

Improving query rewriting and document retrieval for an enterprise QA system

Master Computer Science: Artificial Intelligence

Name:	Tianjun Chen
Student ID:	s4433068
Date:	19/06/2026
Specialisation:	Artificial Intelligence
First Supervisor:	Suzan Verberne
Second Supervisor:	Peter van der Putten
Daily Supervisor:	Yumeng Wang

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
2333 CA Leiden
The Netherlands

Contents

1 Introduction **4**

2 Related Work **5**

2.1 Query rewriting and LLM-based reformulation 5

2.2 Retrieval methods for enterprise question answering 6

2.3 Retrieval-augmented generation 6

3 Methodology **7**

3.1 System Overview 7

3.2 Hybrid retrieval and vector indexing 7

3.3 LLM-based Adaptive Query Rewriting 8

3.3.1 Prompt template 9

3.4 Multi-query Retrieval and Weighted Fusion 9

3.5 Evaluation procedure 10

3.6 Prompt-based retrieval quality judge 11

4 Datasets and evaluation setup **11**

4.1 KB knowledge base corpus 12

4.2 All_Questions evaluation dataset 12

4.3 refined_chat query log dataset 13

4.4 FIQA open-source financial QA dataset 14

4.5 Retrieval evaluation setup and metrics 16

5 Experiments and results **17**

5.1 Experimental setup and baseline configuration 17

5.2 Comparison with the baseline on internal dataset 18

5.3 Ablation study on internal dataset 18

5.3.1 Rewrite-weight exploration 19

5.3.2 Retrieval backend ablation 19

5.3.3 Rewrite count and original-query ablation 20

5.4 Comparison on additional internal dataset 21

5.5 Comparison on open-source dataset 21

6 Discussion **22**

6.1 Reflection on results 22

6.2 Limitations 23

6.3 Future work 24

7 Conclusion **24**

A Prompt for adaptive query rewriting **29**

B Prompt for the retrieval quality judge **30**

Abstract

Enterprise question answering (QA) systems aim to provide employees and customers with efficient access to organizational knowledge, such as policies, procedures, and technical documentation. However, in real-world enterprise environments, user queries are often expressed in informal or incomplete language and may not align with the terminology used in internal documentation. This mismatch between user queries and document content can significantly reduce the effectiveness of document retrieval systems.

This thesis is conducted in collaboration with KPN and investigates methods for improving document retrieval in an enterprise QA system. In the current system, retrieval relies on a hybrid approach combining lexical retrieval (BM25) and semantic retrieval, but no query rewriting techniques are applied. In addition, retrieval performance is evaluated only indirectly through negative user feedback, without the use of systematic evaluation metrics or structured experimentation. Consequently, it is difficult to measure and improve retrieval quality in a principled way.

To address these limitations, this thesis applies and evaluates a combination of established retrieval and query-rewriting techniques in a real enterprise QA setting. Rather than introducing a new retrieval algorithm, the contribution lies in adapting prompt-based LLM query rewriting to the KPN retrieval pipeline, combining multiple rewritten queries with hybrid BM25 and dense retrieval, and fusing their ranked results using weighted reciprocal rank fusion. The pipeline is evaluated systematically using Recall@k, Hit@k, and normalized discounted cumulative gain (nDCG@k).

On the labeled All_Questions dataset (Dutch internal-company dataset), the proposed rewrite-only equal-weight method improves Hit@5 from 0.869 to 0.929, Recall@5 from 0.746 to 0.808, and NDCG@5 from 0.662 to 0.692 compared with the baseline. On FIQA (English open-source dataset), the same method improves Recall@5 from 0.339 to 0.398, NDCG@5 from 0.320 to 0.365, and Hit@5 from 0.527 to 0.596, suggesting that multi-query rewriting can generalize beyond the internal Dutch enterprise QA setting.

For KPN, these improvements mean that relevant knowledge base articles appear more often in the top-five retrieved context set used by the enterprise QA system. This can support more reliable downstream answer generation, reduce missing-evidence retrieval cases, and provide a structured benchmark for comparing future retrieval changes.

