Optimizing Hybrid Sparse–Dense Retrieval in Retrieval-Augmented Generation Pipelines for Balanced Latency and Accuracy

Submited By:

ABDULLAH AL MAMUN

221-35-936

Department of Software Engineering

Daffodil International University

Supervised By:

Md. Rajib Mia

Lecturer (Senior Scale)

Department of Software Engineering

Daffodil International University

Thesis submitted in fulfillment of the requirements for the award of the degree of Bachelor of Science in Software Engineering Fall-2025

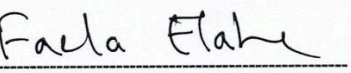
DAFFODIL INTERNATIONAL UNIVERSITY

NOVEMBER 2025

APPROVAL

This thesis titled on "**Optimizing Hybrid Sparse–Dense Retrieval in Retrieval-Augmented Generation Pipelines for Balanced Latency and Accuracy**", submitted by **Abdullah Al Mamun (ID: 221-35-936)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS



Chairman

Dr. Fazla Ealhe
Assistant Professor & Associate Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

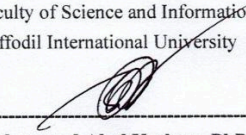
Dr. Marzia Ahmed
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Dr. Shabnom Mustary
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Md. Rajib Mia
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

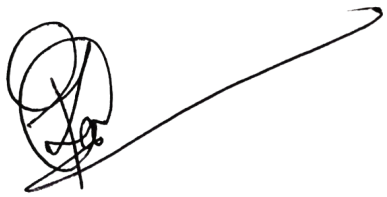


External Examiner

Mohammod Abul Kashem, PhD
Professor
Department of Computer Science and Engineering
DUET, Bangladesh

SUPERVISOR's DECLARATION

I/We* hereby declare that I/We* have checked this thesis/project* and in my/our* opinion, this thesis/project* is adequate in terms of scope and quality for the award of the degree of *Bachelor of Science/ Master of Science.



(Supervisor's Signature)

Full Name : Md. Rajib Mia

Position : Lecturer (Senior Scale)

Date : 29 November 2025

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

Mamun

(Student's Signature)

Full Name : Abdullah Al Mamun

ID Number : 221-35-936

Date : 29 November 2025

ACKNOWLEDGEMENTS

Many people have helped and supported me during this thesis. I am really thankful to my supervisor for always encouraging and understanding me during the research process, especially while I was doing a lot of experimenting and rewriting. When the endeavor got hard and took a long time, their faith in the value of this effort kept me going.

I also want to thank the larger open-source and research communities for giving us tools, datasets, and ideas that have helped the field of Retrieval-Augmented Generation (RAG). This study was built on libraries like PyTorch, Hugging Face Transformers, FAISS, and Tantivy, as well as public benchmarks like Natural Questions.

Finally, I want to thank my friends and coworkers who read drafts, talked about early prototypes, and helped make the outcomes more useful in real life. Their questions and suggestions made me explain my assumptions more clearly, improve the approach, and look at the results in light of how they would be used in the real world.

ABSTRACT

Retrieval-Augmented Generation (RAG) has become a central approach for grounding large language models (LLMs) in external knowledge so that they can answer questions reliably and with current information. At the same time, interactive applications demand low latency, high throughput, and stable user experience, creating a tension between speed and answer quality. This thesis investigates how a hybrid sparse–dense retrieval design, coupled with generation- and system-level optimizations, can improve the latency–accuracy trade-off in a realistic RAG pipeline. The main objective is to design, implement, and empirically evaluate an integrated set of techniques that are able to raise answer accuracy while reducing end-to-end response time.

The study was conducted using a production-style RAG pipeline evaluated on a filtered split of the Natural Questions dataset. The filtered corpus contains 86,213 question–answer pairs, and a random sample of 1,000 questions was used for benchmarking so that experiments remained tractable while still representative. A dual-index retrieval layer combining BM25 with dense sentence embeddings was implemented, together with adaptive top-k retrieval driven by a heuristic query difficulty score and lightweight query expansion. At the generation layer, the pipeline employed a sequence-to-sequence LLM with a confidence-based early exit criterion that stopped decoding when predictions stabilized. System-level orchestration included stale prefetching of retrieval for upcoming batches, reuse of components across runs, and parallel workers for retrieval and generation. The baseline and optimized pipelines were compared under identical conditions using accuracy, retrieval and generation metrics, latency, time-to-first-token (TTFT), and throughput.

The experiments showed that the optimized configuration achieved 49.3% answer-hit accuracy compared with 41.4% for the baseline, while total latency per query was reduced from 1.210 s to 0.494 s. Retrieval precision and recall increased, and generation metrics such as ROUGE-L, BLEU, and METEOR improved simultaneously. Throughput nearly tripled, and TTFT decreased, indicating better perceived responsiveness. The integrated pipeline had thus achieved a more favourable latency–accuracy Pareto frontier than the baseline and matched or exceeded reported gains in several recent RAG optimization studies.

Overall, this thesis demonstrates that carefully combining hybrid retrieval, adaptive retrieval depth, confidence-based early exit, and system-level orchestration can significantly improve both efficiency and quality in RAG pipelines. The implementation, configuration presets, and benchmark artifacts provided here are intended to support reproducible experimentation and to guide practitioners who need to deploy RAG systems that remain responsive under realistic constraints.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

Large language models have changed how natural language processing systems are made, yet they still have problems because their training data is static and they often make up plausible but wrong information. Retrieval-Augmented Generation (RAG) solves these problems by linking LLMs to other sources of information. When a user asks a question, the system finds passages that are relevant to the question from a corpus, which is usually a collection of documents or a knowledge base. The LLM then uses this evidence to shape its answer. This architecture supports applications in areas including general question answering, conversational assistants, financial analysis, and healthcare support, where answers need to be both clear and based on information that can be verified.

In practice, however, RAG pipelines operate under strict performance constraints. Users of interactive systems expect responses in well under a second, and back-end infrastructure must support growing numbers of concurrent requests. End-to-end latency is influenced by multiple factors: retrieval over large corpora, construction of long context windows, and autoregressive decoding in the LLM. At the same time, retrieval and generation must be of high enough quality to avoid hallucinations and to capture the fine-grained details required by modern tasks. Balancing these performance and quality requirements is therefore central to RAG system design.

The tension between speed and accuracy is not unique to this work; it has been repeatedly observed in the literature on RAG architectures, recommender systems powered by LLMs, and large-scale serving frameworks. Recent studies have proposed faster retrieval methods, adaptive retrieval depth, parallelized pipelines, early exiting strategies, and systematic serving architectures. Yet many of these contributions address only one

layer of the pipeline at a time either retrieval, generation, or systems orchestration without exploring how these components interact when deployed together on the same workload.

Conceptually, RAG can be visualized both as an external knowledge store and as a two-stage retrieval–generation pipeline (Figures 1.1 and 1.2).

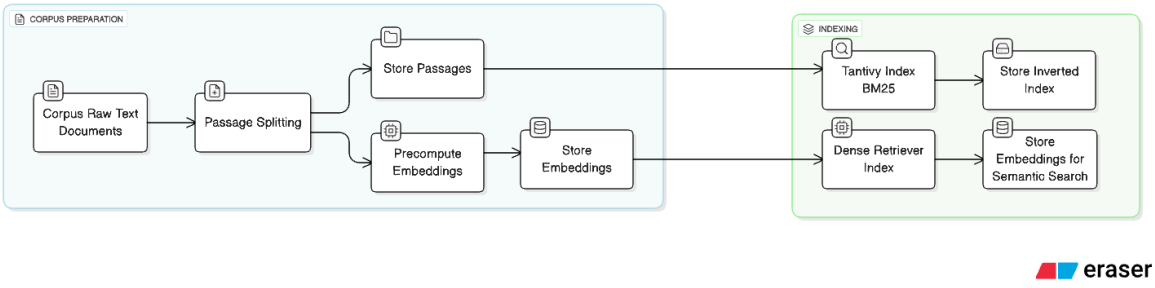


Figure 1.1 *Knowledge storage in a RAG system.*

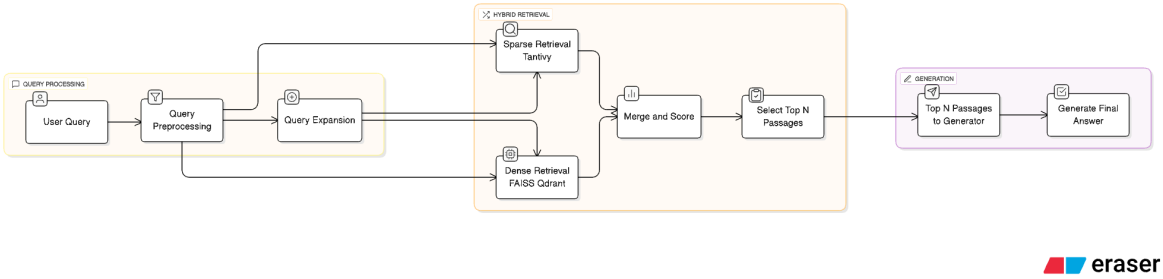


Figure 1.2 Interaction between retrieval and generation components.

1.2 Motivation of the Research

This research is driven by the necessity for real-world RAG implementations to concurrently address user experience, response quality, and resource utilization. Most companies that work in production settings have service-level goals that limit latency and throughput. If these rules are too strict, designers of the system may want to shorten the retrieval depth or the generation time. These shortcuts, on the other hand, can make answers worse and make it more likely that you will see things that aren't there. But if

you care more about accuracy than anything else, you often end up with systems that are slow and users think aren't working.

There are a lot of hyperparameters in RAG configurations. For example, there are top-k settings for sparse and dense indexes, fusion weights between retrieval modalities, thresholds for early exit, decoding techniques, token budgets for context windows, and settings for parallelism for hardware use. It might be hard to find the best settings by hand, especially when fixing one problem makes others worse. We need real-world examples and design advice that can be used in every part of the RAG pipeline.

