Lightweight Retrieval-Augmented Generation for Contract Question-Answering on Local Hardware

University of Bern – CAS AML Final Project

Rolf Moser, rolf.moser@students.unibe.ch

9 June 2025

Abstract

This work presents a compact retrieval-augmented generation pipeline that answers questions about a large IT service agreement entirely on CPU-only local hardware. PDF parsing, sentence-level chunking, hybrid sparse-plus-dense retrieval, and answer generation are executed within a single virtual machine, eliminating reliance on external APIs. Four embedding models, three chunking strategies, and several lightweight language models were benchmarked, orchestrating 26 hyper-parameters via Weights & Biases sweeps. Hybrid BM25-plus-vector retrieval combined with 384-token sentence chunks achieved the highest recall, while the Qwen 3-4B generator provided the best balance of speed and graded accuracy. The final configuration reaches a mean reciprocal rank of 0.63 and a graded-accuracy score of 0.98, delivering answers in under one minute. These results demonstrate that high-quality contract question-answering can be realised in confidentiality-sensitive environments that preclude GPUs or cloud services and offer practical guidance for deploying similar lightweight RAG systems in service-management contexts.

1. Introduction

IT service agreements are typically long, complex documents that play a critical role in operational governance and compliance. Despite their importance, accessing specific information within such contracts remains a time-consuming task, especially in service management contexts where timely answers are essential. Traditional document search tools often fall short in handling queries that require semantic understanding or multi-paragraph reasoning.

This report explores the design and evaluation of a Retrieval-Augmented Generation (RAG) [1] pipeline tailored for question answering over a large real-world IT service agreement procured by a health care provider. The pipeline is implemented entirely on a single local virtual machine without GPU and integrates lightweight models and open-source components. It is intended to be reproducible, transparent, and suitable for constrained environments where confidentiality and data residency requirements prohibit cloud-based solutions and proper on-premises hardware is not available.

The goal is to assess how various pipeline choices—such as chunking strategies, embedding models, retrieval methods, chunk augmentation and prompt construction affect overall performance. Evaluation combines quantitative retrieval metrics (e.g., recall, reciprocal rank) with a qualitative assessment based on LLM-as-a-Judge scoring.

The main contribution is a systematic evaluation of trade-offs in contract QA pipelines, along with practical guidance for deploying such systems in environments where full-scale cloud-based solutions may not be viable.

2. Data

The corpus comprises a 3151-page IT-Service Agreement in English, structured into a Master Service Agreement, thematic schedules, annexes, and amendments. The documents are provided as text-based PDFs, so no optical character recognition was required. In total, the agreement consists of 69 documents that contain 8.2 million characters, 1.1 million words, and 4015 images. Images are excluded from the word count and were not processed.

Textual sections do not strictly adhere to a hierarchical numbering scheme but instead adopt a reader-centric layout. Typography and formatting vary considerably across the constituent documents. Some documents are almost entirely tables, others are mostly text with some tables and images. The PDF documents are converted to Markdown files using marker-pdf [2]. Only minimal post-processing was needed to remove some page footers and normalize the chapter headings.

Data is processed on-premises under a non-disclosure agreement. No document content is transmitted to external APIs, and the resulting vector index resides behind the client's firewall to meet confidentiality requirements.

QA for the ITSM domain is often a reasonably simple task, as the questions are usually answerable based on a single section of the agreement. "Is this work effort covered by the base fee?", "Until what time is the support available?" are examples of common questions. But they are hard to answer quickly for human operators if the relevant section is not found because of the large number of documents and sections. Furthermore, questions often arise in meetings with internal stakeholders or with the service provider and must be answered quickly.

2.1 Ground Truth

To support retrieval and generation evaluation, a benchmark set of 20 ground truth questions was compiled from a list of practical contracting queries relevant to IT service management (ITSM) scenarios. Each question is designed to be answerable based on specific, identifiable sections of the IT service agreement. The answers to the questions were manually extracted and validated against the source documents to ensure precision and factual accuracy. This curated QA set provides a reliable if small basis for assessing the retrieval pipeline's ability to locate relevant context and for measuring the quality of generated answers against known correct responses.

2.2 Ethical & Legal Considerations

The contract corpus is subject to a non-disclosure agreement between the client and the service provider. Raw and processed documents never leave the company's secure drive; only aggregate statistics are stored in the experiment artefacts. Even so, to avoid unintentional disclosure of proprietary content, experiment artefacts can only be shared on request. This report and the codebase do not contain any proprietary text and can be shared publicly. No personal data were processed. Potential jurisdictional bias in the legal language is acknowledged as a limitation for generalising results.

3. Exploratory Data Analysis

3.1 Document-Level Metrics

To understand the structural variability of the documents in the corpus, six metrics were analysed: number of chapters, paragraphs, sentences, tokens, words, and characters. These metrics form the basis for designing chunking strategies and estimating processing load for downstream tasks.

Table 1 summarizes the basic statistics for each metric across the corpus. Documents vary significantly in length and structural complexity, with some documents comprising hundreds of sections and others being near-empty placeholders.

Table 1: Summary statistics for document structure metrics.

Metrics	Chapters	Paragraphs	Sentences	Tokens	Words	Characters
min	0	1	1	7	3	25
mean	12	286	704	32552	16631	118719
std	33	978	2123	83054	47147	317521
max	263	7431	13419	478477	302616	1950905

This high variance is particularly pronounced in the chapter and paragraph counts, where standard deviations exceed means, indicating a long-tailed distribution. The minimum values confirm that some documents contain no substantive content.

The distribution of document metrics was visualized in two forms: standard histograms and violin plots with log-scaled axes.

The histograms in Figure 1 show the skewed nature of the corpus: most documents are relatively short, but a few are extremely long. This asymmetry impacts retrieval granularity and motivates the need for adaptive chunking strategies.

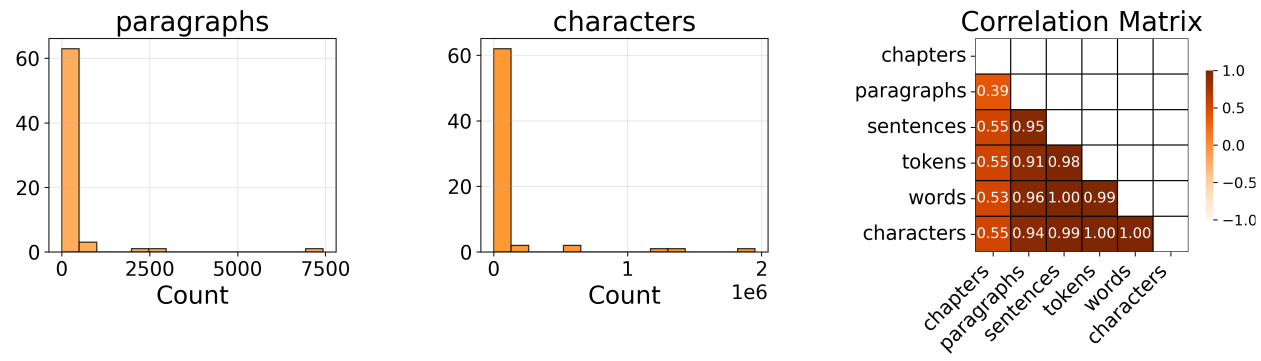

Figure 1: Histograms of chapter, paragraph, and character counts across documents

To better visualize long-tailed distributions, violin plots on a logarithmic scale were used (Figure 2). These emphasize the central tendency and spread while preserving visibility of extreme values. The character count distribution spans over six orders of magnitude, further justifying the need for careful normalization.