1 Introduction

This thesis studies document retrieval in an enterprise QA system developed in collaboration with KPN (a Dutch communications provider). The system is intended to help users find relevant internal knowledge base articles for customer-service and agent-support questions about KPN products and services, such as billing, internet connectivity, TV services, modem troubleshooting, and subscription management. In this setting, retrieval quality is important because retrieval-augmented QA systems depend on retrieved documents as external evidence for producing or supporting answers [17].

In practice, however, user queries often differ substantially from the terminology used in enterprise documentation. Users tend to formulate questions in natural language, frequently including conversational phrases, incomplete descriptions, or ambiguous references. As a result, there is often a mismatch between the wording used in user queries and the vocabulary present in the document collection. This query–document mismatch is a common problem in information retrieval and can reduce retrieval effectiveness when relevant documents use different terminology from the user query [23, 20].

This challenge is particularly relevant in the enterprise QA system developed by KPN, which serves as the context for this thesis. The current system uses a hybrid retrieval approach that combines BM25-based lexical retrieval with an embedding-based dense retriever for semantic similarity search [23, 15, 22]. While this combination provides a strong baseline, the system currently does not apply any form of query rewriting to transform user queries into retrieval-friendly formulations. Furthermore, the system lacks an extensive online evaluation framework. Retrieval quality is primarily assessed through negative user feedback and offline evaluation. This makes it difficult to objectively measure retrieval performance or to compare potential improvements.

Query rewriting has been proposed as a way to reduce query–document mismatch by reformulating the user query before retrieval [20, 11, 35, 31]. Recent work also shows that LLMs can generate query expansions or multiple reformulations to improve retrieval coverage [11, 18]. In this thesis, query rewriting is used to transform conversational or ambiguous questions into concise and retrieval-oriented search queries.

In this thesis, we investigate the value of LLM-based query rewriting for a real-world enterprise QA system. The work does not introduce a new retrieval algorithm; instead, it evaluates whether established techniques from query rewriting, hybrid retrieval, and rank fusion can improve retrieval quality when adapted to the existing KPN retrieval pipeline. The proposed method applies prompt-based query rewriting before retrieval and combines the retrieved results from multiple rewrites using weighted reciprocal rank fusion, a rank-fusion method for combining multiple ranked lists [5].

The main research contribution is the systematic validation of this combination in a production-oriented enterprise QA setting. We evaluate the approach on a labeled internal KPN dataset, a production-style query log, and the open-source FIQA dataset. Retrieval performance is measured using standard information retrieval metrics, including Recall@k, Hit@k and normalized discounted cumulative gain (nDCG@k). In addition to retrieval effectiveness, we discuss practical trade-offs such as additional system complexity, LLM usage, and potential latency and cost overhead.

The central research question guiding this thesis is:

- To what extent can LLM-based query rewriting improve document retrieval performance in an enterprise QA system?

To answer this question, the thesis investigates several additional research questions:

- How does multi-query rewriting affect performance compared with a baseline hybrid retrieval system?
- How do different rewrite-weight configurations affect the final fused retrieval ranking?
- What are the practical trade-offs of applying LLM-based query rewriting in an enterprise retrieval pipeline?

These questions are evaluated through experiments on a labeled internal dataset, an unlabeled production-style query log, and the open-source FIQA retrieval dataset.

The main contributions of this thesis are:

1. A structured evaluation framework for document retrieval in a real enterprise QA setting, using manually labeled ground-truth document identifiers from the internal KPN dataset.

2. A production-oriented validation of prompt-based LLM query rewriting for a Dutch enterprise QA system, showing how existing query rewriting, hybrid retrieval, and rank-fusion methods perform when adapted to the KPN retrieval pipeline.

3. A task-specific prompt design that generates one to three retrieval-oriented rewrites while preserving the language and intent of the original user question.

4. An empirical evaluation showing that the adapted multi-query retrieval pipeline improves top-five retrieval coverage and ranking quality on the internal KPN evaluation dataset, with an additional generalization experiment on the open-source FIQA dataset.

2 Related Work

The thesis builds on three areas of prior research: query rewriting, document retrieval, and retrieval-augmented generation.