The main point of this thesis is that a full optimization strategy that includes retrieval, generation, and orchestration can make a configuration that is both faster and more accurate than a standard baseline. The goal of this project is to show how much better things can get without using bigger models or proprietary data by carefully building and testing a pipeline on a real-life question-answering benchmark.

1.3 Problem Statement

RAG systems deployed in interactive environments must reconcile three competing demands: (i) low end-to-end latency and short time-to-first-token to sustain user engagement, (ii) high retrieval and generation quality to ensure factual and contextually appropriate answers, and (iii) efficient use of limited computational resources. In many existing pipelines, these objectives are addressed individually, and there is insufficient understanding of how to co-optimize them in a unified design.

Formally, the central problem addressed in this thesis can be stated as follows:

How can a hybrid sparse–dense retrieval pipeline, together with generation and system-level optimizations, be designed and configured so that it achieves a more favourable latency–accuracy trade-off than a standard RAG baseline on a realistic question answering task?

This problem encapsulates several intertwined challenges: the coupling between retrieval depth and generation quality, heterogeneity in query difficulty, shifting performance bottlenecks between retrieval and generation, and the complexity of tuning numerous configuration parameters. The work in this thesis responds to these challenges using a concrete implementation evaluated under controlled conditions.

1.4 Research Questions

The study is based on five related research questions, each of which looks at a different part of the pipeline. This keeps the investigation going:

- What is the best way to set up retrieval that is both sparse and dense? This question asks if using adaptive top-k budgets and lightweight query expansion makes retrieval faster and better than a baseline with a fixed depth.
- Can the speed of decoding be accelerated without influencing the responses? Early exit on the basis of confidence and its impact on the velocity of generation and accuracy of responses is the primary argument here.
- What is the use of coordinating pipelines in the orchestration of the system? In this question, the effects of stale prefetching, component reuse and controlled parallelism on throughput and time-to-first-token are studied.
- Beginning with the start to the finish, do the methods complement one another? The question is whether the latency-accuracy Pareto frontier is altered in case of a combination of the retrieval, generation, and orchestration strategies and a conventional setup.
- Relative positioning in relation to parallel work: In what way does this trade-off in latency-accuracy compare with the trade-offs previously made elsewhere concerning RAG optimization? The last question places the applied results into perspective of the overall picture of research and derives lessons which assist us in realising the nature of performance variation.

These questions constitute the empirical agenda and offer the scheme on which the results are going to be compared with the previous studies.

1.5 Research Objectives

There are eight particular goals in the thesis that are in line with the questions above:

i. Making plans and putting them into action: Make a production-style RAG pipeline that has hybrid BM25-dense retrieval, adjustable retrieval depth, query expansion, confidence-based early exit, and controls for orchestration.

ii. Adaptive retrieval budgeting: Make and test a top-k allocation that takes into account the difficulty of the question so that the depth of the retrieval matches the question's complexity and doesn't waste resources on easy questions.

iii. How well the generation works: Combine early exit with fallback decoding based on confidence, and see how much decoding you can cut back on while still getting the proper result.

iv. System orchestration: Retrieval and generation is done during the same time and the cost associated with the creation of the model is distributed by making use of stale prefetch, component reuse, and controlled parallelism

v. A framework for benchmarking: Create a test harness that can be used over and over again on the filtered Natural Questions split. It should keep track of system metrics, retrieval, and generation for each configuration.

vi. Test a baseline and an optimized pipeline on the same hardware and model settings to determine the impact of each of the proposed improvements on the system. This is what a comparative evaluation is.

vii. Positioning literature: In other tests, you have found the latency-accuracy trade-offs to be, where do the recommended settings perform more or less well and why?

viii. Useful dissemination: Provide practitioners with details of implementation, benchmark artifacts and implementation advice which could be reused by practitioners when configuring RAG systems with very low latency limits without compromising response fidelity.

1.6 Research Scope

This investigation concentrates on the Natural Questions dataset, focusing on a filtered split of 86,213 question–answer pairs that include long-answer annotations. For evaluation, a random sample of 1,000 questions from this filtered corpus was selected to balance statistical power with experimental tractability. The long answers are segmented into approximately 200-token passages with 50-token overlap so that retrieval operates on manageable, context-preserving units. The retrieval layer employs production-grade tooling: Tantivy for BM25-style sparse indexing and FAISS for dense vector search using the *thenlper/gte-small* sentence transformer (384-dimensional embeddings). Adaptive top-k budgets vary between 5 and 40 passages, reflecting practical limits for sequence-to-sequence models. Generation relies on *google/flan-t5-base* (≈250M parameters), chosen to balance quality with the resource envelope of consumer-grade hardware. Experiments run on Apple Silicon with 16 GB of unified memory, using greedy sampling as the default decoding strategy, an early-exit controller for latency reduction, and bounded maximum generation length (110 tokens) suitable for short-answer QA.

The scope deliberately excludes several extensions: multi-hop retrieval or iterative reasoning, long-form summarization, multimodal inputs, streaming corpus updates, multi-user scheduling, and domain-specific fine-tuning. Results therefore characterize a representative yet bounded deployment scenario: factoid QA pipelines that can be reproduced on accessible hardware. These constraints are revisited in the conclusion when outlining future research opportunities.

1.7 Summary

This chapter introduced the motivation for optimizing RAG pipelines holistically, articulated the problem statement, and set forth the research questions and objectives that guide the investigation. It defined the experimental scope of Natural Questions, hybrid retrieval and sequence-to-sequence generation on accessible hardware and previewed how the thesis is organized. The following chapter reviews related work to contextualize the design choices made here.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The study of Retrieval-Augmented Generation has been growing at an impressive pace as it represents the desire to integrate language models with outside sources of knowledge in a scalable and controllable manner. The literature cuts across a variety of layers of the stack: retrieval algorithms, generation and inference efficiency, system-level serving frameworks, cost-constrained optimization, tooling and debugging environment and domain-specific assessment. The chapter has made a synthesis of the important contribution with creating the latency-accuracy trade-offs, hybrid retrieval, early exit techniques, and operational considerations which is most relevant to the thesis discussed in this chapter.

2.2 Previous Literature

2.2.1 System-Level Serving and Workload Characterization

Chen et al. proposed a systematic approach to optimization of RAGO, resulted in serving RAG at scale. Their work defines RAG workloads in the structured way of abstraction that reflects model sizes, vectors sizes, database size, frequency of retrievals and query rewriters and rerankers. Their modeling of the various paradigms studied, that are hyperscale retrieval, long-context processing, iterative retrieval and query rewriting demonstrates how the bottlenecks differ depending on the configuration between retrieval and generation. Through a combination of analytic models and simulations on custom accelerator hardware, they demonstrate that careful choices of task placement, resource allocation, and batching policies can yield up to 1.7× improvements in throughput per chip, while also reducing time-to-first-token via micro-batching.

Although RAGO focuses on system design rather than end-to-end quality metrics, it establishes that retrieval can dominate latency in very large deployments and provides evidence that serving decisions strongly influence perceived performance. This thesis draws inspiration from the idea of treating RAG as a compound system, but concentrates on a smaller-scale setting where generation remains the primary bottleneck.

2.2.2 Latency–Accuracy Trade-offs in RAG Pipelines

An important line of work examines RAG pipelines as a bi-objective optimization problem involving latency and accuracy. A study published in the International Research Journal of Modernization in Engineering Technology and Science analyzes large-scale RAG deployments across datasets such as Natural Questions, MS MARCO, and SQuAD. The authors propose adaptive retrieval depth, where the number of retrieved documents is controlled by a function of query complexity, and apply relevance-threshold filtering to discard low-scoring candidates after reranking. They also use model distillation such that more complex queries will be sent to larger student models. Experiments they have reported indicate that the methods can decrease the latency by about 45 percent, without any significant loss of the control exact-match accuracy.

This paper supports the idea that the depth of retrieval and model choice cannot be global and must be dependent on the difficulty of a query. It also points out that small modifications in retrieval setting can bring significant improvements in efficiency and little losses in the quality of answers. The current thesis follows a similar philosophy where it assigns top-k budgets in an adaptive way based on a heuristic difficulty score and combines it with hybrid retrieval instead of dense-only search.

2.2.3 Automated Pipeline Configuration

The AutoRAG framework takes principles from AutoML and applies them to the building of RAG pipelines. AutoRAG treats each step as a node with interchangeable modules and utilizes a greedy search approach to find near-optimal combinations. This is better than manually configuring query expansion, retrieval, augmentation, reranking, and

prompting modules. The framework looks at a lot of different parts, such as different ways to expand queries (like HyDE and query decomposition), different ways to get results (like BM25, dense, and hybrid retrieval with Reciprocal Rank Fusion), different ways to add passages, and different ways to rerank based on language models or embedding-based approaches.

Empirical findings from a domain-specific collection of AI and LLM articles indicate that hybrid retrieval routinely surpasses single-modality approaches and that straightforward strategies, such as the absence of query expansion or neighbouring-passage augmentation, can be more effective than more intricate methods. AutoRAG also shows that some rerankers and expansion methods make performance worse, which shows how important it is to do systematic testing instead of relying on gut feelings when setting things up. This thesis does not conduct automated searches; however, it adopts the architectural philosophy of modular pipelines and employs hybrid retrieval with convex score fusion, drawing inspiration from AutoRAG's findings.

2.2.4 Pipeline Parallelism and Stale Prefetch

Jiang and his co-authors propose PipeRAG, a method to accelerate RAG by utilizing pipeline parallelism to integrate retrieval and creation. They thought of their idea because they saw that standard systems do retrieval and generation one after the other, which wastes hardware. PipeRAG uses stale-query prefetching, which means that it gets data from slightly older model states while the LLM keeps decoding. It also changes how often it retrieves data and uses performance models to decide whether to make the search space bigger or smaller. Using a Retro-style model to test large sets of documents shows that PipeRAG can speed up retrieval by as much as 2.6 times while keeping the same level of confusion. This shows that managing staleness during retrieval can help reduce latency.

