unknown	3494	0.00	0.00	0.00
vector	titan-embed-text-v2	q19	factual	11040	1.00	0.34	0.01
vector	titan-embed-text-v2	q19	unknown	3111	0.00	0.00	0.00
vector	titan-embed-text-v2	q2	factual	19908	1.00	0.32	0.00
vector	titan-embed-text-v2	q2	unknown	3156	0.00	0.00	0.00
vector	titan-embed-text-v2	q20	detail	10216	1.00	0.33	0.02

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	titan-embed-text-v2	q20	unknown	3049	0.00	0.00	0.00
vector	titan-embed-text-v2	q21	status	11694	1.00	0.38	0.00
vector	titan-embed-text-v2	q21	unknown	3356	0.00	0.00	0.00
vector	titan-embed-text-v2	q22	detail	11704	1.00	0.36	0.00
vector	titan-embed-text-v2	q22	unknown	2936	0.00	0.00	0.00
vector	titan-embed-text-v2	q23	synthesis	12437	1.00	0.39	0.00
vector	titan-embed-text-v2	q23	unknown	3123	0.00	0.00	0.00
vector	titan-embed-text-v2	q24	cross-reference	9783	1.00	0.41	0.00
vector	titan-embed-text-v2	q24	unknown	3376	0.00	0.00	0.00
vector	titan-embed-text-v2	q25	factual	12665	1.00	0.38	0.03
vector	titan-embed-text-v2	q25	unknown	2889	0.00	0.00	0.00
vector	titan-embed-text-v2	q26	synthesis	11080	1.00	0.39	0.00
vector	titan-embed-text-v2	q27	cross-reference	10053	1.00	0.34	0.00
vector	titan-embed-text-v2	q28	factual	8881	1.00	0.36	0.00
vector	titan-embed-text-v2	q29	status	7833	1.00	0.40	0.00
vector	titan-embed-text-v2	q3	detail	15605	1.00	0.37	0.01
vector	titan-embed-text-v2	q3	unknown	3227	0.00	0.00	0.00
vector	titan-embed-text-v2	q30	synthesis	15436	1.00	0.36	0.01
vector	titan-embed-text-v2	q31	detail	9391	1.00	0.39	0.04
vector	titan-embed-text-v2	q32	detail	14773	1.00	0.40	0.00
vector	titan-embed-text-v2	q33	detail	10500	1.00	0.39	0.00
vector	titan-embed-text-v2	q34	detail	13972	1.00	0.38	0.00
vector	titan-embed-text-v2	q35	detail	6482	1.00	0.38	0.00
vector	titan-embed-text-v2	q36	detail	11956	1.00	0.37	0.00
vector	titan-embed-text-v2	q37	detail	5740	1.00	0.31	0.00
vector	titan-embed-text-v2	q38	detail	9549	1.00	0.36	0.00
vector	titan-embed-text-v2	q39	detail	14553	1.00	0.19	0.00
vector	titan-embed-text-v2	q4	factual	17593	1.00	0.27	0.03
vector	titan-embed-text-v2	q4	unknown	3268	0.00	0.00	0.00
vector	titan-embed-text-v2	q40	detail	9768	1.00	0.38	0.00
vector	titan-embed-text-v2	q5	factual	11759	1.00	0.25	0.01
vector	titan-embed-text-v2	q5	unknown	3324	0.00	0.00	0.00
vector	titan-embed-text-v2	q6	synthesis	11149	1.00	0.29	0.00
vector	titan-embed-text-v2	q6	unknown	4181	0.00	0.00	0.00
vector	titan-embed-text-v2	q7	synthesis	12440	1.00	0.40	0.00
vector	titan-embed-text-v2	q7	unknown	3831	0.00	0.00	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector	titan-embed-text-v2	q8	status	7263	1.00	0.37	0.00
vector	titan-embed-text-v2	q8	unknown	3034	0.00	0.00	0.00
vector	titan-embed-text-v2	q9	status	8417	1.00	0.37	0.00
vector	titan-embed-text-v2	q9	unknown	5328	0.00	0.00	0.00
vector_reranker	embed-english-v3	q1	factual	8733	1.00	0.36	0.00
vector_reranker	embed-english-v3	q10	factual	8629	1.00	0.38	0.00
vector_reranker	embed-english-v3	q11	detail	5028	1.00	0.35	0.00
vector_reranker	embed-english-v3	q12	detail	8258	1.00	0.38	0.00
vector_reranker	embed-english-v3	q13	synthesis	9651	1.00	0.35	0.01
vector_reranker	embed-english-v3	q14	cross-reference	8525	1.00	0.21	0.02
vector_reranker	embed-english-v3	q15	cross-reference	22898	1.00	0.37	0.29
vector_reranker	embed-english-v3	q16	status	9329	1.00	0.37	0.47