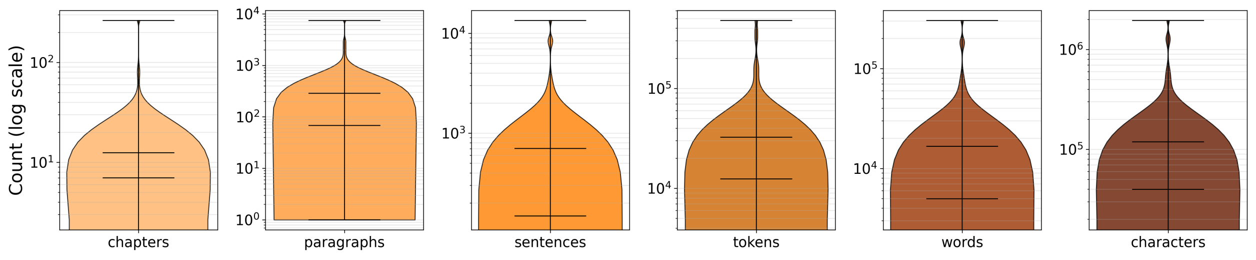

Figure 2: Log-scaled violin plots and correlation matrix of document metrics

The pairwise correlation matrix reveals strong linear relationships between most metrics, especially:

- Words - Tokens and Words - Sentences: confirming lexical consistency.
- Sentences - Paragraphs and Characters - Tokens: indicating reliable structural depth.

The weakest correlation is between chapter count and other metrics, suggesting that chapter boundaries do not consistently reflect document size or complexity. This insight supports the decision to de-emphasize chapters as primary chunking anchors in favor of sentence-based strategies.

3.2 Word Frequency Analysis

To understand the lexical distribution of the corpus, word frequencies were analysed across all 69 documents. Common stopwords were excluded to focus on semantically meaningful terms, and all tokens were lowercased. The resulting word-document frequency matrix forms the basis of the analysis presented below.

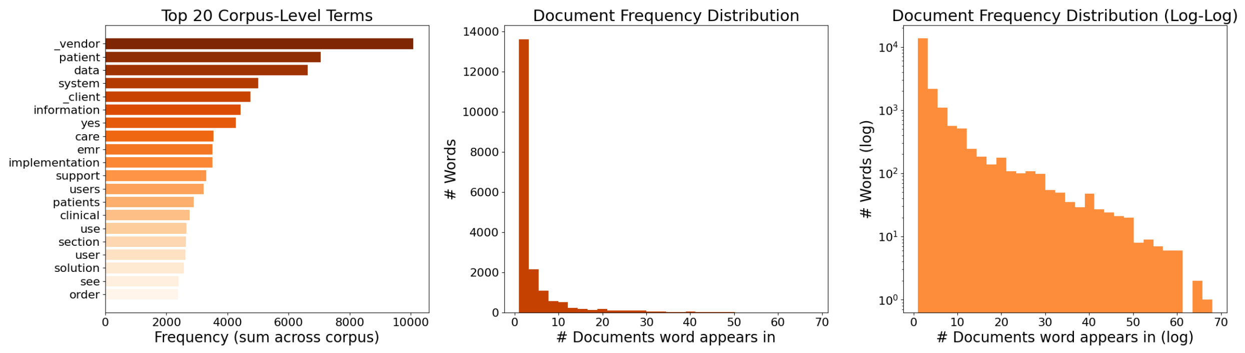

Figure 3: Word frequency analysis—top terms, document frequency histogram, and log-log distribution

The left panel of Figure 3 shows the 20 most frequent terms in the corpus after stopword removal. The names of the parties have been anonymized to "_vendor" and "_client".

The vocabulary is clearly domain-specific, with prominent terms including "patient", "system", and "care". The vocabulary is also rich and diverse, with a long tail of infrequent terms.

The middle panel of Figure 3 presents a histogram of document frequency—how many documents each word appears in. The distribution is heavily skewed: most words appear in only a small subset of documents, while a limited vocabulary is used corpus-wide. This supports the need for term-weighting strategies like TF-IDF [3] in retrieval pipelines, to avoid over-prioritizing ubiquitous but uninformative terms.

To further characterize this distribution, the right panel of Figure 3 shows the same data on a log-log scale. The approximately linear decay indicates a Zipf-like pattern [4], typical of natural language corpora. The long tail of infrequent terms confirms the corpus's lexical richness and supports the use of semantic embeddings that preserve rare but meaningful tokens.

3.3 t-SNE Visualization of Embedding Spaces

t-SNE projections [5] across different embedding models allow for a qualitative assessment of how well semantically similar content is clustered in a reduced 2D space. Figure 4 shows four open-source embedding model embeddings on the same set of document chunks generated using a token-based splitter with a chunk size of 384 tokens and 10% overlap. Each point represents a document chunk and is colored according to its contract document type (e.g., Master Service Agreement, Addendum, Annex).

Across all models, smaller document types such as Addendum and Schedule tend to form compact and well-separated clusters, indicating strong internal semantic consistency. In contrast, larger classes like Annex show more dispersed patterns, reflecting greater variability in content. While all models capture some degree of topical separation, the clarity and compactness of clusters vary, suggesting differences in how effectively each model encodes domain-specific semantics.

This t-SNE visualization confirms that the choice of embedding model directly affects the quality of semantic clustering. The strong clustering of less frequent types like Addendum and Schedule across most models

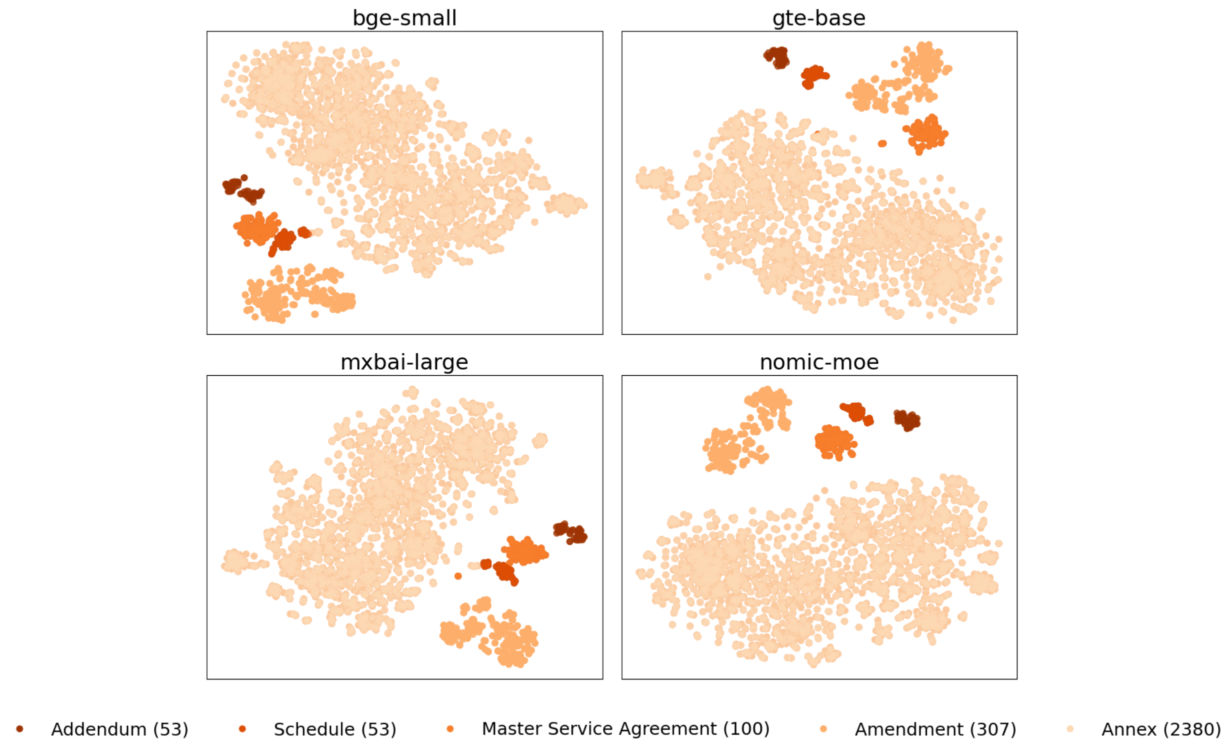

Figure 4: t-SNE visualization of chunk-level embeddings across different embedding models of a representative subset of the corpus.

illustrates that these categories carry distinctive semantic signals effectively captured by modern embedding architectures. However, performance varies with model capacity and training corpus—highlighting the importance of empirical validation when selecting embeddings for enterprise-scale document understanding tasks.