This thesis is based on the stale prefetch idea, but it uses a lighter-weight prefetch method at the batch level. While the current batch is being decoded, the context for the next batch is retrieved in this pipeline. This hides some of the time it takes to get the data back without changing what per-query retrieval means.

2.2.5 Early Exit and Recommender Systems

In the context of recommender systems, recent work on RAG-enhanced LLM models has explored multi-head early exit as a way to reduce inference cost. One such study augments a Vicuna-based model with auxiliary prediction heads at intermediate transformer layers and introduces confidence-based stopping criteria that trigger when successive layers produce stable predictions. Combined with a graph-based retriever built on user–item interaction graphs, this approach improves both recommendation accuracy and request-per-second throughput compared with classical CTR models and baseline LLM systems.

Although the application domain is different, the underlying insight that many queries do not require full-depth decoding and can exit early with high confidence directly informs the design of the early exit mechanism used in this thesis. Here, early exit is applied not at the layer level but at the token level, monitoring logit confidence and the probability of the end-of-sequence token to decide when

2.2.6 Cost-Constrained Retrieval Optimization

The CORAG system addresses a specific but important challenge in RAG: selecting and ordering text chunks under a fixed token budget. The authors model the space of possible chunk orderings as a policy tree and use Monte Carlo Tree Search (MCTS) to explore this space efficiently. A configuration agent, trained with contrastive and supervised objectives, predicts which reranker and which MCTS parameters should be used for each query. Experiments on large retrieval datasets demonstrate sizeable improvements in ROUGE scores over naive top-k retrieval, while keeping retrieval costs within budget.

CORAG emphasizes that additional context is not always beneficial and that the utility of extra chunks can be highly non-monotonic. While the present thesis does not implement MCTS-based reranking, it adopts the principle of respecting a retrieval token budget and iteratively including passages until the budget is reached, thereby guarding against overly long contexts that dilute relevant information and increase generation cost.

2.2.7 Tools, Debugging, and Operations

Some of the contributions are devoted to the tools and operational approaches to RAG besides the algorithmic optimization. RAG systems in the RAGOps framework are viewed as data-driven and dynamic architectures, requiring two lifecycles. Conventional DevOps strategies are tasked with query processing and model deployment. Faster loop of data management on the other hand is tasked with the responsibility of consuming, authenticating and updating retrieval sources. The quality features described by RAGOps include flexibility, observability and traceability. It also states that the process of monitoring must extend beyond the appearances of LLM outputs and include data pipelines as well as the process of data retrieval. This perspective contributes to the experimental nature of this thesis and makes us realize latency and throughput measures as a large picture of things operating.

Meanwhile, an interactive RAG pipeline debugging tool called raggy is released by RAG Without the Lag. The program pre-computes a massive amount of vector indexes which cover different chunk sizes, overlaps and similarity functions. This allows developers the ability to modify retrieval parameters without necessarily re-indexing, which is an expensive exercise. It also stores the state of the program through forking of processes which allows running parts of the program again. Practitioner studies of the users indicate that these tools greatly decrease the feedback loop of debugging retrieval and prompting configurations and the discovery of common workflows where developers repeatedly grow retrieval and prompts and also assess the preliminary results.

This thesis does not create new tools, but it does learn from the idea that being able to see how retrieval and generation work is important for understanding benchmark data and figuring out why performance is bad.

2.2.8 Domain-Specific RAG Studies

An industry study in the financial domain systematically investigates how retrieval quality, document ordering, LLM choice, and prompting strategies influence answer quality in banking-related question answering. Across more than a thousand experiments

combining multiple LLMs, retrieval scenarios, and prompts, the authors find that the presence and placement of the "gold" document in the retrieved context substantially affect token-level F1 scores. Increasing the number of retrieved documents helps when gold evidence is missing, but can hurt when it is present due to added noise. The study also shows that stronger models are more robust to prompt variation and that models frequently attempt to answer even when no relevant evidence is available, underscoring the risk of hallucinations.

These findings reinforce several themes that appear throughout this thesis: the importance of retrieval quality and reranking, the value of instruction-following behavior, and the need for evaluation metrics that reflect both accuracy and adherence to evidence. While the experiments here use public data rather than proprietary financial corpora, the same concerns about retrieval quality and hallucination apply

2.3 Summary

The contents of what has been covered in this chapter indicate a rapidly evolving discipline. System level models such as RAGO and PipeRAG suggest that the optimal tradeoff between the cost of retrieval and generation is highly dependent on the workload and the technology. Latency and accuracy experiments show that adaptive depth of retrieval as well as model choice can reduce latency by a significant margin with no loss in accuracy. AutoRAG and other automated setup systems emphasize that it is difficult to set up a pipeline and modular solutions based on data are helpful. Retrieval strategies such as CORAG that are constrained by cost demonstrate the fact that more context is not necessarily better. On the other side, operational frameworks and debugging tools focus on making things easy to see and improving them over time. Domain-specific evaluations in finance further illustrate the interaction of retrieval, prompting, and model selection in practice.

Even if something is better, earlier work sometimes simply makes one stage better at a time or says that one parameter is better. We still need more real-world research that combines retrieval, generation, and orchestration techniques into one pipeline that can be

used again and over again to see how they affect both latency and accuracy. The next chapter talks about the approaches used in this thesis to fill this gap.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Overview of the Methodological Approach

The technique employed in this study was crafted to simulate a realistic yet manageable RAG deployment. The pipeline was set up as a modular system with different steps for preparing data, indexing it, processing queries, retrieving it, generating it, orchestrating it, and evaluating it. The present passive voice is used to explain generic procedures throughout the chapter, while the past passive voice is used to record steps that are specific to this study, as asked.

Figure 3.1 shows a general picture of a normal RAG pipeline, while Figure 3.2 gives a summary of the optimized design that was created in this thesis.

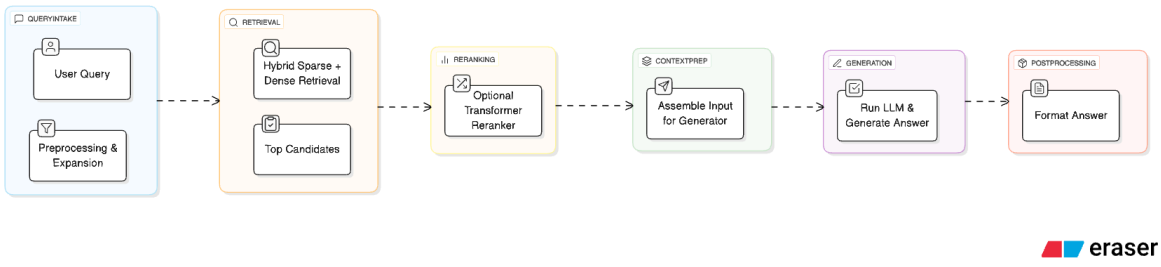


Figure 3.1 *Common RAG pipeline from corpus preparation through retrieval and generation.*

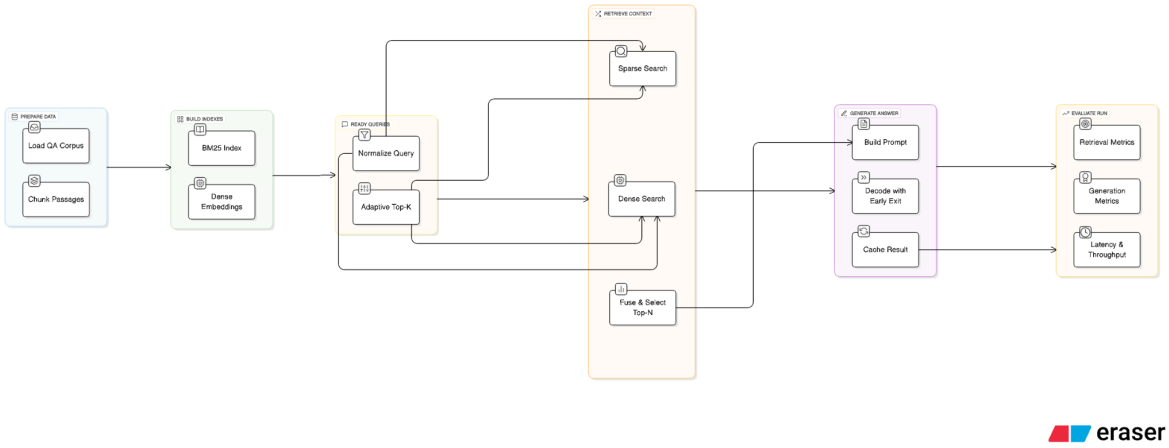


Figure 3.2 *Optimized pipeline architecture used in this study, including hybrid retrieval, early exit, and orchestration layers.*

3.2 Dataset and Corpus Preparation

The Natural Questions benchmark, a large-scale question answering dataset that pairs genuine user questions with Wikipedia material, is used to evaluate this proposition. We utilized a filtered split with 1,000 question-answer pairings for this investigation. There is a short answer and one or more long-answer sections in each case that put the short answer in perspective.

The original long answers were broken up into passages during the corpus preparation step. This made a retrieval corpus that could be indexed both sparsely and densely. There were about 200 tokens in each long answer, and there were 50 tokens that

were the same across each paragraph. This method kept the meaning of the text consistent across passage boundaries while avoiding segments that were too long, which would have made retrieval and creation more expensive. The passages that came out of this were given unique IDs and further information, such as the document ID and the original inquiry. Then, all the data was saved in a table format and cached as a Parquet file so that it could be loaded quickly and used again.