vector_reranker	embed-english-v3	q17	synthesis	7112	1.00	0.36	0.00
vector_reranker	embed-english-v3	q18	factual	7105	1.00	0.40	0.07
vector_reranker	embed-english-v3	q19	factual	5419	1.00	0.40	0.00
vector_reranker	embed-english-v3	q2	factual	5479	1.00	0.35	0.00
vector_reranker	embed-english-v3	q20	detail	6895	1.00	0.39	0.00
vector_reranker	embed-english-v3	q21	status	7021	1.00	0.39	0.00
vector_reranker	embed-english-v3	q22	detail	9600	1.00	0.34	0.00
vector_reranker	embed-english-v3	q23	synthesis	5437	1.00	0.41	0.00
vector_reranker	embed-english-v3	q24	cross-reference	6701	1.00	0.37	0.00
vector_reranker	embed-english-v3	q25	factual	8469	1.00	0.38	0.00
vector_reranker	embed-english-v3	q26	synthesis	68777	1.00	0.38	0.04
vector_reranker	embed-english-v3	q27	cross-reference	8028	1.00	0.34	0.04
vector_reranker	embed-english-v3	q28	factual	41279	1.00	0.37	0.00
vector_reranker	embed-english-v3	q29	status	6726	1.00	0.39	0.00
vector_reranker	embed-english-v3	q3	detail	10269	1.00	0.37	0.03
vector_reranker	embed-english-v3	q30	synthesis	9166	1.00	0.38	0.02
vector_reranker	embed-english-v3	q31	detail	8831	1.00	0.38	0.00
vector_reranker	embed-english-v3	q32	detail	6998	1.00	0.40	0.05
vector_reranker	embed-english-v3	q33	detail	6452	1.00	0.39	0.00
vector_reranker	embed-english-v3	q34	detail	9081	1.00	0.35	0.00
vector_reranker	embed-english-v3	q35	detail	20895	1.00	0.38	0.00
vector_reranker	embed-english-v3	q36	detail	8665	1.00	0.37	0.00
vector_reranker	embed-english-v3	q37	detail	4947	1.00	0.36	0.00
vector_reranker	embed-english-v3	q38	detail	8144	1.00	0.36	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector_reranker	embed-english-v3	q39	detail	5747	1.00	0.39	0.00
vector_reranker	embed-english-v3	q4	factual	8471	1.00	0.40	0.00
vector_reranker	embed-english-v3	q40	detail	6972	1.00	0.39	0.05
vector_reranker	embed-english-v3	q5	factual	7374	1.00	0.41	0.00
vector_reranker	embed-english-v3	q6	synthesis	12009	1.00	0.35	0.00
vector_reranker	embed-english-v3	q7	synthesis	10246	1.00	0.40	0.00
vector_reranker	embed-english-v3	q8	status	6308	1.00	0.37	0.00
vector_reranker	embed-english-v3	q9	status	6552	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q1	factual	72872	1.00	0.36	0.02
vector_reranker	embed-multilingual-v3	q10	factual	5608	1.00	0.38	0.00
vector_reranker	embed-multilingual-v3	q11	detail	17009	1.00	0.34	0.01
vector_reranker	embed-multilingual-v3	q12	detail	8554	1.00	0.38	0.00
vector_reranker	embed-multilingual-v3	q13	synthesis	11960	1.00	0.30	0.02
vector_reranker	embed-multilingual-v3	q14	cross-reference	12037	1.00	0.36	0.00
vector_reranker	embed-multilingual-v3	q15	cross-reference	7696	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q16	status	8641	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q17	synthesis	7222	1.00	0.36	0.00
vector_reranker	embed-multilingual-v3	q18	factual	10977	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q19	factual	7448	1.00	0.34	0.00
vector_reranker	embed-multilingual-v3	q2	factual	13843	1.00	0.37	0.03
vector_reranker	embed-multilingual-v3	q20	detail	8192	1.00	0.31	0.01
vector_reranker	embed-multilingual-v3	q21	status	8702	1.00	0.38	0.00
vector_reranker	embed-multilingual-v3	q22	detail	14110	1.00	0.32	0.01
vector_reranker	embed-multilingual-v3	q23	synthesis	5542	1.00	0.41	0.00
vector_reranker	embed-multilingual-v3	q24	cross-reference	6284	1.00	0.41	0.00