2.1 Query rewriting and LLM-based reformulation

Query rewriting has long been studied as an approach to address the vocabulary mismatch problem in information retrieval (IR)[23, 20]. This problem occurs when users express their information needs using terms that differ from the terminology used in the document collection. Even when a relevant document exists in the corpus, it may not be retrieved if the query and the document use different words, levels of specificity, abbreviations, or domain-specific expressions. This issue is especially important in enterprise search, where users often formulate short or informal questions, while internal documents are written using procedural, technical, or organizational terminology.

Traditional query rewriting methods often focus on query expansion. Query expansion aims to add related terms to the original query so that the retriever has more opportunities to match relevant documents. Early approaches include thesaurus-based expansion, pseudo-relevance feedback, and statistical term co-occurrence methods[11, 29, 7, 35]. These methods are designed to increase recall by enriching the original query with terms that are likely to appear in relevant documents. However, query expansion can also introduce noise when added terms are too broad or only weakly related to the original information need. This creates a trade-off between improving recall and preserving query precision.

Recent advances in large language models (LLMs) have created new opportunities for query rewriting and reformulation. Instruction-tuned language models can transform informal or ambiguous user questions into more structured and retrieval-oriented formulations. Ma et al. [20] propose query rewriting for retrieval-augmented large language models and show that reformulated queries can improve downstream task performance. Jagerman et al. [11] study prompt-based query expansion with large language models and show that useful query expansions can be generated without additional supervised training data.

Recent work has also explored multi-query and feedback-based rewriting. Li et al. [18] introduce diverse multi-query rewriting for retrieval-augmented generation, where several reformulations are generated to improve retrieval coverage. RQ-RAG learns to refine queries for retrieval-augmented generation by improving the retrieval input before document retrieval [3]. RaFe uses ranking feedback to improve query rewriting, showing that retrieval signals can help guide better reformulations [21]. These methods are relevant because they optimize the query before retrieval. However, most of them either require training a rewriter or use feedback during optimization. In this thesis, I use a fixed prompt-based rewriter because the available KPN data does not contain supervised query-rewrite pairs.

Recent work has also studied LLMs as direct query rewriters for retrieval-augmented systems. Ma et al. propose a Rewrite-Retrieve-Read framework in which the query is reformulated before retrieval, and they further train a small rewriter to better align the query with the frozen retriever and reader [20]. Jagerman et al. show that LLMs can be prompted to generate useful query expansions without supervised training data [11]. More recent work extends this idea to diverse multi-query rewriting and feedback-based rewriting for RAG [18, 3, 21].

Generated text has also been used as an intermediate retrieval signal. Query2doc uses large language models to generate pseudo-documents for query expansion, while HyDE generates hypothetical documents to improve zero-shot dense retrieval [29, 7]. These approaches show that generated text can help bridge the gap between short user queries and longer document representations. However, this thesis focuses on generating multiple retrieval-oriented query rewrites rather than hypothetical documents. The proposed method also differs from training-based rewriting methods by using a prompt-based rewriting strategy without training a separate rewriter or relying on ranking feedback during optimization.

2.2 Retrieval methods for enterprise question answering

Enterprise question answering systems aim to help employees and customers access internal organizational knowledge, such as support documentation, policy guidelines, and technical manuals. In such systems, document retrieval plays a crucial role in identifying relevant knowledge base entries that can answer user queries. Retrieval in enterprise settings is challenging because user questions are often conversational or incomplete, while internal documents often use standardized titles, product-specific terminology, and procedural descriptions.

Lexical retrieval methods such as BM25 remain important because they are efficient and reliable for exact term matching. This is especially useful in enterprise search, where queries and documents may contain product names, error codes, abbreviations, or internal identifiers. BM25 is grounded in the probabilistic relevance framework [23], and retrieval benchmarks such as BEIR show that BM25 remains a strong baseline across heterogeneous retrieval tasks [28]. However, lexical retrieval can struggle when the query and relevant document express the same meaning using different surface forms.