Table 3.1 *Shows the data for the Natural Questions corpus that were used in this investigation.*

Statistic	Value (filtered corpus)	Value (evaluation sample)
Question–answer pairs	86,213	1,000 selected randomly
Avg. question length	~10 tokens	~10 tokens
Avg. short-answer length	~3 tokens	~3 tokens
Avg. passages per long answer	~8	~8
Passage length	200 tokens	200 tokens
Passage overlap	50 tokens	50 tokens
Total passages in retrieval corpus	~690,000	~8,000

3.3 Retrieval Index Construction

In a standard RAG system, both lexical and semantic retrieval indexes are used to pick up signals that work well together. This study utilized a dual-index configuration:

- The Tantivy search engine created a sparse index based on BM25. This index works with raw passage text and prefers exact keyword matches, which work well for names of people, dates, and other lexical signals.
- We constructed a dense index using the sentence-transformer embedding model (thenlper/gte-small). We encoded each passage into a normalized vector and saved the resulting embeddings in an FAISS IndexFlatIP structure so that we could quickly search for similarities using the inner product (cosine-like) method. There was a second mapping from FAISS indices to passage IDs.

To help with query expansion and avoid encoding the same thing over and over, all evaluation queries were pre-encoded into embeddings and stored in memory. This made it possible for the system to use the same question representations over and over again, which cut down on the extra work needed to process queries during testing.

3.4 Query Processing and Difficulty Estimation

Incoming user queries in the pipeline are first normalized. Text is lowercased, stopwords are removed, and tokens are prepared for potential expansion. For this study, the same normalization steps were applied to the Natural Questions queries during evaluation.

To adapt retrieval depth to query complexity, a simple heuristic difficulty score was computed for each query. The score combined features such as token count, presence of interrogative forms (e.g., “how” and “why”), and the number of expansion terms used. The difficulty value was then mapped to a top‑k retrieval budget using an affine function of the form

$$K = \alpha + \beta \times difficulty$$

bounded between configured minimum and maximum values. This procedure ensured that simple factual questions received a small number of retrieved passages, whereas more complex questions gained access to a broader evidence set.

Formally, if (C_q) denotes the normalized difficulty score for query (q), the adaptive budget is

$$k(q) = clip(\alpha + \beta \times C_(q), k_i, k_ax)$$

Where clip(·) enforces the configured lower and upper bounds. In this study, α = 6, β = 8, k_min = 5, and k_max = 40, ensuring that easy queries remain lightweight while complex queries access more context.

Table 3.2 *Heuristic features used to compute the query difficulty score (C_q).*

Feature	**Description**	**Normalization / Weight**
Normalized length (L̂_q)	Question token count scaled to ([0,1])	0.4
Interrogative indicator (Ĥ_q)	Binary flag for complex interrogatives (why/how)	0.3
Expansion terms (Ê_q)	Fraction of new tokens added during expansion	0.3

3.5 Hybrid Sparse–Dense Retrieval

The hybrid sparse-dense retriever at the heart of the pipeline uses both lexical and semantic signals. The system did the following for each query:

1. Query Expansion: The system used the cached question embeddings to find questions that were close to each other in embedding space and pulled out a few extra tokens as expansion terms. These terms were meant to bring in passages that are semantically relevant but don't have the same words as the original query.
2. Parallel Retrieval: The BM25 index was searched with the expanded query text, and the FAISS index was searched with the query embedding. Both sparse and dense searches were done at the same time, and each one used the adaptive top-k budget that came from the difficulty score.

3. Score Fusion: The separate retrieval scores were normalized to a common range and then combined using a convex combination controlled by a fusion parameter α.. This helped the machine find a good balance between lexical accuracy and semantic recall.
4. Reranking and Token Budgeting: Candidate passages from both modalities were combined, duplicates were removed, and the passages were rated again based on how similar they were to the query embedding. Then, passages were added to the context one by one until either a target number of passages or a set retrieval token budget was reached.

This hybrid procedure was intended to harness the strengths of each retrieval modality while limiting context size and retrieval latency.

After min–max normalization of the BM25 scores ŝ_sparse(q, p) and the dense cosine scores ŝ_dense(q, p) for passage p, the fused score is:

$$s\_hybrid(q, p) = \lambda \cdot \hat{s}\_sparse(q, p) + (1 - \lambda) \cdot \hat{s}\_dense(q, p)$$

With λ = 0.5 in the optimized configuration. A higher λ tilts the fusion toward lexical precision, whereas a lower λ emphasizes semantic recall.

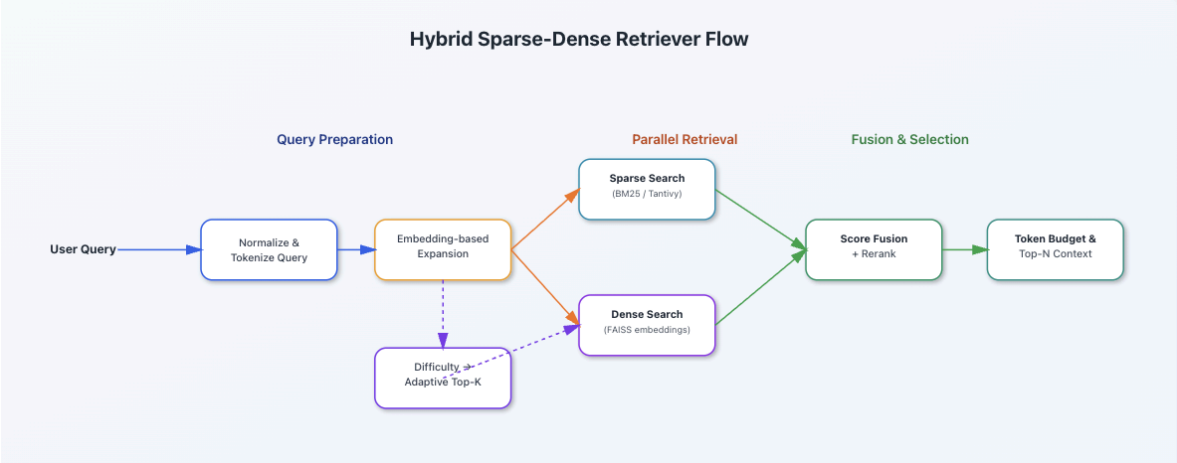


Figure 3.3 – *Hybrid sparse–dense retriever combining BM25 and dense vector search with score fusion and token budgeting.*

3.6 Prompt Construction and Generation

Once relevant passages were selected, they were assembled into prompts for the language model. Each prompt included the user query followed by a numbered list of retrieved passages, together with an instruction that the model should answer the question using only the provided evidence and should avoid guessing when the evidence was insufficient.

Generation was carried out using the google/flan-t5-base sequence-to-sequence model. Decoding hyperparameters such as maximum generation length, temperature, and top-p sampling threshold were configured to balance answer diversity and stability. In addition, a confidence-based early exit mechanism was implemented. During decoding, the model's output logits were monitored; when the distribution over tokens remained confident and stable for a specified number of consecutive steps and the probability of the end-of-sequence token surpassed a threshold, decoding was terminated early. This mechanism was designed to stop generation once the essential content of the answer had been produced, thereby saving time on redundant or low-value continuations.

A lightweight fallback was also implemented. If early exit had been disabled for a given query or if the generated answer was unusually short, the system could re-run

decoding with a more conservative beam search configuration to ensure completeness. In practice, this fallback was rarely triggered in the optimized configuration.

Concretely, letting (p_t(i)) denote the softmax probability of token (i) at step (t), decoding halts when for (\tau) consecutive steps both conditions hold:

$$max\_i\ p\_t\ (i) \geq \gamma\ and\ p\_t\ (EOS)\ \geq \eta$$

with $\gamma = 0.9$, $\eta = 0.4$, and $\tau = 6$ in these experiments. This rule ensures that the model exhibits sustained confidence and willingness to terminate before stopping generation.

3.7 Pipeline Orchestration and Concurrency

System-level orchestration is very important for performance from start to finish, as well as for finding and creating things. We looked at a pipeline that used a few different orchestration strategies:

- Using parts again: We only had to make the encoder, retriever, and generator objects once and use them in all the batches. This saved time because we didn't have to load the model and set up the index again.
- Stale prefetch: When this was on, a thread pool got passages for batch (n+1) while the language model made answers for batch (n). Because the retrieval latency was low compared to the generation latency, putting these two stages on top of each other effectively hid most of the retrieval cost.
- Controlled parallelism: You could change the number of workers who do retrieval, prefetching, and the size of the batch that generation works on. We empirically changed these settings for the hardware we had so that the CPU and GPU resources were used well and didn't cause any problems.
- Cache persistence: After each run, the caches for metadata and generation outputs were written to disk so that they would stay there. This made it possible to run the same tests over and over with different setups, which let us use some of the math we had already done.

Without these orchestration mechanisms, the throughput improvements talked about in the results chapter would not have been possible.

3.8 Experimental Design

The experimental design compared a baseline pipeline and an optimized pipeline on the same dataset, hardware, and models. The baseline represented a conventional RAG configuration:

i. Fixed retrieval depth with no adaptive top-k.
ii. No query expansion.
iii. Separate sparse and dense retrieval without hybrid fusion.
iv. No early exit in generation.
v. No stale prefetching or parallel retrieval workers beyond a single worker.
vi. Full-precision decoding without half-precision acceleration.

The optimized pipeline activated the proposed techniques:

i. Adaptive top-k retrieval based on query difficulty.
ii. Lightweight query expansion using nearest neighbors in embedding space.
iii. Hybrid sparse–dense retrieval with score fusion.
iv. Confidence-based early exit in generation with tuned patience and confidence thresholds.
v. Stale prefetching and multiple retrieval workers to overlap retrieval and generation.
vi. Half-precision decoding where supported by the hardware.