This qualitative assessment complements quantitative evaluations discussed in later sections by providing visual, intuitive insight into the strengths and weaknesses of different embedding spaces, reinforcing model selection decisions for the production RAG system.

4. System Architecture and Evaluation Setup

Building on the structural and lexical insights gained from exploratory analysis, this chapter describes the design of the end-to-end question answering system and the methodology used to evaluate its performance. The design goal is to create a QA system with a roundtrip latency of less than one minute and a graded accuracy of at least 80%.

4.1 System Architecture

The system architecture implements a modular, locally hosted RAG pipeline optimized for legal document question answering under compute constraints. It is built around the principle of maximizing semantic retrievability and answer quality using lightweight components. All experiments were executed with the open-sourced codebase available on Github [6].

The RAG pipeline is composed of the following stages (bold terms refer to components shown in Figure 5):

- **Document Chunking** The pipeline starts with the ingestion and parsing of the full IT service agreement. The documents are split into smaller, semantically meaningful **chunks** using LlamaIndex. This ensures each chunk contains a coherent unit of information that can independently answer a query.

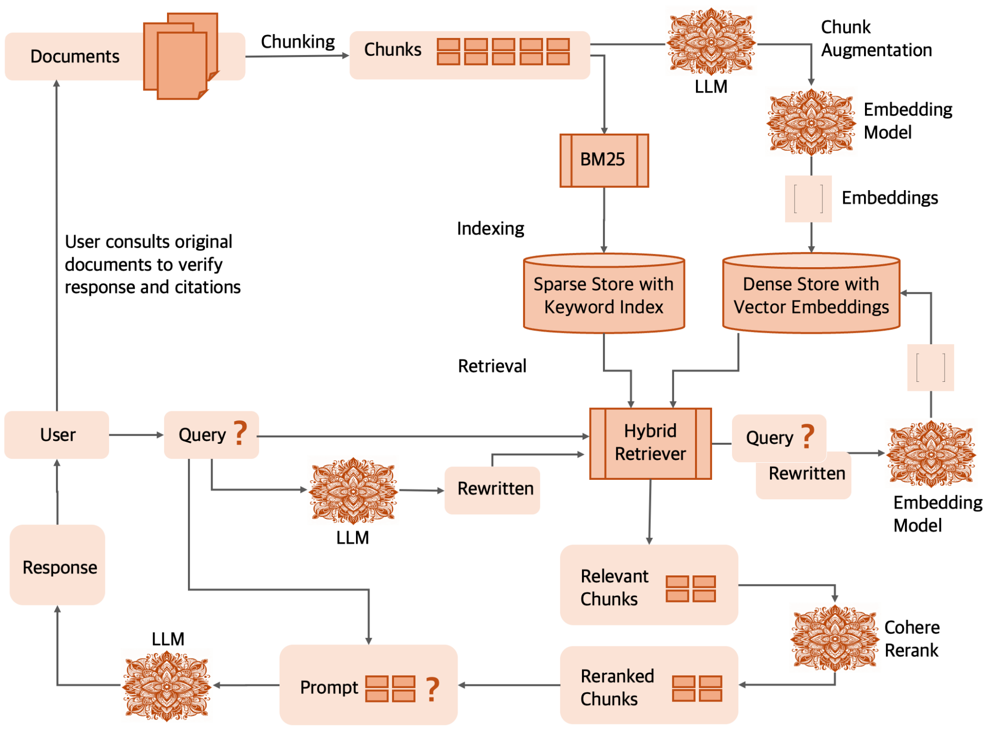

Figure 5: System Architecture of the RAG pipeline integrating hybrid retrieval, chunk augmentation, and reranking

- **Indexing** The system builds two parallel retrieval stores:
  - A **Sparse Store** using the BM25 algorithm [7] for keyword-based indexing and scoring.
  - A **Dense Store** that holds vector embeddings generated by a local embedding model. It is implemented with LlamaIndex SimpleVectorStore. During indexing, a very basic **Chunk Augmentation** is optionally applied: for each chunk, a local LLM generates multiple synthetic questions that the chunk could plausibly answer. These questions are prefixed to the chunk text and stored as embedding, potentially enhancing retrievability by increasing semantic overlap with future queries.
- **Query Handling** When a user submits a query, it gets passed through a local LLM for **query rewriting**, which reformulates or clarifies the original input to better match the indexed content. The original or rewritten query is then embedded and sent to both the sparse and dense stores via a **Hybrid Retriever**.
- **Retrieval and Reranking** The hybrid retriever combines scores from both stores to return a list of **relevant chunks**. These are reranked using **Cohere Rerank**, a transformer-based model that refines the relevance order based on the full text of the query and chunks [8] [9].
- **Prompt Construction and Response Generation** Reranked chunks are assembled with the query into a **prompt**, which is passed to a local LLM for final **response generation**. The resulting answer is returned to the user.
- **Verification Loop** Users are encouraged to consult the original documents, which are accessible and cited in the output, to verify the response and validate the context.

This architecture supports modular experimentation: components such as embedding models, LLMs, chunkers, and rerankers can be swapped or tuned independently. All operations run locally under strict data governance, ensuring compliance with confidentiality requirements. The chunk augmentation technique significantly boosts recall and answer relevance while staying within the compute budget of on-premises infrastructure.

The initial choice of open source embedding models is informed by the current ranking of huggingface MTEB benchmark [10] and personal experience.

Five families of large language models are used in the system architecture. All are open source and served locally by Ollama [11]. The choice of LLM was based on popularity and availability of small distilled models.

The target platform for the experiments is a single local virtual machine without GPU, running on Ubuntu under a KVM hypervisor, using 64 GB of RAM and 32 Intel Xeon CPU cores. That machine is built on spare failover capacity and only temporarily available for this project.

4.2 Evaluation Setup

The system architecture has 26 hyperparameters that can be tuned, see Table 8 in the appendix. In the resource constrained environment, only a subset of this grid can be evaluated. This happens in multiple steps. After each step, the most promising configurations are picked for downstream generation and evaluation.

Evaluation of the RAG chatbot system was conducted at three levels: retrieval, generation and, as a sub-step, the graded accuracy (human-grounded benchmark) to select the final model. All layers were assessed using appropriate metrics to reflect practical trade-offs between accuracy and latency.

Retrieval Evaluation

Retrieval quality was evaluated using two core information retrieval metrics:

- Recall: Measures the proportion of relevant documents (e.g., those containing the correct answer) that are successfully retrieved within the top-k results [12].
- Mean Reciprocal Rank (MRR): Captures how early the first relevant chunk appears in the ranked list [13].

Retrieval experiments were not static but part of a broad sweep with four embedding models, three chunking strategies and varying chunk sizes. Overlap was fixed at 10% of chunk size. The two most promising configurations were used in downstream QA generation and evaluation.

Generation Evaluation

Reference-based metrics were not used to assess surface-level answer quality. While fast and broadly used, BLEU [14] and ROUGE [15] often fail to capture deeper factual correctness and semantic relevance — hence the need for graded evaluation [16], [17]. Therefore, a Graded Accuracy (GA) framework was created, implementing the LLM-as-a-judge pattern [18] to evaluate the quality of generated answers.

A curated set of 84 QA records was constructed, each including a question, a ground-truth response, and a candidate answer. The candidate answers were generated by different LLMs at diverse temperatures. The candidate answers are then graded by human judges using a rubric that evaluated factuality, relevance, and completeness, assigning one of five qualitative ratings:

- `Wrong` The answer is mostly or completely incorrect, irrelevant, or misleading, score 0.0.
- `Poor` The answer has some relevance but contains factual inaccuracies or omits key information, score 0.25.
- `Fair` The answer is mostly correct but may lack completeness or precision, score 0.50.
- `Good` The answer is accurate and relevant with only minor omissions or imprecisions, score 0.75.
- `Perfect` The answer is factually accurate, complete, and fully relevant to the question, score 1.0.