Dense retrieval methods address this limitation by representing queries and documents as continuous vector embeddings. Dense Passage Retrieval demonstrated the effectiveness of dual-encoder retrieval for open-domain question answering [15]. Sentence-BERT introduced efficient sentence-level embeddings for semantic similarity search [22]. Later dense and interaction-based models such as ANCE and ColBERT further improved retrieval quality through hard negative mining and late interaction mechanisms [32, 16]. More recent embedding and retrieval models have continued this development. E5 introduced weakly supervised contrastive pre-training for text embeddings [30], while ColBERTv2 improved the efficiency and effectiveness of late-interaction retrieval [25].

Dense retrieval does not fully replace lexical retrieval. Dense retrievers can be less reliable for exact matching of rare terms, numbers, codes, or domain-specific expressions. For this reason, many practical systems use hybrid retrieval, combining lexical and dense retrieval signals. Score-level fusion combines normalized scores from different retrievers, while rank-level fusion combines ranked lists directly. The choice of fusion function and normalization strategy can affect final ranking quality [2]. Reciprocal Rank Fusion (RRF) is a commonly used rank-fusion method that combines ranked lists based on reciprocal rank positions and does not require raw scores from different systems to be directly comparable [5].

Efficient indexing is also important for practical dense retrieval systems. FAISS provides scalable similarity search over large vector collections and is widely used for vector retrieval implementations [14]. In this thesis, the retrieval backend combines BM25 and dense vector retrieval, and the ranked lists produced from multiple query rewrites are combined using weighted RRF.

2.3 Retrieval-augmented generation

Retrieval-augmented generation (RAG) systems use retrieved documents as external evidence for answer generation. The purpose of RAG is to ground the QA system in a specific document collection or domain, rather than relying only on the internal knowledge of a language model. This is especially relevant for enterprise QA, where the required information is often contained in internal knowledge base articles, procedures, or policy documents that may not be part of the model's training data. In a RAG pipeline, the system first retrieves relevant documents and then conditions the generated answer on the retrieved context. RAG was introduced as a framework for knowledge-intensive NLP tasks [17]. Related approaches such as REALM and Fusion-in-Decoder also show that retrieval can improve question answering by providing external evidence to the model [8, 9].

The quality of the retrieved context is critical in retrieval-augmented question answering. If relevant information is not retrieved, the answer generator may produce incomplete or unsupported answers. This is particularly important in enterprise QA systems, where answers should be grounded in internal knowledge base articles, procedures, or policy documents. In such settings, improving the top-five retrieved document set can be more useful than optimizing only the first retrieved document, because the downstream answer generation step can require several retrieved documents as context.

Recent RAG research has moved beyond the simple retrieve-then-generate pipeline. Atlas shows that retrieval-augmented language models can improve few-shot learning by conditioning generation on retrieved evidence from a large text index [10]. Adaptive-RAG argues that retrieval strategies should depend on question complexity, selecting different retrieval and generation behaviours for different types of questions [12]. These works support the broader view that retrieval-augmented systems benefit from adapting the retrieval process to the information need, rather than treating retrieval as a fixed one-step operation.

Recent work has further shown that retrieval-augmented systems should not treat retrieval as a fixed one-step process. Active Retrieval Augmented Generation introduces a setting in which the model actively decides when and what to retrieve during generation, rather than relying only on a single initial retrieval step [13]. Self-RAG similarly argues that retrieval should be adaptive: the model learns to retrieve passages on demand and to critique both retrieved evidence and its own generations through self-reflection [1]. These approaches are relevant to this thesis because they show that retrieval behaviour can be adapted to the information need instead of being fixed for all queries.

Other recent work focuses on making RAG more practical and robust. REPLUG treats the language model as a black box and augments it by prepending retrieved documents to the input, making retrieval augmentation applicable without modifying the underlying generator [27]. Corrective RAG addresses the problem that retrieved documents may be incomplete or irrelevant by introducing a retrieval evaluator and corrective retrieval actions [33]. These works highlight that RAG performance depends strongly on the quality of the retrieved context. This supports the focus of this thesis on improving the top-five retrieved document set before answer generation.