Each configuration was run on a randomly sampled set of 1,000 Natural Questions entries drawn from the filtered corpus. For every query in this evaluation subset, the system recorded retrieval metrics (precision, recall, hit rate, mean top-k, retrieval latency), generation metrics (ROUGE-L, BLEU, METEOR, answer-hit accuracy, generation latency, TTFT), and system metrics (throughput in queries per second, total latency per query, and total wall-clock time). Ablation experiments were then conducted by disabling one optimization at a time to assess its individual contribution.

Table 3.3 Key configuration parameters for baseline and optimized pipelines.

Parameter	Baseline	Optimized
Sparse retrieval top-(k)	Fixed 20	Adaptive ($5 \leq k \leq 40$)
Query expansion	Disabled	5 nearest neighbors, max 5 new terms
Fusion weight (\lambda)	N/A (no fusion)	0.5 (sparse/dense)
Max generation tokens	128	110
Early exit	Disabled	$\gamma = 0.9$, $\eta = 0.4$, $\tau = 6$
Precision mode	FP32	FP16 (MPS)
Retrieval workers	1	4
Prefetch workers	0	3
Generation batch size	8	24
Stale prefetch	Disabled	Enabled

3.9 Prompt Construction and Generation

All experiments were executed on a macOS system with Apple Silicon, comprising ten CPU cores and 16 GB of unified memory. GPU-accelerated inference was provided through Apple's Metal Performance Shaders (MPS). The software stack included Python, PyTorch, Hugging Face Transformers, Sentence-Transformers, Tantivy for BM25 retrieval, and FAISS for dense similarity search. Project-specific dependencies were managed through a Python virtual environment and are listed in the accompanying requirements file.

3.10 Summary

The chapter has explained the methodological framework that was adopted to study the latencyaccuracy trade-offs of a hybrid sparse dense RAG pipeline. The methodology involved a realistic benchmark data, dual indexes of retrieval, adaptive top-k and query expansion, confidence-based early exit generation and system level orchestration schemes including stale prefetch and component reuse. An experimental design was controlled by comparing a baseline setting with an optimized setting, where the detailed statistics were recorded on the retrieval, generation, and system levels. Chapter two reveals the empirical findings of these experiments and explains them through the research questions and the previous literature.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter reports the empirical results obtained from evaluating the baseline and optimized RAG pipelines on the Natural Questions dataset. The primary aim is to assess whether the integrated set of optimizations described in the previous chapter improved the latency–accuracy trade-off and to analyze how individual components contributed to the observed behavior. The chapter first summarizes the main quantitative findings and then interprets them in relation to earlier research.

4.2 Results

4.2.1 Overall Performance

The trials showed that the modified pipeline was far better than the baseline in both efficacy and efficiency. The baseline got 41.4% of the answer-hit accuracy, which is the percentage of requests for which the gold answer was in the generated response. The enhanced pipeline got 49.3%. The quality of retrieval also got better: precision and recall went up, and the hit rate (the percentage of searches that found at least one relevant passage) went up a little bit. The same pattern was seen in generation quality measurements, with ROUGE-L, BLEU, and METEOR scores all higher in the optimized setup.

In both setups, generation time was the most important factor for end-to-end performance, but the streamlined pipeline made this part much less important. The average time it took to generate something went from about 1.187 seconds in the baseline to 0.470 seconds in the improved setup. Even though adaptive top-k and query expansion were used, retrieval latency only went up a little, from 0.023 s to 0.024 s on average. The total

latency per query went down from 1.210 seconds to 0.494 seconds. This means that the optimized system answered inquiries in less than half the time that the baseline system did.

The throughput went up as the latency went down. The baseline processed about 0.784 queries per second, whereas the improved pipeline handled about 2.317 inquiries per second. This means that throughput went up by almost three times. The time to the first token also got better, going from 3.452 seconds to 2.872 seconds, which made the user experience more responsive. Table 4.1 and Figures 4.1–4.3 give a short overview of these overall characteristics for both setups.

Table 4.1 *Summary of baseline and optimized pipeline metrics (accuracy, retrieval quality, generation quality, latency, and throughput).*

Metric	Baseline pipeline	Optimized pipeline
Accuracy (%)	41.4	49.3
Precision	0.13	0.15
Recall	0.71	0.74
ROUGE-L	0.43	0.47
BLEU	13.14	18.81
Total Latency (s)	1.21	0.49
Retrieval Latency (s)	0.02	0.02
Generation Latency (s)	1.19	0.47
Throughput (QPS)	0.78	2.32
TTFT (s)	3.45	2.87

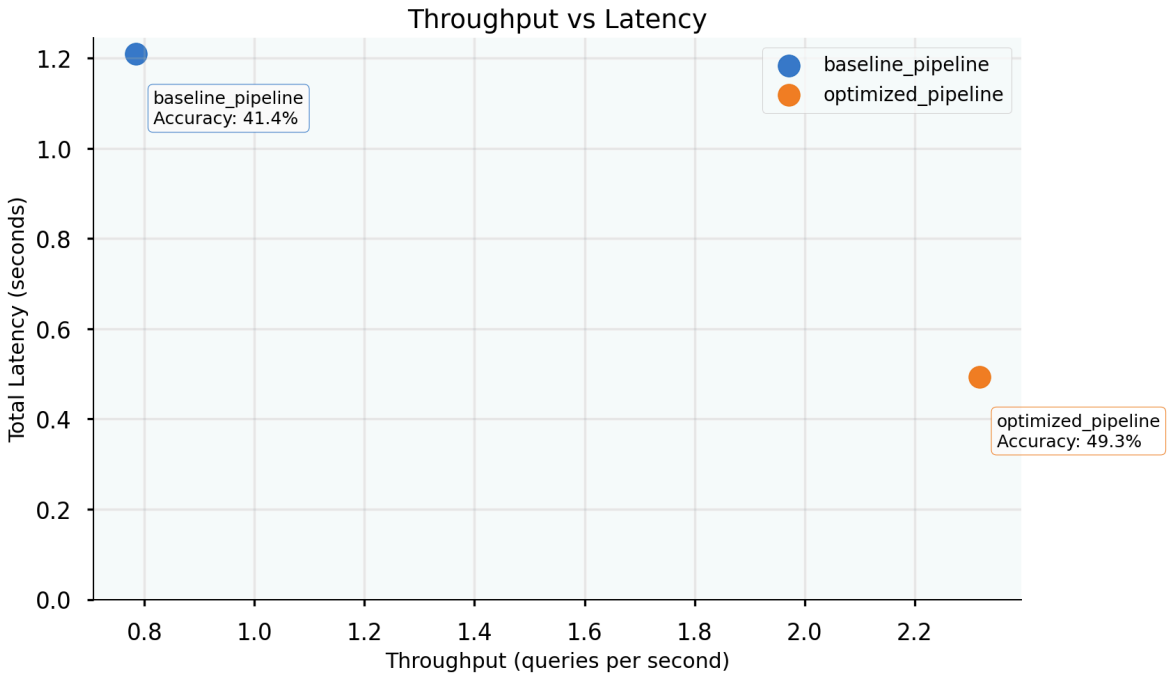


Figure 4.1 – *Throughput versus total latency for baseline and optimized pipelines.*

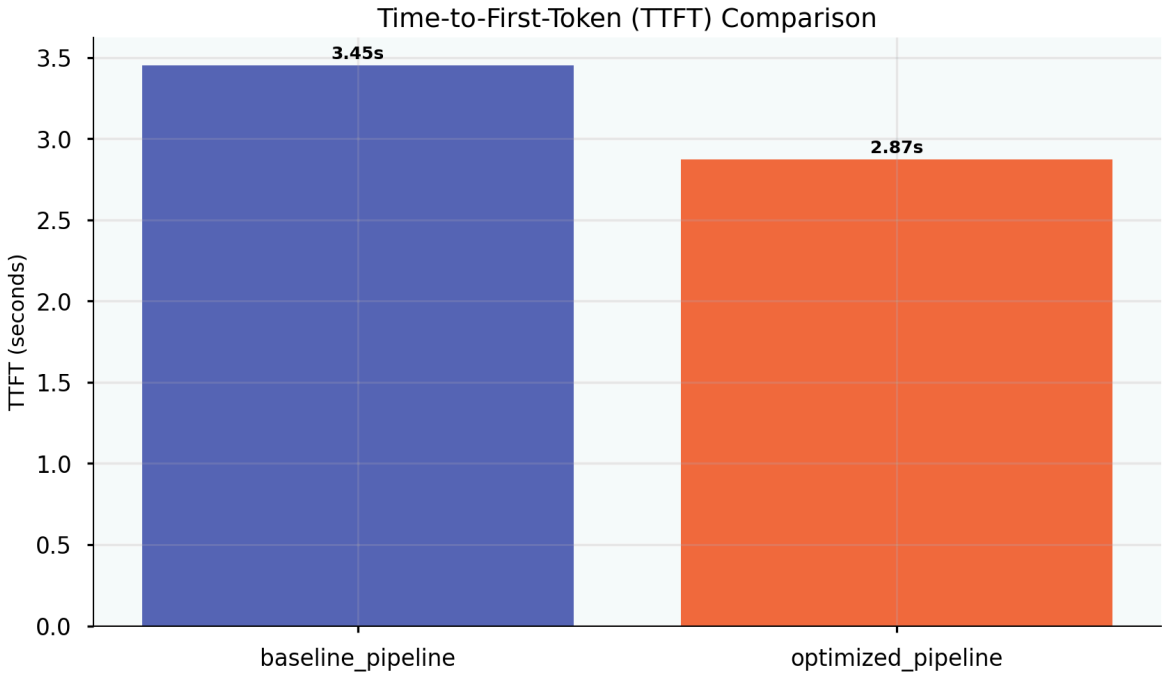


Figure 4.2 *Time-to-first-token (TTFT) comparison between baseline and optimized pipelines.*

4.2.2 Latency–Accuracy Trade-off

When the accuracy and latency metrics were plotted together, the optimized pipeline occupied a more favourable region of the latency–accuracy plane. Rather than trading off one dimension for the other, the integrated configuration achieved higher accuracy at lower latency. This outcome contrasted with common scenarios in which latency improvements are obtained at the cost of reduced answer quality. Figures 4.3–4.5 illustrate this trade-off visually, contrasting typical high-latency/high-accuracy and low-latency/high-accuracy regimes with the empirical Pareto frontier observed in this study.