vector_reranker	embed-multilingual-v3	q25	factual	10077	1.00	0.38	0.01
vector_reranker	embed-multilingual-v3	q26	synthesis	9659	1.00	0.39	0.01
vector_reranker	embed-multilingual-v3	q27	cross-reference	7705	1.00	0.38	0.00
vector_reranker	embed-multilingual-v3	q28	factual	7885	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q29	status	6768	1.00	0.39	0.00
vector_reranker	embed-multilingual-v3	q3	detail	15849	1.00	0.37	0.01
vector_reranker	embed-multilingual-v3	q30	synthesis	10273	1.00	0.38	0.01
vector_reranker	embed-multilingual-v3	q31	detail	12758	0.00	0.39	0.01
vector_reranker	embed-multilingual-v3	q32	detail	9425	1.00	0.40	0.00
vector_reranker	embed-multilingual-v3	q33	detail	6918	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q34	detail	9717	1.00	0.35	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector_reranker	embed-multilingual-v3	q35	detail	5868	1.00	0.38	0.00
vector_reranker	embed-multilingual-v3	q36	detail	12504	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q37	detail	6008	1.00	0.36	0.00
vector_reranker	embed-multilingual-v3	q38	detail	11841	1.00	0.40	0.01
vector_reranker	embed-multilingual-v3	q39	detail	5102	1.00	0.38	0.04
vector_reranker	embed-multilingual-v3	q4	factual	8023	1.00	0.40	0.00
vector_reranker	embed-multilingual-v3	q40	detail	7684	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q5	factual	9816	1.00	0.25	0.01
vector_reranker	embed-multilingual-v3	q6	synthesis	9800	1.00	0.32	0.00
vector_reranker	embed-multilingual-v3	q7	synthesis	16082	1.00	0.40	0.02
vector_reranker	embed-multilingual-v3	q8	status	5592	1.00	0.37	0.00
vector_reranker	embed-multilingual-v3	q9	status	5738	1.00	0.37	0.00
vector_reranker	titan-embed-text-v1	q1	factual	13305	1.00	0.36	0.01
vector_reranker	titan-embed-text-v1	q10	factual	9902	1.00	0.38	0.00
vector_reranker	titan-embed-text-v1	q11	detail	12084	1.00	0.34	0.01
vector_reranker	titan-embed-text-v1	q12	detail	14141	1.00	0.38	0.00
vector_reranker	titan-embed-text-v1	q13	synthesis	13168	1.00	0.38	0.03
vector_reranker	titan-embed-text-v1	q14	cross-reference	21642	1.00	0.36	0.00
vector_reranker	titan-embed-text-v1	q15	cross-reference	11911	1.00	0.31	0.01
vector_reranker	titan-embed-text-v1	q16	status	10521	1.00	0.25	0.00
vector_reranker	titan-embed-text-v1	q17	synthesis	15739	1.00	0.36	0.04
vector_reranker	titan-embed-text-v1	q18	factual	11312	1.00	0.37	0.00
vector_reranker	titan-embed-text-v1	q19	factual	8396	1.00	0.34	0.01
vector_reranker	titan-embed-text-v1	q2	factual	26001	1.00	0.37	0.01
vector_reranker	titan-embed-text-v1	q20	detail	17673	1.00	0.35	0.03
vector_reranker	titan-embed-text-v1	q21	status	9387	1.00	0.38	0.00
vector_reranker	titan-embed-text-v1	q22	detail	10481	1.00	0.34	0.00
vector_reranker	titan-embed-text-v1	q23	synthesis	5775	1.00	0.41	0.00
vector_reranker	titan-embed-text-v1	q24	cross-reference	11104	1.00	0.37	0.00
vector_reranker	titan-embed-text-v1	q25	factual	9079	1.00	0.38	0.00
vector_reranker	titan-embed-text-v1	q26	synthesis	10383	1.00	0.38	0.00
vector_reranker	titan-embed-text-v1	q27	cross-reference	9365	1.00	0.34	0.00
vector_reranker	titan-embed-text-v1	q28	factual	6555	1.00	0.37	0.00
vector_reranker	titan-embed-text-v1	q29	status	6877	1.00	0.22	0.00
vector_reranker	titan-embed-text-v1	q3	detail	13457	1.00	0.37	0.00
vector_reranker	titan-embed-text-v1	q30	synthesis	12863	1.00	0.40	0.01