LLMs were then asked to compare the candidate answers to the ground truth and provide a score based on the same rubric. Asking the LLMs to judge the candidate answers allowed us to evaluate the quality of the generated answers against known correct responses. For each configuration the following metrics were computed:

- RMSE between the model-generated score and the human score

- Pearson correlation with human judgment across examples
- Mean and standard deviation of inference time per model

This benchmark allowed to quantitatively rank models based on how closely their outputs aligned with human quality judgments. With this it was possible to automatically grade the accuracy of the generated answers of sweeped configurations.

5. Results and Analysis

This section presents and analyzes the outcomes of the RAG pipeline experiments across the evaluated configurations. This project evaluated RAG performance across a broad set of configurations combining:

- Three chunking strategies (token-based, sentence-based, semantic)
- Four embedding models (models see Table 6 in the Appendix)
- Five chunk sizes (256, 384, 512, 768, 1024 tokens)
- Sparse, dense, and hybrid retrieval
- Chunk augmentation and query rewriting
- Fourteen language models (models see Table 7 in the Appendix)

5.1 Dense Index Retrieval

Figure 6 presents a comparative analysis of how different chunk splitters—token-, sentence-, and semantic-based—affect Retrieval-Augmented Generation (RAG) pipeline performance across varying chunk sizes.

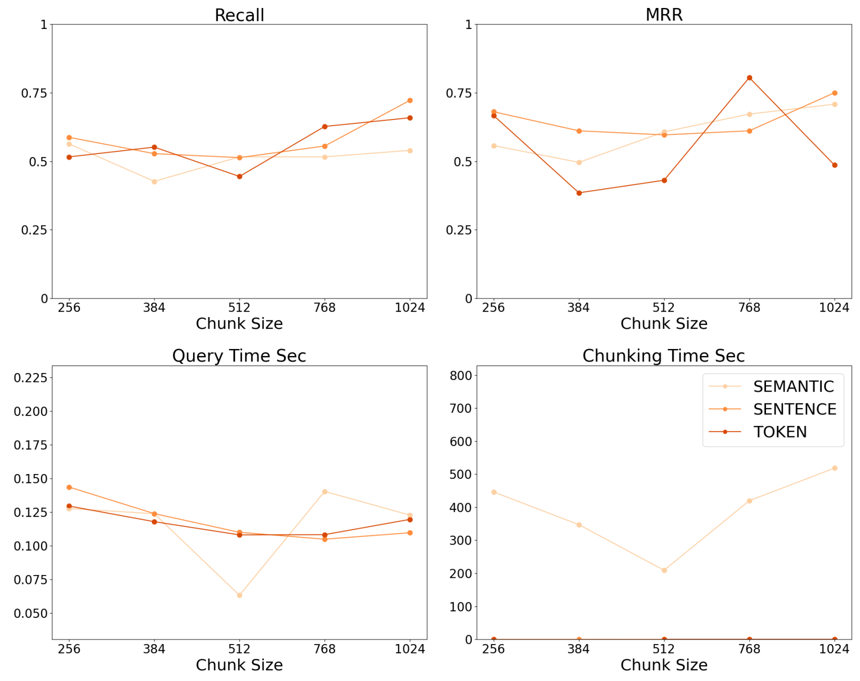

Figure 6: Impact of chunk splitter and chunk size on retrieval performance with bge-small embedding model, hybrid retrieval @8, and a fixed 10% overlap.

The sentence-based splitter on average outperforms the others across recall and MRR. Token-based splitting shows less stable retrieval quality, suggesting that fixed-size chunks without semantic boundaries limit retrievability and answer fluency. Semantic splitting shows the poorest performance, likely due to misaligned or inconsistently segmented content.

Latency is comparable for all splitter types. However, chunking time during ingestion is much longer for semantic splitting, likely due to the need to process and segment the content. Given these trade-offs, sentence-based splitting is chosen as the default strategy in the final pipeline, striking the best balance between retrieval effectiveness and runtime efficiency.

Figure 7 illustrates how the choice of embedding model influences the RAG pipeline's retrieval and generation quality across different chunk sizes.

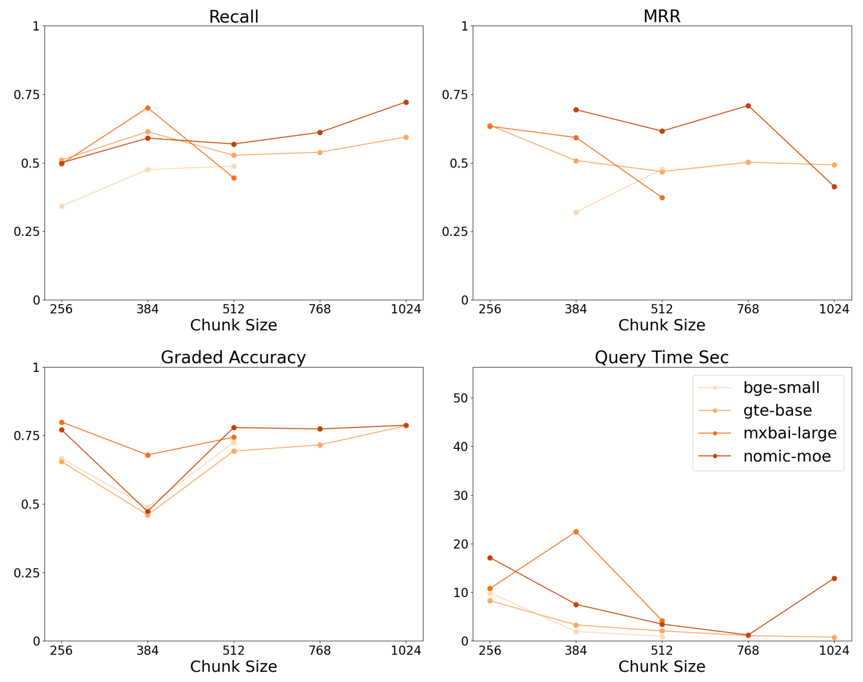

Figure 7: Effect of embedding model and chunk size on retrieval and QA performance. Sentence splitter, hybrid retrieval @8, fixed chunk overlap of 10%. bge-small and mxbai-large have a maximal chunk size of 512 tokens.

Across the larger chunk sizes, nomic-moe consistently outperforms the other models in both recall and MRR for dense retrieval [19]. Bge-small and gte-base achieve inferior performance, but are faster and more compact. The behavior of mxbai-large is interesting. It achieves the best performance at chunk size of 384 tokens, but is also the slowest model over all. Because sentence-based chunking takes the chunk size as a suggestion and not as a hard limit, the 512 token limit of mxbai-large is relevant. Even with 384 tokens, sentence-based chunking creates 10% of chunks that are longer than the 512 tokens. This happens because some large tables lack internal sentence boundaries. Content beyond the context window of mxbai-large is ignored and therefore lost to dense retrieval.

These results validate the hypothesis that embedding model choice significantly affects retrieval quality, and that latency–accuracy trade-offs must be considered when deploying RAG systems under local compute constraints. For the final pipeline, mxbai-large with a chunk size of 384 tokens and nomic-moe with a chunk size of 1024 tokens are used because they offer the best retrieval performance. This comes at the cost of longer generation times (up to a third of the allocated roundtrip time), but the trade-off is justifiable given the higher retrieval quality.

5.2 Hybrid Retrieval Results

This section evaluates the added value of hybrid retrieval [20] over sparse-only (BM25) and dense-only (vector) methods. The benchmark is based on the 20 evaluation questions defined in the chapter 2.1.