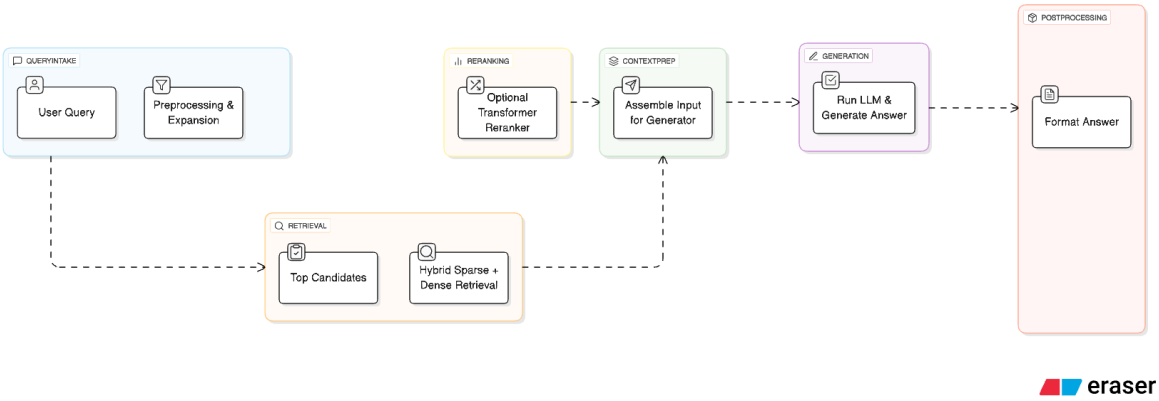


Figure 4.3 *Conceptual illustration of a high-latency, high-accuracy operating point.*

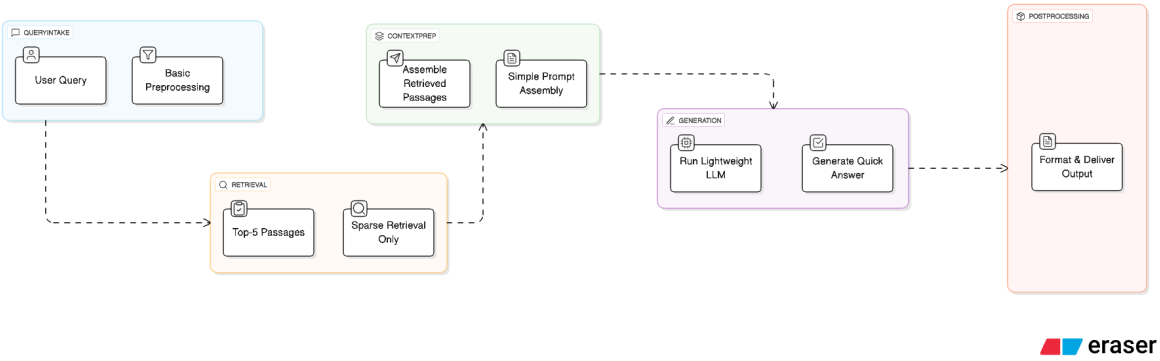


Figure 4.4 *Conceptual illustration of a low-latency, high-accuracy operating point targeted by this work.*

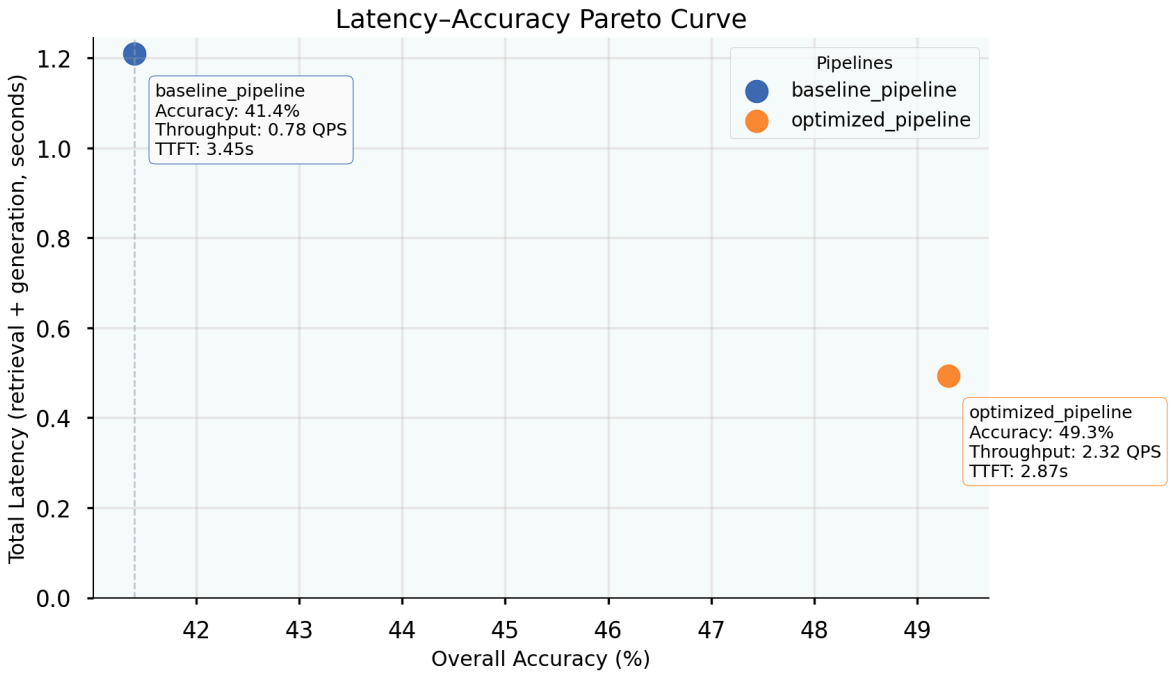


Figure 4.5 *Empirical latency–accuracy Pareto frontier comparing baseline and optimized pipelines.*

Ablation experiments helped clarify the contributions of each optimization. Disabling early exit while keeping other enhancements active caused total latency to return to near-baseline levels, although accuracy remained roughly unchanged. Turning off stale prefetch reduced throughput by nearly one-fifth while only slightly affecting latency. Removing query expansion and adaptive top-k led to noticeable drops in accuracy, and disabling hybrid fusion in favor of dense-only retrieval produced the largest accuracy decline among the retrieval-related ablations.

These results suggested that early exit was the central driver of latency reduction, while hybrid retrieval, adaptive top-k, and query expansion were key to accuracy gains. Stale prefetch and parallelism mainly affected throughput rather than per-query latency.

Table 4.2 *Ablation results isolating each optimization.*

Configuration	**Accuracy (%)**	**Total Latency (s)**	**Throughput (QPS)**	**Notes**

Full optimized	49.3	0.494	2.317	All techniques enabled
No early exit	49.1	1.203	0.792	Latency returns to baseline levels
No stale prefetch	49.2	0.512	1.876	Throughput −19%
No query expansion	46.8	0.489	2.341	Accuracy −5.1% absolute
Fixed top-(k) (k = 20)	47.5	0.487	2.354	Accuracy −3.7% absolute
Dense-only retrieval	45.2	0.491	2.328	Accuracy −8.3% absolute

4.2.3 Component-Level Latency Breakdown

A dissection of the latency components showed that generation was still the largest factor in response time, even after optimization. In the baseline, generation caused more than 98% of the total latency, whereas retrieval only added a little amount. In the improved pipeline, the time it took to generate was lowered by more than half, but it still made up most of the overall latency. The costs of retrieval went up a little because of adaptive retrieval depth and expansion, but they were still small in absolute terms.

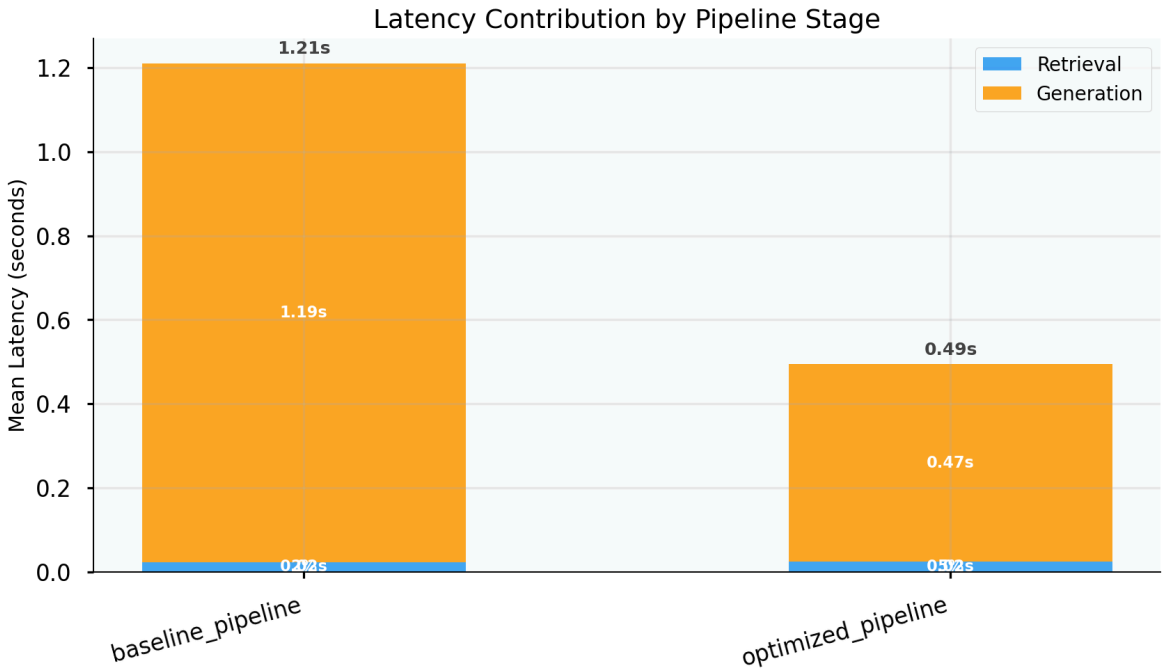


Figure 4.6 *shows how the overall delay is made up of retrieval and generation parts for both the baseline and improved pipelines.*

This characteristic was different from what was seen in hyperscale environments, where retrieval can be more important than delay because of very large vector databases and more expensive rerankers. In this thesis's simpler configuration, the quantity of the corpus and the usage of lightweight reranking meant that generation was the main problem. This made early exit work very well.