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector_reranker	titan-embed-text-v1	q31	detail	8575	1.00	0.39	0.01
vector_reranker	titan-embed-text-v1	q32	detail	7630	1.00	0.40	0.00
vector_reranker	titan-embed-text-v1	q33	detail	7644	1.00	0.35	0.03
vector_reranker	titan-embed-text-v1	q34	detail	9028	1.00	0.31	0.00
vector_reranker	titan-embed-text-v1	q35	detail	8078	1.00	0.38	0.00
vector_reranker	titan-embed-text-v1	q36	detail	12227	1.00	0.37	0.00
vector_reranker	titan-embed-text-v1	q37	detail	8381	1.00	0.36	0.00
vector_reranker	titan-embed-text-v1	q38	detail	9083	1.00	0.40	0.00
vector_reranker	titan-embed-text-v1	q39	detail	12331	1.00	0.39	0.00
vector_reranker	titan-embed-text-v1	q4	factual	10419	1.00	0.40	0.00
vector_reranker	titan-embed-text-v1	q40	detail	10865	1.00	0.26	0.02
vector_reranker	titan-embed-text-v1	q5	factual	10704	1.00	0.25	0.00
vector_reranker	titan-embed-text-v1	q6	synthesis	12474	1.00	0.35	0.00
vector_reranker	titan-embed-text-v1	q7	synthesis	13654	1.00	0.40	0.00
vector_reranker	titan-embed-text-v1	q8	status	6316	1.00	0.37	0.01
vector_reranker	titan-embed-text-v1	q9	status	5022	1.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q1	factual	12087	1.00	0.36	0.00
vector_reranker	titan-embed-text-v2	q10	factual	9508	1.00	0.38	0.02
vector_reranker	titan-embed-text-v2	q11	detail	17570	1.00	0.34	0.02
vector_reranker	titan-embed-text-v2	q12	detail	13473	0.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q13	synthesis	11534	1.00	0.37	0.01
vector_reranker	titan-embed-text-v2	q14	cross-reference	10880	1.00	0.36	0.00
vector_reranker	titan-embed-text-v2	q15	cross-reference	12665	1.00	0.29	0.01
vector_reranker	titan-embed-text-v2	q16	status	9306	1.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q17	synthesis	12065	1.00	0.31	0.01
vector_reranker	titan-embed-text-v2	q18	factual	10908	1.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q19	factual	10095	1.00	0.34	0.01
vector_reranker	titan-embed-text-v2	q2	factual	18153	1.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q20	detail	12983	1.00	0.38	0.02
vector_reranker	titan-embed-text-v2	q21	status	9423	1.00	0.38	0.00
vector_reranker	titan-embed-text-v2	q22	detail	12225	1.00	0.36	0.00
vector_reranker	titan-embed-text-v2	q23	synthesis	12691	1.00	0.39	0.00
vector_reranker	titan-embed-text-v2	q24	cross-reference	12668	1.00	0.41	0.00
vector_reranker	titan-embed-text-v2	q25	factual	12181	1.00	0.37	0.03
vector_reranker	titan-embed-text-v2	q26	synthesis	13357	1.00	0.38	0.00
vector_reranker	titan-embed-text-v2	q27	cross-reference	10201	1.00	0.34	0.00