Figure 8 demonstrates that the hybrid approach consistently outperforms both baselines in terms of Recall@k:

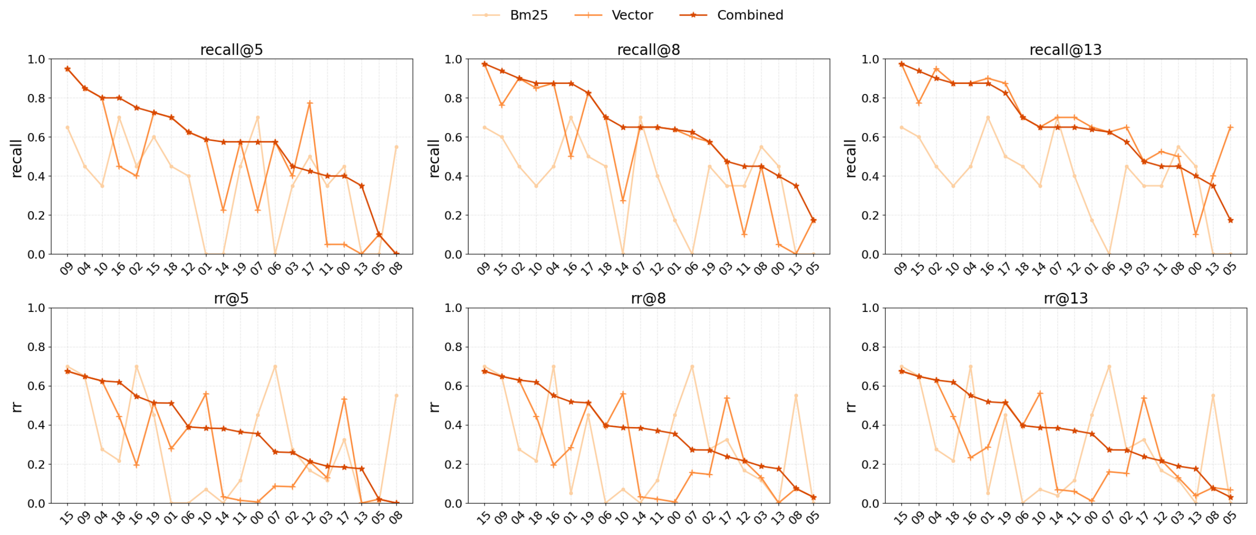

Figure 8: Retrieval effectiveness across the benchmark questions using Recall@k and Reciprocal Rank@k (k = 5, 8, 13). Questions are sorted in descending order of for the hybrid retriever y-value. Bge-small embedding model, sentence-based chunking, 378 tokens, 10% overlap.

- Recall@5 shows that the hybrid retriever achieves higher recall on most questions, indicating improved early retrieval performance.
- For Recall@8 and Recall@13, the hybrid method maintains a clear advantage, suggesting that combining sparse and dense signals yields more relevant results as the cutoff increases.

In terms of ranking quality, RR@5, RR@8, and RR@13 indicate that the hybrid retriever typically ranks relevant documents higher than either baseline. Although there is some variability across questions, the hybrid approach demonstrates consistently strong performance.

The relative performance of BM25 and vector retrieval varies significantly across individual questions, highlighting their complementary strengths. In contrast, the hybrid method exhibits both greater stability and higher effectiveness, especially in challenging cases where neither BM25 nor vector alone is sufficient. These results confirm the added value of combining sparse and dense retrieval signals.

5.3 Chunk Augmentation Results

To evaluate the effectiveness of chunk-level augmentation, this project compared the retrieval performance across two configurations—augmented and non-augmented—for two embedding models: Mxbai-Large (384 tokens) and Nomic-Moe (1024 tokens). Chunk augmentation involves prepending each chunk with several synthetic questions, automatically generated by prompting a local LLM to suggest plausible user queries for that chunk. The exact prompt used for question generation is included in the publicly available code repository.

Mxbai-Large exhibits a slight decrease in average MRR at all cutoff levels after augmentation (e.g., from 0.667 to 0.627 at k=8). The per-question plot reveals several instances where augmentation improves ranking, but also numerous cases where performance is unaffected or slightly degraded. This is interesting, because of the oversized chunks from the sentence tokenizer for mxbai-large. There was a hypothesis that prefixing questions about the entire content of the chunk, including the part that mxbai-large can not see, would have helped retrieval quality. This was not the case.

Nomic-Moe benefits more noticeably: average MRR improves from 0.375 to 0.583 at k=8, despite a small drop in recall. This suggests that augmentation helps surface relevant context higher in the candidate list—especially for questions the model previously struggled with. The token limit of 2048 tokens for nomic-moe is not a problem, the sentence tokenizer only exceeds it for 3 chunks.

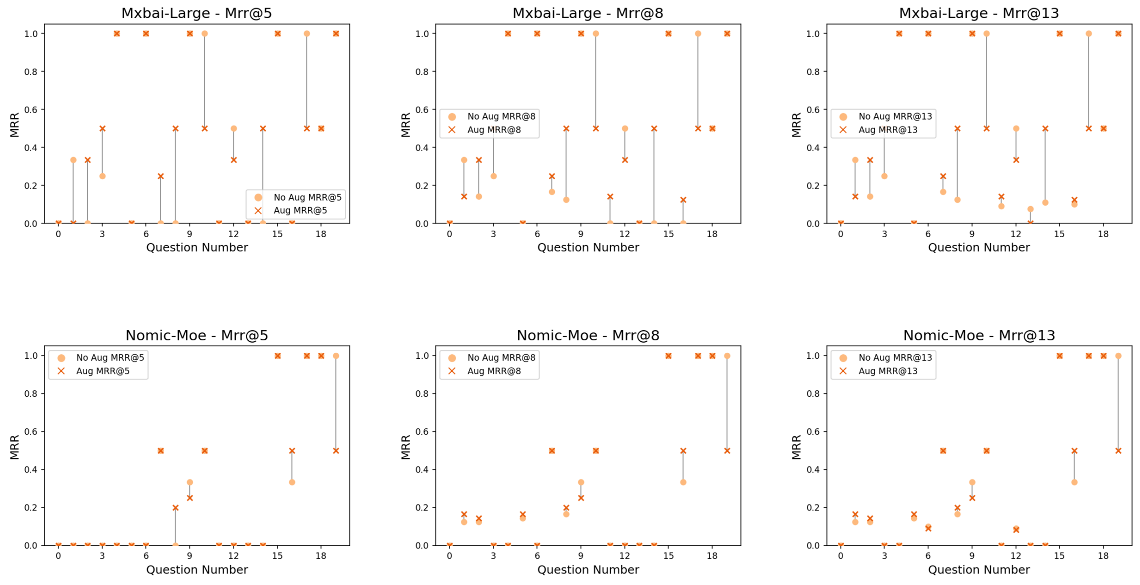

Figure 9: Chunk augmentation results for 20 benchmark questions using three core metrics at different cutoffs: Recall@k and Reciprocal Rank@k (k = 5, 8, 13)

Crucially, graded accuracy measured via LLM-as-a-judge scoring improves for both models post-augmentation. Mxbai-Large sees a modest gain from 0.925 to 0.975, while Nomic-Moe increases from 0.908 to 0.950. This indicates that even when retrieval quality (MRR) fluctuates, the downstream answer generation still somehow benefits. Note that the model generating the answer sees the original chunk without augmentation.

Table 2: Chunk augmentation results for benchmark questions using three core metrics at different cutoffs: Recall@8 and Reciprocal Rank@8, qwen3:4b model.

Embedding Model	Recall@8 non-augmented	Recall@8 augmented	MRR@8 non-augmented	MRR@8 augmented	Graded Accuracy non-augmented	Graded Accuracy augmented
mxbai-large	0.875	0.800	0.667	0.627	0.925	0.975
nomic-moe	0.733	0.675	0.375	0.583	0.908	0.950

5.4 Query Rewriting Results

The effectiveness of query rewriting was evaluated by comparing retrieval and question-answering performance for Mxbai-Large and Nomic-Moe, with the gemma3:4b model for rewriting and qwen3:4b for answer generation. Results are summarized in Table 3 and visualized in Figure 10.

Table 3: Query rewriting results for 20 benchmark questions using three core metrics at different cutoffs: Recall@8 and Reciprocal Rank@8, qwen3:4b model.

Embedding model	Config count	Avg recall no rewrite	Avg recall rewrite	Avg MRR no rewrite	Avg MRR rewrite	Avg GA no rewrite	Avg GA rewrite
mxbai-large	14.000	0.925	0.825	0.692	0.642	0.816	0.798
nomic-moe	14.000	0.700	0.750	0.375	0.374	0.754	0.778

The average metrics show mixed results. For mxbai-large, query rewriting decreased average recall (from 0.925 to 0.825), MRR (from 0.692 to 0.642), and graded accuracy (from 0.816 to 0.798). This suggests that the rewritten queries sometimes introduced semantic drift, reducing their alignment with the document content.