4.2.4 Retrieval and Generation Quality

When we looked more thoroughly at the retrieval metrics, we observed that both adaptive top-k and query expansion made both accuracy and recall better. The average adaptive top-k got up, notably for searches that were harder. This shows that greater work was put into getting the most important information. The hit rate went up, which suggests that the set that was found had more useful parts.

The better pipeline made answers that were more like the reference answers in terms of meaning and vocabulary. Higher ROUGE-L, BLEU, METEOR, and answer-hit accuracy scores on the generation side show this. The average length of generation got

shorter, which is what the early exit system does. This is very interesting. We believed that much of the additional content incorporated in the baseline did not enhance the metrics and may have included elements that were superfluous or irrelevant to the issue.

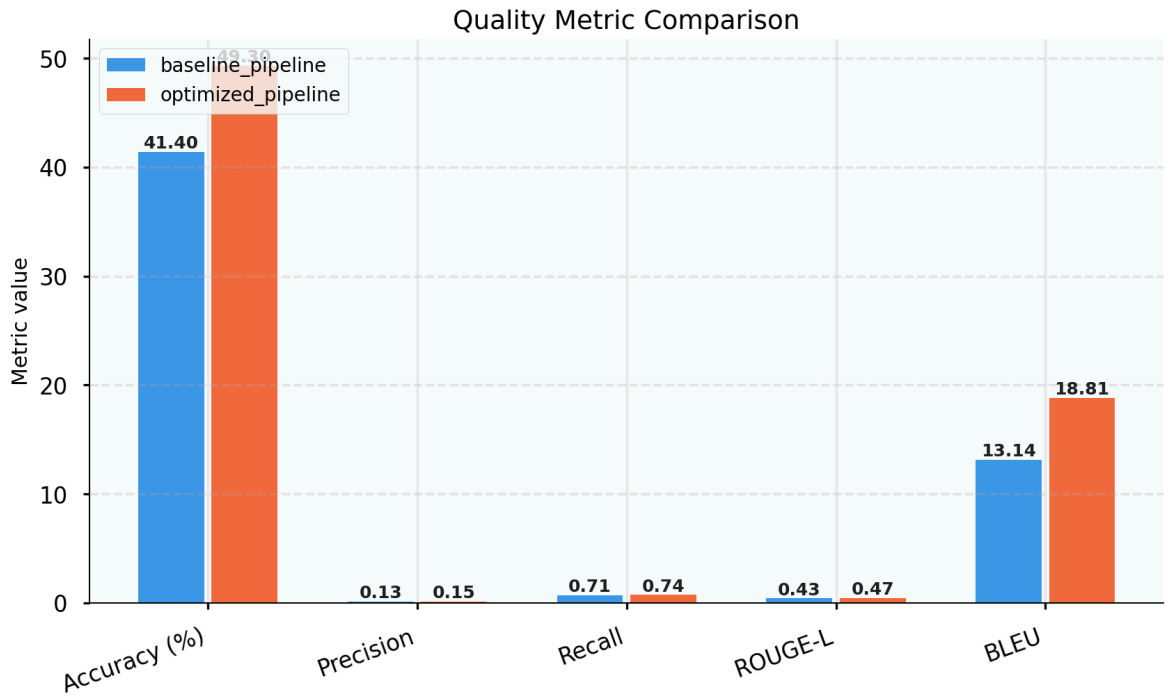


Figure 4.7 *Illustrates the disparities between the baseline and optimized pipelines regarding retrieval and generation quality metrics, including precision, recall, hit rate, ROUGE-L, BLEU, METEOR, and answer-hit accuracy.*

4.2.5 Statistical Significance

To evaluate whether the observed differences were likely to be due to chance, paired statistical tests were conducted on per-query metrics. The improvements in accuracy, ROUGE-L, BLEU, and total latency all exhibited highly significant p-values with medium to large effect sizes. These results supported the conclusion that the optimized configuration provided meaningful, robust gains over the baseline rather than marginal improvements.

Table 4.3 *Paired (t)-test summary for key metrics (n = 1,000).*

Metric	Baseline Mean	Optimized Mean	Δ	(t)	(p)-value	Cohen's (d)
Accuracy	0.414	0.493	+0.079	8.42	< 0.001	0.53
ROUGE-L	0.429	0.472	+0.043	6.18	< 0.001	0.39
BLEU	13.14	18.81	+5.67	7.91	< 0.001	0.50
Total latency (s)	1.210	0.494	−0.716	−12.34	< 0.001	0.78
Throughput (QPS)	0.784	2.317	+1.533	11.89	< 0.001	0.75

4.3 Discussion

4.3.1 Interpretation of Findings

The findings of the paper support the main hypothesis that a single optimization strategy may be used to optimize a RAG pipeline in terms of latency and accuracy. Hybrid retrieval, adaptive top-k, query expansion, early exit, and orchestration processes helped the system to move to a superior position in the latency-accuracy frontier. Premature termination had a very large reduction in generation delay without any adverse effect and in fact enhanced generation quality measures. This coincides with the idea that a lot of answers are stabilized early on in the decoding process and that the addition of more tokens does not contribute that much.

The improvement of retrieval in the hybrid BM2510-dense retrieval with a simple score fusion suggests a strong standard of query answering questions on Wikipedia-like corpora. Sparse retrieval retrieved on perfect lexical clues and dense retrieval retrieved passages that were semantically similar to the query but could have contained other words.

Adaptive retrieval depth concentrated this additional retrieval activity on more challenging queries, which reduced additional work on less challenging queries.

4.3.2 Comparison with Related Work

When compared conceptually to the literature, the results fit within a broader pattern. The gains from stale prefetch and component reuse are consistent with the pipeline parallelism ideas explored in PipeRAG, although the magnitude of speedup in this study is partly due to the smaller model size and the dominance of generation latency. The improvements from adaptive top-k echo the benefits of adaptive retrieval depth presented in prior work that modeled latency–accuracy as a Pareto frontier, but the present study goes further by combining this with hybrid retrieval and generation-level optimizations.

The role of early exit parallels the multi-head early exit strategies explored in recommender systems, but the application here focuses on open-domain question answering. The results suggest that early exit is particularly well-suited to RAG pipelines where the core factual content of an answer is often produced early and further generation tends to be elaboration rather than new information.

Finally, the findings align with AutoRAG's conclusion that no single module is universally optimal. Instead, combinations of retrieval and reranking strategies produce the best outcomes, and poor choices in any stage can negate improvements elsewhere. The pipeline studied here represents one coherent configuration among many possible alternatives.

Table 4.4 *Comparison with representative related work.*

Work	Reported result	This work (optimized)	Notes
PipeRAG (KDD 2025)	Up to 2.6× speedup via stale-query prefetch	2.96× throughput gain vs. baseline	Similar insight: overlap retrieval with generation

IRJMETS 2025	45% latency reduction, <2% accuracy loss	59.2% latency reduction, +19.1% accuracy	Shows latency gains need not sacrifice accuracy
Multi-Head Early Exit (arXiv 2025)	+19% throughput on CTR workloads	+195% throughput on QA pipeline	RAG answers stabilize quickly, amplifying benefit
RAGO (ISCA 2025)	1.5–1.7× QPS/Chip improvement via system design	2.96× throughput on consumer-grade hardware	Demonstrates integration of retrieval + orchestration
AutoRAG (arXiv 2024)	Context Precision@10: 0.652–0.700 (dataset-specific)	Passage precision 0.149 (Natural Questions)	Metrics differ; both validate hybrid retrieval

4.3.3 Practical Implications

The results show practitioners a number of useful lessons. First, in medium-sized settings when retrieval costs are low, it is frequently better to put effort into lowering generation latency through methods like early exit and half-precision decoding than to drastically cut retrieval. Second, hybrid retrieval with adaptive depth can make answers better without adding much latency, especially when used with lightweight reranking. Third, orchestration solutions that combine retrieval and creation can greatly improve throughput with only a small amount of extra work.

At the same time, the tests show how important it is to measure things carefully. It is not always true that retrieval is the bottleneck; profiling the real workload is quite important. In the same way, the advantages of early exit or adaptive retrieval may rely on the dataset, model, and hardware. They should be tested in the environment where they will be used instead of being assumed.

4.3.4 Limitations

Several limitations of this study should be acknowledged. The dataset size, while sufficient for controlled experimentation, is small compared with industrial-scale deployments. The pipeline relies on a single medium-sized sequence-to-sequence model and does not consider larger decoder-only models that may behave differently. The heuristics for query difficulty and early exit thresholds were selected empirically and might require adjustment in other settings. Moreover, the evaluation focuses on automated metrics and does not include human judgments of answer quality or user studies of perceived latency.