This thesis follows the same general direction but focuses on pre-retrieval query rewriting in an enterprise QA setting. The proposed method does not change the answer generator itself. Instead, it aims to improve the retrieved evidence by generating multiple retrieval-oriented rewrites, retrieving documents independently for each rewrite, and combining the resulting ranked lists. This focus is chosen for both practical and methodological reasons. The existing KPN QA system is accessed through an end-to-end API, while the internal retrieval and generation components of the production system are not fully exposed as separately controllable modules. As a result, this thesis evaluates the part of the pipeline that can be isolated and measured reliably: document retrieval. Improving the quality and coverage of the retrieved context is an important first step, since the answer generator can only produce grounded answers when relevant evidence is available in the retrieved documents. A direct evaluation of downstream answer generation is therefore left for future work.

3 Methodology

3.1 System Overview

Figure 1 gives an overview of the proposed LLM-based multi-query hybrid retrieval pipeline. The pipeline consists of two main parts. First, the document collection is prepared offline by chunking the documents and building both a sparse BM25 index and a dense vector index. Second, at query time, the user question is rewritten into multiple retrieval-oriented queries using an LLM. Each rewrite is retrieved independently using hybrid retrieval, after which the ranked results are combined using weighted reciprocal rank fusion. Finally, the fused chunk ranking is mapped back to document identifiers and evaluated against the ground-truth contexts.

The baseline system uses the same document corpus, chunking strategy, BM25 index, dense vector index, hybrid retrieval backend, and document-level evaluation procedure. The difference is that the baseline does not apply query rewriting. It submits the original user question directly to the hybrid retriever and returns the top-five unique document identifiers after mapping retrieved chunks back to their source documents. The proposed method keeps the same retrieval backend but replaces the single original query with multiple LLM-generated rewrites and fuses the resulting ranked lists. This setup allows the experiments to isolate the effect of query rewriting and rank fusion from changes in the underlying retrieval infrastructure.

3.2 Hybrid retrieval and vector indexing

The retrieval backend consists of a sparse lexical retriever and a dense semantic retriever. The sparse component uses BM25 over tokenized document chunks. This component is useful for matching exact terms, such as product names, error codes, abbreviations, and domain-specific terminology. The dense component represents each document chunk as a vector embedding and retrieves semantically similar chunks using vector similarity search.

For dense retrieval, chunk embeddings are generated using the embedding model available through the enterprise LLM infrastructure. The resulting vectors are normalized before indexing. The dense vector index is implemented using FAISS, which provides efficient similarity search over large vector collections [14]. In this implementation, the normalized embeddings are stored in a FAISS inner-product index. Since the vectors are normalized, inner-product search is equivalent to cosine similarity.

At query time, the input query or generated rewrite is embedded using the same embedding model and normalization procedure. The FAISS index returns the most similar chunk vectors, while BM25 returns the strongest lexical matches.