TABLE D.1. Detailed per-query results for CXNPL corpus (continued).

Arch	Model	QID	Cat	Lat	Faith	AnsRel	CtxRel
vector_reranker	titan-embed-text-v2	q28	factual	9366	1.00	0.36	0.00
vector_reranker	titan-embed-text-v2	q29	status	8117	1.00	0.40	0.00
vector_reranker	titan-embed-text-v2	q3	detail	16894	1.00	0.37	0.01
vector_reranker	titan-embed-text-v2	q30	synthesis	13773	1.00	0.38	0.01
vector_reranker	titan-embed-text-v2	q31	detail	9508	1.00	0.39	0.04
vector_reranker	titan-embed-text-v2	q32	detail	9730	1.00	0.40	0.00
vector_reranker	titan-embed-text-v2	q33	detail	9696	1.00	0.39	0.00
vector_reranker	titan-embed-text-v2	q34	detail	18194	1.00	0.38	0.00
vector_reranker	titan-embed-text-v2	q35	detail	7522	1.00	0.38	0.00
vector_reranker	titan-embed-text-v2	q36	detail	10703	1.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q37	detail	5496	0.00	0.31	0.00
vector_reranker	titan-embed-text-v2	q38	detail	9721	1.00	0.40	0.00
vector_reranker	titan-embed-text-v2	q39	detail	11287	1.00	0.39	0.00
vector_reranker	titan-embed-text-v2	q4	factual	17056	1.00	0.27	0.03
vector_reranker	titan-embed-text-v2	q40	detail	8797	1.00	0.22	0.00
vector_reranker	titan-embed-text-v2	q5	factual	11371	1.00	0.25	0.00
vector_reranker	titan-embed-text-v2	q6	synthesis	10705	1.00	0.30	0.00
vector_reranker	titan-embed-text-v2	q7	synthesis	12282	1.00	0.40	0.00
vector_reranker	titan-embed-text-v2	q8	status	9029	1.00	0.37	0.00
vector_reranker	titan-embed-text-v2	q9	status	8881	1.00	0.37	0.00

D2 Haystack Corpus Results

This section contains the detailed per-query results for the NIAH external validation.

TABLE D.2. Detailed per-query results for the Haystack benchmark.