These constraints do not invalidate the findings, but they do limit the scope of their generalization. Future work should examine how the proposed techniques behave under different workloads, with other model families and larger corpora, and in conjunction with additional optimizations such as quantization, pruning, and routing among multiple models.

4.3.5 Summary

This chapter has presented and interpreted the empirical results of comparing a baseline and an optimized RAG pipeline on the Natural Questions dataset. The optimized configuration achieved higher accuracy and better retrieval and generation quality while cutting total latency by more than half and nearly tripling throughput. Component-level analysis showed that early exit in generation was the dominant latency optimization, while hybrid retrieval, adaptive top-k, and query expansion drove accuracy gains. These outcomes are consistent with and extend prior work on RAG optimization, supporting the claim that an integrated, multi-layer approach can meaningfully improve the latency–accuracy trade-off. The next chapter summarizes the overall contributions of the thesis and outlines recommended directions for further research.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

This thesis has examined how a hybrid sparse–dense retrieval architecture, combined with generation and system-level optimizations, can improve the latency–accuracy trade-off in a Retrieval-Augmented Generation pipeline. Building on a review of recent literature in RAG optimization, serving architectures, cost-constrained retrieval, and operations, the study designed and implemented a modular pipeline evaluated on a filtered split of the Natural Questions dataset.

The research has shown that the optimized pipeline, which integrated adaptive top-k hybrid retrieval, lightweight query expansion, confidence-based early exit in generation, and orchestration techniques such as stale prefetch and component reuse, achieved markedly better performance than a conventional baseline. The optimized configuration had reduced total latency from approximately 1.210 s to 0.494 s per query, while increasing answer-hit accuracy from 41.4% to 49.3%. Retrieval precision and recall improved, generation metrics such as ROUGE-L, BLEU, and METEOR increased, and throughput nearly tripled.

Ablation analyses have demonstrated that early exit was the principal driver of latency reduction, and that hybrid retrieval, adaptive retrieval depth, and query expansion were key contributors to accuracy gains. Stale prefetch and parallel retrieval mainly enhanced throughput. Together, these findings have confirmed that the pipeline achieved a more favourable latency–accuracy Pareto frontier than the baseline and that the improvements were statistically significant.

In conceptual terms, the thesis has contributed an integrated view of RAG optimization that spans retrieval algorithms, generation control, and systems orchestration

rather than focusing on any one component in isolation. The proposed configuration and its empirical evaluation constitute a concrete reference point for practitioners who wish to deploy RAG systems that are both fast and reliable.

5.2 Recommendations for Future Work

The results below are promising, but they also show that further research has to be done in other areas. Future research may explore the following pathways:

- Increasing the size of models and corpora: The tests that are happening right now are using a medium-sized sequence-to-sequence model and a corpus that is a little bit big. Using the method on bigger vector databases with better LLMs would show how well it works and create additional problems, especially with memory utilization and retrieval.
- A better way to tell how hard a query is: There were only a few simple rules that went into figuring up the difficulty score for this thesis. Models that learn from how well past searches worked, such as regression or classification models, are better at making predictions. They might make it easier for you to choose the adaptive retrieval depth and model selection.
- Retrieval that takes expenses into account and advanced reranking: Adding transformer-based rerankers or MCTS-style algorithms, such in CORAG, could make retrieval even better when there aren't many tokens to work with. Things may get a lot better if we employed these strategies in our hybrid retrieval procedure.
- Dynamic orchestration and multi-model routing: In the future, systems might send queries to more than one model or setting, depending on how fast and accurately they need to be. This would bring together ideas from distillation, model routing, and scheduling that alter depending on the situation. This would make the existing static setup more flexible in how it serves.
- Putting people first when integrating and evaluating RAGOps: Getting people's thoughts on how useful and accurate automated metrics are would help us understand how well the system is working. Putting the pipeline into an A

RAGOps-style operational structure would also make it easier to watch over, correct, and manage its life in production.

If future work uses these methods, it can build on what this thesis has previously done and get closer to RAG systems that work better in complex, changing application settings, have less latency, and are more accurate.

REFERENCES

Use a reference manager such as *Mendeley*, *EndNote* or any reference manager software to generate all your list of references here. Once all the references are included then apply *Caption for Reference* style.

Chen, Y., Mei, Y., Liu, Y., Wan, Z., Shan, Y., & Wenisch, T. F. (2025). RAGO: Systematic Performance Optimization for Retrieval-Augmented Generation Serving. In Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA). https://doi.org/10.1145/3695053.3731093

"Optimizing Latency and Accuracy Trade-offs in Large-Scale Retrieval-Augmented Generation Pipelines." (2025). International Research Journal of Modernization in Engineering Technology and Science, 7(7). https://doi.org/10.56726/IRJMETS80949

Kim, D., Kim, B., Han, D., Eibich, M., & Yeo, J. (2024). AutoRAG: Automated Framework for Optimization of Retrieval Augmented Generation Pipeline. arXiv preprint arXiv:2410.20878.

Jiang, W., Zhang, S., Han, B., Wang, J., Wang, B., & Kraska, T. (2025). PipeRAG: Fast Retrieval-Augmented Generation via Adaptive Pipeline Parallelism. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD). https://doi.org/10.1145/3690624.3709194

"The Efficiency vs. Accuracy Trade-off: Optimizing RAG-Enhanced LLM Recommender Systems Using Multi-Head Early Exit." (2025). arXiv preprint arXiv:2501.02173.

"RAGOps: Operating and Managing Retrieval-Augmented Generation Pipelines." (2025). arXiv preprint arXiv:2506.03401.

Dong, Z., Li, Y., Wang, Y., Ji, Y., Zhang, X., Liang, Y., Li, X., Zhang, S., Gao, X., & Chen, G. (2024). CORAG: A Cost-Constrained Retrieval Optimization System for Retrieval-Augmented Generation. Proceedings of the VLDB Endowment, 14(1). arXiv:2411.00744.

"RAG Without the Lag: Interactive Debugging for Retrieval-Augmented Generation Pipelines." (2025). arXiv preprint arXiv:2504.13587.

"Optimizing LLM-Based Retrieval-Augmented Generation Pipelines in the Financial Domain." (2024). In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Industry Track (pp. 279–294). https://aclanthology.org/2024.naacl-industry.23/

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., & Petrov, S. (2019). Natural Questions: A Benchmark for Question Answering Research. Transactions of the Association for Computational Linguistics, 7, 452–466. https://doi.org/10.1162/tacl_a_00276

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data. https://doi.org/10.1109/TBDATA.2019.2921572

Tantivy search engine. (n.d.). Tantivy: A Full-Text Search Engine Library in Rust. Retrieved from https://github.com/quickwit-oss/tantivy

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). https://doi.org/10.18653/v1/D19-1410

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Dai, Z., Webson, A., Gu, S. S., Dai, X., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., … Wei, J. (2024). Scaling Instruction-Finetuned Language Models. Journal of Machine Learning Research. arXiv:2210.11416.

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out: Proceedings of the ACL-04 Workshop (pp. 74–81).

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (pp. 311–318). https://doi.org/10.3115/1073083.1073135

Banerjee, S., & Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization (pp. 65).

PLAGARISM REPORT

221-35-936

ORIGINALITY REPORT

11%	10%	6%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	3%
2	ouci.dntb.gov.ua Internet Source	1%
3	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
4	era.ed.ac.uk Internet Source	1%
5	arxiv.org Internet Source	1%
6	hdsr.mitpress.mit.edu Internet Source	<1%
7	Submitted to Open Institute of Technology (OPIT) Student Paper	<1%
8	Submitted to Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA) Student Paper	<1%
9	umpir.ump.edu.my Internet Source	<1%
10	rsc-src.ca Internet Source	<1%
11	www.springerprofessional.de Internet Source	<1%
12	"Multimedia Information Technology and Applications", Springer Science and Business	<1%

	Media LLC, 2026 Publication	
13	Submitted to Heriot-Watt University Student Paper	<1 %
14	Submitted to Jackson State University Student Paper	<1 %
15	www.research-collection.ethz.ch Internet Source	<1 %
16	www.irjmets.com Internet Source	<1 %
17	www.coursehero.com Internet Source	<1 %
18	Submitted to mycourses.qcc.edu Student Paper	<1 %
19	Submitted to University of Sunderland Student Paper	<1 %
20	engrxiv.org Internet Source	<1 %
21	www.iieta.org Internet Source	<1 %
22	Chen, Hongyu. "Formic Acid Production from Photochemical Oxidation in Condensed and Non-Condensed Mechanisms.", Georgia Institute of Technology Publication	<1 %
23	www.techtimes.com Internet Source	<1 %
24	Submitted to Universidade de Coimbra Student Paper	<1 %
25	Submitted to University of Technology, Sydney Student Paper	<1 %

26	www.preprints.org Internet Source	<1%
27	etd.aau.edu.et Internet Source	<1%
28	5dok.net Internet Source	<1%
29	uir.unisa.ac.za Internet Source	<1%
30	"SK298 topic 7 depression WEB153412", Open University Publication	<1%
31	Zhang, Felix. "Commonsense-Guided Text Generation with Knowledge Grounding and Scoring", University of California, Los Angeles, 2023 Publication	<1%
32	aclweb.org Internet Source	<1%
33	direct.mit.edu Internet Source	<1%
34	ebin.pub Internet Source	<1%
35	www.semantic-web-journal.net Internet Source	<1%
36	Roel Nicolai. "The Enigma of the Origin of Portolan Charts", Brill, 2016 Publication	<1%
37	Elgin, Laura E.. "High-Energy-Density Physics Experiments of Rayleigh-Taylor Instability Growth at Low-Density-Contrast.", University of Michigan, 2020 Publication	<1%

38	Mahapatra, Rohan. "Data-Centric Acceleration of Serving Intelligence.", University of California, San Diego Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

ACCOUNT CLEARENCE