In contrast, the nomic-moe model benefits from rewriting across all metrics: recall increases from 0.700 to 0.750, and graded accuracy from 0.754 to 0.778, while MRR remains virtually unchanged. These results indicate that nomic-moe, a multilingual Mixture-of-Experts model, is more robust to the syntactic variability introduced by rewritten queries and potentially benefits from them in edge cases where the original phrasing was insufficient.

Figure 10 provides a question-by-question breakdown of graded accuracy with and without query rewriting. The scatter plot illustrates that while performance is stable for many questions, there are notable outliers where rewriting causes significant deterioration. A particularly illustrative failure is Question 8: "What languages does AVS support?" was rewritten to "Which languages are supported by the Avast antivirus software?". Here, the rewriter misinterpreted the domain-specific acronym (AVS) and substituted it with a well-known brand (Avast) anti-virus software, leading the model entirely off-topic. This highlights a critical risk of over-interpretation or hallucination by the rewriting LLM, especially when the original query contains domain-specific terms not covered in its training data.

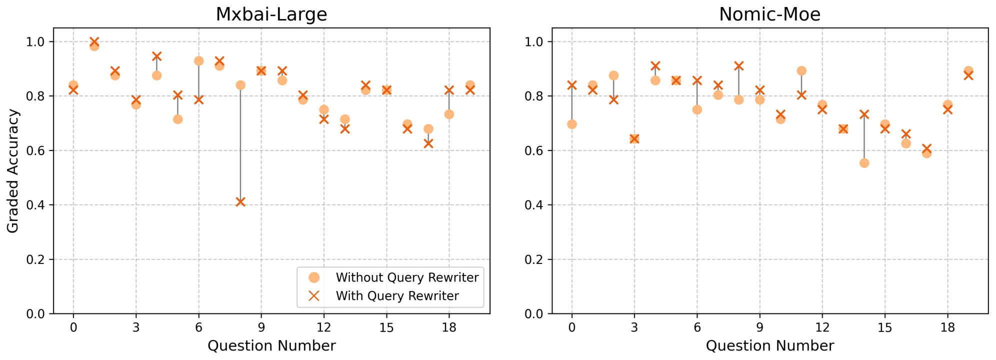

Figure 10: Effect of query rewriting on graded accuracy

Overall, these findings suggest that query rewriting can offer modest gains—especially when the base query lacks clarity—but it also introduces the risk of semantic misalignment. The benefit varies significantly by embedding model. Further tuning of the prompt template and model selection for rewriting might reduce the risks.

5.5 Graded Accuracy

The goal of this evaluation was to assess the quality of LLM-generated answers against human-graded scores. This enabled selection of the model and temperature configuration that produced the most human-aligned outputs while maintaining reasonable inference time. Table 4 shows the results. Temperature 0.1 produced lower RMSE and often higher correlation with all models (experiments were done with temperatures 0.1 and 0.7).

Table 4: Graded Accuracy evaluation for 11 models at temperature 0.1. Lower RMSE and higher correlation indicate closer alignment with human-graded scores.

Model	RMSE	Correlation	Mean Time Sec	Stdev Time Sec
gemma3:1b	0.51	0.28	**0.95**	0.39
gemma3:4b	0.33	0.71	2.59	1.58
gemma3:12b	0.35	0.67	7.19	5.07
qwen3:1.7b	0.33	0.70	6.99	4.17
qwen3:4b	**0.31**	**0.75**	20.64	47.00
qwen3:8b	0.37	0.70	31.58	67.80
mistral:7b	0.43	0.29	4.98	4.11
mistral-nemo:12b	**0.31**	0.70	6.47	4.86
deepseek-r1:1.5b	0.53	-0.01	5.20	2.05
deepseek-r1:7b	0.37	0.57	22.89	44.52
deepseek-r1:8b	0.36	0.62	18.19	5.70

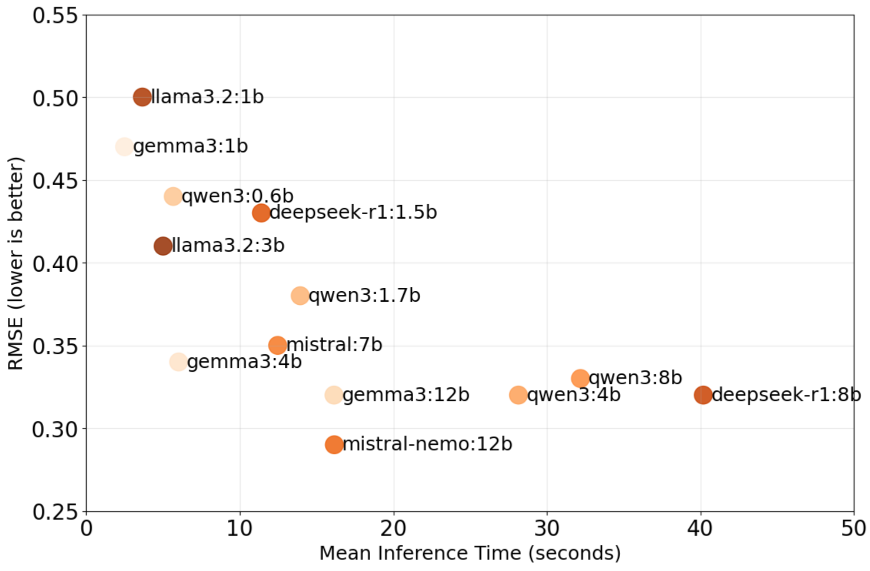

Figure 11: Model accuracy (RMSE) vs. mean inference time at temperature 0.1. Models in the lower-left quadrant are both fast and accurate.

Figure 11 plots the RMSE and inference times of the models at temperature 0.1. Qwen3:4b has the best correlation with human scores, but it is also four times slower than mistral-nemo:12b with the same RMSE. Again twice as fast is gemma3:4b, which as an only slightly higher RMSE and correlation. It presents a good trade-off between speed and accuracy.

Gemma3:4b is used as the LLM-as-a-judge for evaluating QA outputs during large-scale hyperparameter sweeps where human evaluation is infeasible. To avoid frequent loading of new models into Ollama during hyperparameter sweeps, gemma3:4b is also used as utility model for query rewriting and augmenting. Like this, only two models need to be loaded into Ollama at once, gemma3:4b and the LLM used for answer generation.

5.6 Generation Results

To conclude our evaluation, the trade-off between accuracy and latency across all model configurations was investigated. Figure 12 visualizes the Pareto front of Grade Accuracy versus average answer time per question, helping to identify configurations that are optimal in the sense that no other configuration achieves both higher accuracy and lower latency.

Figure 12 shows all evaluated models as grey circles, with the Pareto-optimal configurations highlighted in orange. Each orange point represents a model pair for which no other model outperforms it in both dimensions. The x-axis denotes average answer time in seconds, while the y-axis indicates accuracy on the Grade benchmark.

From this visualization, several insights emerge:

- Qwen models stand out: The `mxbai-large + qwen3:4b` and `mxbai-large + qwen3:1.7b` combinations consistently perform well, offering strong accuracy with moderate latency.
- Top-end accuracy: The highest accuracy is achieved by `nomic-moe + deepseek-r1:8b`, which also exhibits one of the highest latencies. This model serves as an upper bound in terms of performance but is a less practical choice for real-time applications due to response delay.
- Lightweight efficiency: Configurations like `mxbai-large + qwen3:0.6b` deliver acceptable accuracy at very low latency, making them suitable candidates for use cases where speed is critical and minor sacrifices in accuracy are acceptable.