Model	Question	Expected	Correct
amazon_titan-embed-text-v1	What is the secret animal #1 in the document?	giraffe	Yes
amazon_titan-embed-text-v1	What is the secret animal #2 in the document?	penguin	Yes
amazon_titan-embed-text-v1	What is the secret animal #3 in the document?	spider	Yes
amazon_titan-embed-text-v1	What is the secret animal #4 in the document?	cow	No
amazon_titan-embed-text-v1	What is the secret animal #5 in the document?	squirrel	Yes
amazon_titan-embed-text-v1	What is the secret clothing in the document?	sock	Yes
amazon_titan-embed-text-v1	What is the secret currency in the document?	ruble	Yes
amazon_titan-embed-text-v1	What is the secret drink in the document?	smoothie	Yes
amazon_titan-embed-text-v1	What is the secret flower in the document?	daisy	Yes
amazon_titan-embed-text-v1	What is the secret food in the document?	chocolate	Yes
amazon_titan-embed-text-v1	What is the secret fruit in the document?	lemon	Yes
amazon_titan-embed-text-v1	What is the secret instrument in the document?	violin	Yes
amazon_titan-embed-text-v1	What is the secret kitchen appliance in the document?	microwave	Yes
amazon_titan-embed-text-v1	What is the secret landmark in the document?	Big Ben	No
amazon_titan-embed-text-v1	What is the secret object #1 in the document?	clock	Yes
amazon_titan-embed-text-v1	What is the secret object #2 in the document?	bottle	Yes
amazon_titan-embed-text-v1	What is the secret object #3 in the document?	bowl	No
amazon_titan-embed-text-v1	What is the secret object #4 in the document?	pillow	Yes
amazon_titan-embed-text-v1	What is the secret object #5 in the document?	vase	Yes
amazon_titan-embed-text-v1	What is the secret office supply in the document?	calculator	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
amazon_titan-embed-text-v1	What is the secret shape in the document?	heart	Yes
amazon_titan-embed-text-v1	What is the secret sport in the document?	surfing	Yes
amazon_titan-embed-text-v1	What is the secret tool in the document?	ruler	Yes
amazon_titan-embed-text-v1	What is the secret transportation in the document?	bike	Yes
amazon_titan-embed-text-v1	What is the secret vegetable in the document?	cauliflower	Yes
amazon_titan-embed-text-v2:0	What is the secret animal #1 in the document?	giraffe	Yes
amazon_titan-embed-text-v2:0	What is the secret animal #2 in the document?	penguin	Yes
amazon_titan-embed-text-v2:0	What is the secret animal #3 in the document?	spider	Yes
amazon_titan-embed-text-v2:0	What is the secret animal #4 in the document?	cow	No
amazon_titan-embed-text-v2:0	What is the secret animal #5 in the document?	squirrel	Yes
amazon_titan-embed-text-v2:0	What is the secret clothing in the document?	sock	Yes
amazon_titan-embed-text-v2:0	What is the secret currency in the document?	ruble	Yes
amazon_titan-embed-text-v2:0	What is the secret drink in the document?	smoothie	Yes
amazon_titan-embed-text-v2:0	What is the secret flower in the document?	daisy	Yes
amazon_titan-embed-text-v2:0	What is the secret food in the document?	chocolate	Yes
amazon_titan-embed-text-v2:0	What is the secret fruit in the document?	lemon	Yes
amazon_titan-embed-text-v2:0	What is the secret instrument in the document?	violin	Yes
amazon_titan-embed-text-v2:0	What is the secret kitchen appliance in the document?	microwave	Yes
amazon_titan-embed-text-v2:0	What is the secret landmark in the document?	Big Ben	Yes
amazon_titan-embed-text-v2:0	What is the secret object #1 in the document?	clock	No
amazon_titan-embed-text-v2:0	What is the secret object #2 in the document?	bottle	Yes
amazon_titan-embed-text-v2:0	What is the secret object #3 in the document?	bowl	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
amazon_titan-embed-text-v2:0	What is the secret object #4 in the document?	pillow	Yes
amazon_titan-embed-text-v2:0	What is the secret object #5 in the document?	vase	Yes
amazon_titan-embed-text-v2:0	What is the secret office supply in the document?	calculator	Yes
amazon_titan-embed-text-v2:0	What is the secret shape in the document?	heart	Yes
amazon_titan-embed-text-v2:0	What is the secret sport in the document?	surfing	Yes
amazon_titan-embed-text-v2:0	What is the secret tool in the document?	ruler	Yes
amazon_titan-embed-text-v2:0	What is the secret transportation in the document?	bike	Yes
amazon_titan-embed-text-v2:0	What is the secret vegetable in the document?	cauliflower	Yes
cohere_embed-english-v3	What is the secret animal #1 in the document?	giraffe	Yes
cohere_embed-english-v3	What is the secret animal #2 in the document?	penguin	Yes
cohere_embed-english-v3	What is the secret animal #3 in the document?	spider	Yes
cohere_embed-english-v3	What is the secret animal #4 in the document?	cow	Yes
cohere_embed-english-v3	What is the secret animal #5 in the document?	squirrel	Yes
cohere_embed-english-v3	What is the secret clothing in the document?	sock	Yes
cohere_embed-english-v3	What is the secret currency in the document?	ruble	Yes
cohere_embed-english-v3	What is the secret drink in the document?	smoothie	Yes
cohere_embed-english-v3	What is the secret flower in the document?	daisy	Yes
cohere_embed-english-v3	What is the secret food in the document?	chocolate	Yes
cohere_embed-english-v3	What is the secret fruit in the document?	lemon	Yes
cohere_embed-english-v3	What is the secret instrument in the document?	violin	Yes
cohere_embed-english-v3	What is the secret kitchen appliance in the document?	microwave	Yes
cohere_embed-english-v3	What is the secret landmark in the document?	Big Ben	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
cohere_embed-english-v3	What is the secret object #1 in the document?	clock	Yes
cohere_embed-english-v3	What is the secret object #2 in the document?	bottle	Yes
cohere_embed-english-v3	What is the secret object #3 in the document?	bowl	Yes
cohere_embed-english-v3	What is the secret object #4 in the document?	pillow	Yes
cohere_embed-english-v3	What is the secret object #5 in the document?	vase	Yes
cohere_embed-english-v3	What is the secret office supply in the document?	calculator	Yes
cohere_embed-english-v3	What is the secret shape in the document?	heart	Yes
cohere_embed-english-v3	What is the secret sport in the document?	surfing	Yes
cohere_embed-english-v3	What is the secret tool in the document?	ruler	Yes
cohere_embed-english-v3	What is the secret transportation in the document?	bike	Yes
cohere_embed-english-v3	What is the secret vegetable in the document?	cauliflower	Yes