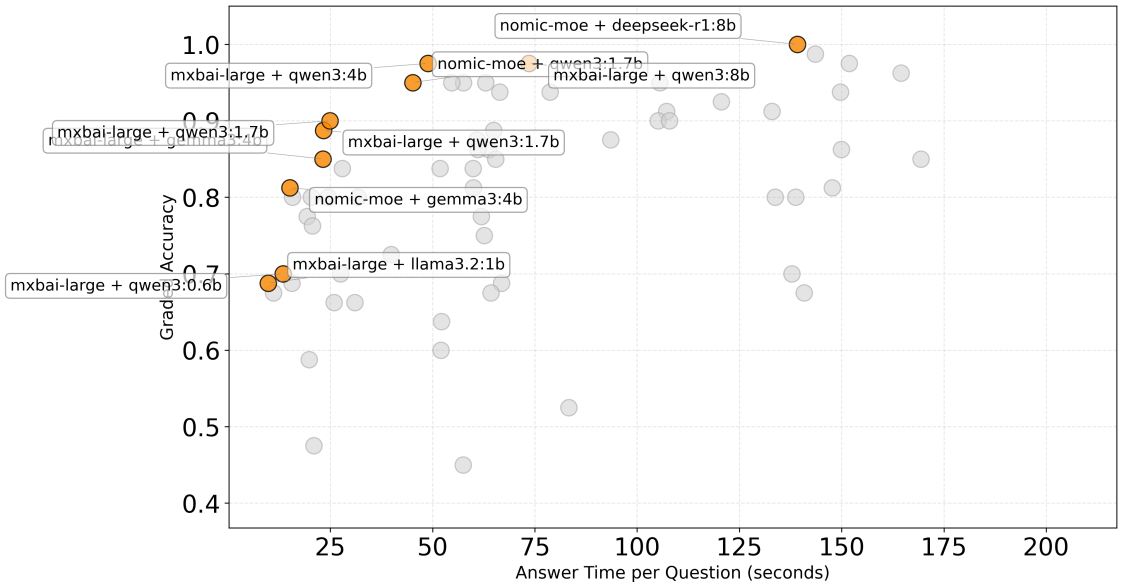

Figure 12: Model performance comparison showing graded accuracy versus retrieval and generation time. The nomic-moe embeddings paired with deepseek-r1:8b and qwen3:8b achieve the highest accuracy (near 1.0) but with longer processing times, while mxbai-large combinations offer faster performance at lower accuracy. Optimal model-embedding pairs can be found in the upper-left region, balancing high accuracy with reasonable processing times.

Overall, the Pareto front supports a strategic selection of configurations depending on system constraints. Applications requiring low-latency responses can opt for smaller models on the frontier, while scenarios prioritizing high accuracy might accept longer response times.

Table 5: Pareto front of Grade Accuracy versus average answer time per question. Answer time is the combined time to retrieve chunks with hybrid retriever and generate answer for one question.

Models	MRR	Graded Accuracy	Augment Chunks	Query Rewrite	Answer Time (sec)
nomic-moe + deepseek-r1:8b	0.374	**1.000**	False	True	139
mxbai-large + qwen3:4b	0.627	**0.975**	**True**	**True**	**48**
mxbai-large + qwen3:8b	0.627	0.975	True	True	73
nomic-moe + qwen3:1.7b	0.583	0.950	True	True	45
mxbai-large + qwen3:1.7b	0.662	0.900	False	True	24
mxbai-large + qwen3:1.7b	0.692	0.887	False	False	23
mxbai-large + gemma3:4b	0.463	0.850	False	False	23
nomic-moe + gemma3:4b	0.621	0.812	False	True	15
mxbai-large + qwen3:0.6b	0.662	0.700	False	True	13
mxbai-large + llama3.2:1b	0.692	0.688	False	False	**9**

This final table summarizes the exact metrics of the Pareto-optimal configurations. These findings serve as a foundation for informed deployment decisions, providing concrete guidance on how to navigate the trade-offs between model quality and responsiveness. The best performing configuration with an answer time under one minute is `mxbai-large + qwen3:4b` with a MRR of 0.627 and a graded accuracy of 0.975 and an answer time of 48 seconds.

Due to practical constraints, this analysis does not include confidence intervals or statistical significance tests. The reported metrics should thus be considered indicative, based on limited sample sizes (20 retrieval questions, 84 generation pairs), without quantification of variability or statistical certainty.

6. Conclusion and Outlook

This project set out to evaluate a Retrieval-Augmented Generation (RAG) pipeline for legal question answering under compute constraints, using a large and heterogeneous IT service agreement as testbed. The key goal was to determine how pipeline design choices—particularly chunking strategies, embedding models, and retrieval configurations—affect system performance in realistic enterprise settings.

Several insights emerged from the experiments:

- Pipeline design matters: Sentence-based chunking with empirically tuned token sizes (384, 1024 tokens) proved optimal, balancing semantic coherence with retrieval granularity.
- Hybrid retrieval is superior: Combining sparse (BM25) and dense (embedding-based) retrieval consistently outperformed either method alone in both recall and ranking quality. This robustness makes hybrid retrieval especially appealing in noisy or heterogeneous data environments.
- Augmentation improves retrieval only marginally: Prepending synthetic questions to chunks only sometimes improved retrievability.

- Query rewriting has trade-offs: While query rewriting can increase recall by aligning user input with document language, it can also introduce errors if the reformulation adds irrelevant context. Prompt design for rewriting remains a sensitive parameter.
- Model choice defines capability frontier: The best-performing configuration (nomic-moe embeddings with deepseek-r1:8b) reached perfect accuracy but incurred high latency. Mid-sized models like qwen3:4b offered a more practical trade-off with ~98% accuracy at much lower runtime.
- Together, these findings demonstrate that high-quality legal QA is achievable even under severe hardware limitations, provided that the pipeline is carefully constructed and tuned.

Limitations of this work:

- The evaluation focused on a single contract type (IT service agreement), where questions tend to be practical rather than strictly legal. Results may not generalize to more formal legal domains (e.g., employment law).
- Metrics are point estimates; no confidence intervals or statistical tests; ground-truth sets are small (20 retrieval Qs, 84 generation Q-A pairs).
- Large sections of the hyperparameter grid remain unexplored. There are many more configurations to try, but the time or resources required to evaluate them are prohibitive.

Several directions remain open for extension:

- Multilingual and multi-contract expansion: The system could be extended to support multiple contracts and languages.
- Active learning and continual improvement: Incorporating user feedback and ground truth expansion into an active loop could drive ongoing model refinement.
- Agent integration: The system could be extended such that the generating model can retrieve additional context from the contracts and use it to generate a more accurate answer.

This project demonstrates that practical RAG systems for contract analysis can be built and evaluated entirely on modest hardware without sacrificing effectiveness. Through systematic benchmarking and judicious use of open-source models and tools, it is possible to support practically relevant question answering in confidentiality-sensitive environments. This not only empowers service management professionals but also showcases a replicable methodology for other document-heavy domains.

7. Acknowledgements

The author would like to thank:

- The CAS programme faculty and fellow students for both a valuable education and an enjoyable experience.
- Ana and Matteo for their inspiring presentation on chunking strategies, which helped shape the direction of this project.
- My supervisors and coworkers for their consistent support and encouragement throughout the project.
- The creators and maintainers of the many open-source toolkits and models that made this work possible.

Language refinement and re-phrasing were assisted by commercial LLMs (Anthropic, OpenAI); the authors take sole responsibility for the final text.

The system was developed following a structured prototyping approach, but the actual implementation emerged through “vibe coding”—an exploratory, improvisational coding style supported by the Windsurf IDE with AI assistance from Anthropic, Google, and OpenAI. All source code has been reviewed, tested, and remains the sole responsibility of the author.

References

[1] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” *arXiv preprint arXiv:2005.11401*, 2020.

[2] “Marker-pdf.” Available: https://pypi.org/project/marker-pdf/. [Accessed: May 14, 2025]

[3] K. Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[4] G. K. Zipf, *Human behavior and the principle of least effort.* Cambridge, MA: Addison-Wesley Press, 1949.

[5] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[6] R. Moser, “Lightweight Retrieval-Augmented Generation pipeline – code.” https://github.com/glacial-haws/LightweightRAG, 2025.

[7] S. Robertson and H. Zaragoza, “The probabilistic relevance framework: BM25 and beyond,” *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[8] Cohere Inc., “Cohere rerank API.” Software service accessible via https://cohere.com/rerank, 2025. Available: https://cohere.com/rerank

[9] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using siamese BERT-networks,” in *Proceedings of the 2019 conference on empirical methods in natural language processing (EMNLP)*, 2019, pp. 3973–3983.

[10] “MTEB: Massive text embedding benchmark.” Available: https://huggingface.co/blog/mteb. [Accessed: May 22, 2025]

[11] Ollama, “Ollama.” Available: https://ollama.com/. [Accessed: May 22, 2025]

[12] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval.* Cambridge, UK: Cambridge University Press, 2008.

[13] E. M. Voorhees and D. K. Harman, “Overview of the eighth text retrieval conference (TREC-8),” National Institute of Standards; Technology, NIST Special Publication 500-246, 1999.

[14] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the association for computational linguistics*, 2002, pp. 311–318.

[15] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Proceedings of the ACL workshop on text summarization branches out*, 2004, pp. 74–81.

[16] E. Sulem, O. Abend, and A. Rappoport, “BLEU is not suitable for the evaluation of text simplification,” in *Proceedings of the 2018 conference on empirical methods in natural language processing*, Association for Computational Linguistics, 2018, pp. 738–744. Available: https://aclanthology.org/D18-1081

[17] Y. Kim and J. Lee, “Reference and document aware semantic evaluation methods for text summarization,” in *Proceedings of the 28th international conference on computational linguistics*, International Committee on Computational Linguistics, 2020, pp. 5585–5597. Available: https://aclanthology.org/2020.coling-main.491

[18] H. Li, Q. Dong, J. Chen, *et al.*, “LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods.” 2024. Available: https://arxiv.org/abs/2412.05579

[19] V. Karpukhin *et al.*, “Dense passage retrieval for open-domain question answering,” in *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, 2020, pp. 6769–6781.

[20] Z. Ma *et al.*, “Hybrid passage retrieval with dense and sparse representations,” in *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 2021, pp. 2271–2276.

[21] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, “C-pack: Packaged resources to advance general chinese embedding.” 2023. Available: https://arxiv.org/abs/2309.07597

[22] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, “Bge-small-en-v1.5.” Available: https://huggingface.co/BAAI/bge-small-en-v1.5. [Accessed: May 22, 2025]

[23] X. Zhang *et al.*, “mGTE: Generalized long-context text representation and reranking models for multilingual text retrieval,” in *Proceedings of the 2024 conference on empirical methods in natural language processing: Industry track*, 2024, pp. 1393–1412.

[24] X. Zhang *et al.*, “[Gte-multilingual-base].” Available: https://huggingface.co/Alibaba-NLP/gte-multilingual-base. [Accessed: May 22, 2025]

[25] S. Lee, A. Shakir, D. Koenig, and J. Lipp, “Open source strikes bread - new fluffy embeddings model,” 2024. Available: https://www.mixedbread.ai/blog/mxbai-embed-large-v1

[26] X. Li and J. Li, “AnglE-optimized text embeddings,” *arXiv preprint arXiv:2309.12871*, 2023.

[27] X. Li and J. Li, “Mxbai-embed-large-v1.” Available: https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1. [Accessed: May 22, 2025]

[28] Z. Nussbaum and B. Duderstadt, “Training sparse mixture of experts text embedding models.” 2025. Available: https://arxiv.org/abs/2502.07972

[29] Z. Nussbaum and B. Duderstadt, “Nomic-embed-text-v2-moe.” Available: https://huggingface.co/nomic-ai/nomic-embed-text-v2-moe. [Accessed: May 22, 2025]

[30] DeepSeek-AI, “DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning.” 2025. Available: https://arxiv.org/abs/2501.12948

[31] DeepSeek, “Deepseek-r1.” Available: https://ollama.com/library/deepseek-r1. [Accessed: May 22, 2025]

[32] G. DeepMind, “Gemma: Open models built from the same research and technology used to create gemini.” https://ai.google.dev/gemma, 2024.

[33] Google, “gemma3.” Available: https://ollama.com/library/gemma3. [Accessed: May 22, 2025]

[34] M. AI, “Llama 3.2: Revolutionizing edge AI and vision with open, customizable models.” https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, 2024.

[35] Meta, “llama3.2.” Available: https://ollama.com/library/llama3.2. [Accessed: May 22, 2025]

[36] M. AI, “Mistral NeMo: Our new best small model.” 2024. Available: https://mistral.ai/news/mistral-nemo

[37] M. AI, “Mistral.” Available: https://ollama.com/library/mistral. [Accessed: May 22, 2025]

[38] Qwen, “Qwen3: Think deeper, act faster.” 2024. Available: https://qwenlm.github.io/blog/qwen3/

[39] Qwen, “qwen3.” Available: https://ollama.com/library/qwen3. [Accessed: May 22, 2025]

Appendix

Table 6: Embedding Models used for dense indexing.

Short Name	Embedding Model Full Name	Max Sequence	Reason for Initial Choice and References
bge-small	bge-small-en-v1.5	512 Tokens	Fast and small English-only embedding model, [21], [22]
gte-base	gte-multilingual-base	8192 Tokens	Best performing model on MTEB Legal leader board with less than 1 GB memory usage, [23], [24]
mxbai-large	mxbai-embed-large-v1	512 Tokens	Best performing model on MTEB Multilingual leader board with less than 1 GB memory usage, [25], [26], [27]
nomic-moe	nomic-embed-text-v2-moe	2048 Tokens	Multilingual Mixture of Experts embedding model, [28], [29]

Table 7: Language models used for response generation.

Family	Model	Max Sequence	References
deepseek-r1	deepseek-r1:1.5b	131072 Tokens	[30], [31]
	deepseek-r1:7b	131072 Tokens	
	deepseek-r1:8b	131072 Tokens	
gemma3	gemma3:1b	32768 Tokens	[32], [33]
	gemma3:4b	131072 Tokens	
	gemma3:12b	131072 Tokens	
llama3.2	llama3.2:1b	131072 Tokens	[34], [35]
	llama3.2:3b	131072 Tokens	
mistral	mistral:7b	32768 Tokens	[36], [37]
	mistral-nemo:12b	1024000 Tokens	
qwen3	qwen3:0.6b	40960 Tokens	[38], [39]
	qwen3:1.7b	40960 Tokens	
	qwen3:4b	40960 Tokens	
	qwen3:8b	40960 Tokens	

Table 8: Tunable hyperparameters of the best performing configuration `mxbai-large + qwen3:4b` with a MRR of 0.627 and a graded accuracy of 0.975 and an answer time of 48 seconds.

Parameter	Value	Description
`embedding_model`	`mxbai-large`	Indexing parameters
`splitter_type`	`SENTENCE`	Indexing parameters
`chunk_size`	`384`	Indexing parameters
`chunk_overlap_pct`	`0.1`	Indexing parameters
`use_cached_index`	`True`	Indexing parameters
`augment_chunks`	`True`	Augmenting parameters
`augmenting_model`	`gemma3:4b`	Augmenting parameters
`augmenting_temperature`	`0.7`	Augmenting parameters
`augmenting_max_tokens`	`1024`	Augmenting parameters
`similarity_top_k`	`8`	Retrieval parameters
`vector_top_k`	`13`	Retrieval parameters
`bm25_top_k`	`8`	Retrieval parameters
`deduplicate`	`True`	Hybrid retriever parameters
`rerank`	`True`	Hybrid retriever parameters
`crossencoder_model`	`cross-encoder/ ms-marco-MiniLM-L6-v2`	Hybrid retriever parameters
`query_rewrite`	`True`	Query rewrite parameters
`query_rewrite_model`	`gemma3:4b`	Query rewrite parameters
`query_rewrite_temperature`	`0.7`	Query rewrite parameters
`query_rewrite_max_tokens`	`950`	Query rewrite parameters
`llm_model`	`qwen3:4b`	Chatbot LLM parameters
`llm_temperature`	`0.1`	Chatbot LLM parameters
`llm_max_tokens`	`10000`	Chatbot LLM parameters
`evaluate_graded_accuracy`	`True`	Evaluation
`evaluating_model`	`gemma3:4b`	Evaluation
`evaluating_temperature`	`0.1`	Evaluation
`evaluating_max_tokens`	`4096`	Evaluation