cohere_embed-multilingual-v3	What is the secret animal #1 in the document?	giraffe	Yes
cohere_embed-multilingual-v3	What is the secret animal #2 in the document?	penguin	Yes
cohere_embed-multilingual-v3	What is the secret animal #3 in the document?	spider	Yes
cohere_embed-multilingual-v3	What is the secret animal #4 in the document?	cow	Yes
cohere_embed-multilingual-v3	What is the secret animal #5 in the document?	squirrel	Yes
cohere_embed-multilingual-v3	What is the secret clothing in the document?	sock	Yes
cohere_embed-multilingual-v3	What is the secret currency in the document?	ruble	Yes
cohere_embed-multilingual-v3	What is the secret drink in the document?	smoothie	Yes
cohere_embed-multilingual-v3	What is the secret flower in the document?	daisy	Yes
cohere_embed-multilingual-v3	What is the secret food in the document?	chocolate	Yes
cohere_embed-multilingual-v3	What is the secret fruit in the document?	lemon	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
cohere_embed-multilingual-v3	What is the secret instrument in the document?	violin	Yes
cohere_embed-multilingual-v3	What is the secret kitchen appliance in the document?	microwave	Yes
cohere_embed-multilingual-v3	What is the secret landmark in the document?	Big Ben	Yes
cohere_embed-multilingual-v3	What is the secret object #1 in the document?	clock	Yes
cohere_embed-multilingual-v3	What is the secret object #2 in the document?	bottle	Yes
cohere_embed-multilingual-v3	What is the secret object #3 in the document?	bowl	Yes
cohere_embed-multilingual-v3	What is the secret object #4 in the document?	pillow	Yes
cohere_embed-multilingual-v3	What is the secret object #5 in the document?	vase	Yes
cohere_embed-multilingual-v3	What is the secret office supply in the document?	calculator	Yes
cohere_embed-multilingual-v3	What is the secret shape in the document?	heart	Yes
cohere_embed-multilingual-v3	What is the secret sport in the document?	surfing	Yes
cohere_embed-multilingual-v3	What is the secret tool in the document?	ruler	Yes
cohere_embed-multilingual-v3	What is the secret transportation in the document?	bike	Yes
cohere_embed-multilingual-v3	What is the secret vegetable in the document?	cauliflower	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret animal #1 in the document?	giraffe	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret animal #2 in the document?	penguin	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret animal #3 in the document?	spider	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret animal #4 in the document?	cow	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret animal #5 in the document?	squirrel	No
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret clothing in the document?	sock	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret currency in the document?	ruble	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret drink in the document?	smoothie	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret flower in the document?	daisy	No
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret food in the document?	chocolate	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret fruit in the document?	lemon	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret instrument in the document?	violin	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret kitchen appliance in the document?	microwave	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret landmark in the document?	Big Ben	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret object #1 in the document?	clock	No
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret object #2 in the document?	bottle	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret object #3 in the document?	bowl	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret object #4 in the document?	pillow	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret object #5 in the document?	vase	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret office supply in the document?	calculator	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret shape in the document?	heart	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret sport in the document?	surfing	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret tool in the document?	ruler	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret transportation in the document?	bike	Yes
graphrag_us_anthropic_claude-3-5-sonnet-20241022-v20	What is the secret vegetable in the document?	cauliflower	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret fruit in the document?	lemon	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret food in the document?	chocolate	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret shape in the document?	heart	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret instrument in the document?	violin	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret tool in the document?	ruler	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret animal #1 in the document?	giraffe	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret clothing in the document?	sock	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret object #2 in the document?	bottle	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret currency in the document?	ruble	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret object #1 in the document?	clock	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret kitchen appliance in the document?	microwave	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret object #3 in the document?	bowl	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret landmark in the document?	Big Ben	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret animal #4 in the document?	cow	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret vegetable in the document?	cauliflower	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret flower in the document?	daisy	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret animal #5 in the document?	squirrel	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret object #4 in the document?	pillow	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret office supply in the document?	calculator	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret animal #2 in the document?	penguin	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret object #5 in the document?	vase	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret drink in the document?	smoothie	Yes

TABLE D.2. Detailed per-query results for Haystack benchmark (continued).

Model	Question	Expected	Correct
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret sport in the document?	surfing	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret transportation in the document?	bike	Yes
graphrag_us_anthropic_claude-sonnet-4-5-20250929-v10	What is the secret animal #3 in the document?	spider	Yes

APPENDIX E

GraphRAG Prompts

The following prompts were used for the GraphRAG implementation, adapted from the Microsoft GraphRAG methodology.

E1 Entity Extraction

```
-Goal-
Given a text document that is potentially relevant to this activity and a list of entity
    types, identify all entities of those types from the text and all relationships among
    the identified entities.

-Steps-
1. Identify all entities. For each identified entity, extract the following information:
   - entity_name: Name of the entity, capitalised
   - entity_type: One of the following types: [{entity_types}]
   - entity_description: Comprehensive description of the entity's attributes and activities
       . INCLUDE any "secret" or "hidden" attributes explicitly mentioned.
Format each entity as ("entity"{tuple_delimiter}<entity_name>{tuple_delimiter}<entity_type>{
    tuple_delimiter}<entity_description>)

2. From the entities identified in step 1, identify all pairs of (source_entity,
    target_entity) that are *clearly related* to each other.
For each pair of related entities, extract the following information:
   - source_entity: name of the source entity, as identified in step 1
   - target_entity: name of the target entity, as identified in step 1
   - relationship_description: explanation as to why you think the source entity and the
       target entity are related to each other
   - relationship_strength: a numeric score indicating strength of the relationship between
       the source entity and target entity
Format each relationship as ("relationship"{tuple_delimiter}<source_entity>{tuple_delimiter
    }<target_entity>{tuple_delimiter}<relationship_description>{tuple_delimiter}<
    relationship_strength>)

3. Return output in English as a single list of all the entities and relationships
    identified in steps 1 and 2. Use {record_delimiter} as the list delimiter.

4. When finished, output {completion_delimiter}
```

E2 Entity Summarisation

```
You are a helpful assistant responsible for generating a comprehensive summary of the data
    provided below.
Given one or two entities, and a list of descriptions, all related to the same entity or
    group of entities.
Please concatenate all of these into a single, comprehensive description. Make sure to
    include information collected from all the descriptions.
If the provided descriptions are contradictory, please resolve the contradictions and
    provide a single, coherent summary.
Make sure it is written in third person, and include the entity names so we have the full
    context.

#######
-Data-
Entities: {entity_name}
Description List: {description_list}
#######
Output:
```

E3 Relationship Summarisation

The relationship summarisation prompt is identical to the entity summarisation prompt, as both perform the same consolidation function for multiple descriptions of the same element.

```
You are a helpful assistant responsible for generating a comprehensive summary of the data
    provided below.
Given one or two entities, and a list of descriptions, all related to the same entity or
    group of entities.
Please concatenate all of these into a single, comprehensive description. Make sure to
    include information collected from all the descriptions.
If the provided descriptions are contradictory, please resolve the contradictions and
    provide a single, coherent summary.
Make sure it is written in third person, and include the entity names so we have the full
    context.

#######
-Data-
Entities: {entity_name}
Description List: {description_list}
#######
Output:
```

E4 Community Summarisation

```
You are a helpful assistant responsible for generating a comprehensive summary of a
    community in a knowledge graph.
The community is defined by a group of entities and their relationships. Your task is to
    create a summary that captures the key themes, patterns, and insights about this
    community.

Given the entities and their relationships within this community, create a summary that:
1. Identifies the main topics and themes
2. Describes the key entities and their roles
3. Explains the relationships and interactions between entities
4. Highlights any important patterns or insights

Make sure the summary is comprehensive yet concise, written in third person.

#######
-Data-
Community ID: {community_id}

Entities in this community:
{entities_list}

Relationships in this community:
{relationships_list}
#######
Output:
```

E5 Query Keyword Extraction

This prompt is used during query processing to extract salient keywords for graph traversal.

```
Extract the most important keywords, entity names, or specific attributes from this query.
Return ONLY a comma-separated list of keywords. Do not include the query itself.

Query: {query}
Keywords:
```

E6 Answer Generation

This prompt is used for the final answer generation step across all RAG architectures.

```
System: You are a helpful assistant. Answer the question based ONLY on the provided context.
    If the answer is not in the context, say 'I don't know'.

User: Context:
{combined_context}

Question: {query}

Answer:
```