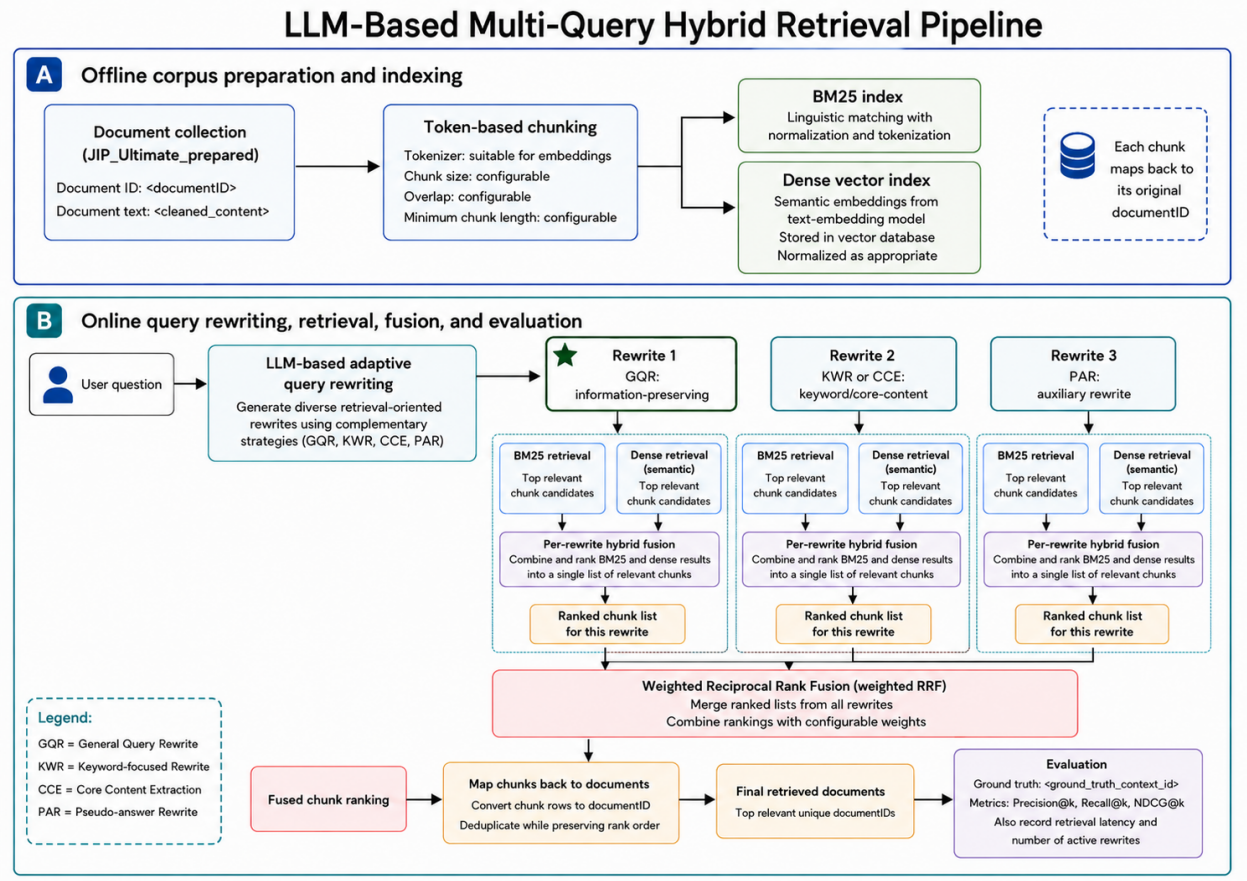


Figure 1: Overview of the proposed LLM-based multi-query hybrid retrieval pipeline. The offline stage chunks and indexes the document corpus, while the online stage rewrites the query, performs hybrid retrieval, fuses ranked chunk lists, and maps results back to document identifiers.

The two retrieval outputs are combined in the hybrid retriever before being passed to the later multi-query fusion step. This backend is shared by both the baseline and the proposed rewriting method, ensuring that the experiments compare different query formulation strategies rather than different retrieval infrastructures.

3.3 LLM-based Adaptive Query Rewriting

This section describes the main method-specific contribution of this thesis: a prompt-based adaptive query rewriting setup for enterprise document retrieval. The setup uses a single LLM prompt to generate one to three retrieval-oriented rewrites of the original user question. The goal is not to answer the question, but to produce alternative query formulations that preserve the original user intent while increasing the chance that the retriever matches the terminology used in the document collection.

Before document retrieval, an adaptive query rewriting step is introduced. The purpose of this step is not to answer the user question, but to reformulate it into a small set of retrieval-oriented rewrites. These rewrites are intended to preserve the original user intent while improving the chance that the retriever matches the terminology used in the document collection.

The query rewriting module is implemented using GPT-5[1]. For each input question, the model receives a prompt that instructs it to generate between one and three rewrites. The prompt is designed to be language-independent: the output language is required to match the language of the original question. As a result, Dutch questions in the internal KPN dataset produce Dutch rewrites, while English questions in FIQA produce English rewrites.

The number of rewrites is adaptive rather than fixed. This adaptiveness is implemented through a single prompt, not through an iterative loop or a separately trained decision model. The prompt instructs the LLM to generate between one and three rewrites depending on the complexity of the input question. If the original question is already short, precise, and retrieval-friendly, the model may output only one rewrite. If the question contains multiple important entities, actions, products, error codes, policy terms, or domain-specific concepts, the model may output two or three rewrites. The prompt also instructs the model to avoid near-duplicate rewrites and to generate additional rewrites only when they provide a different retrieval perspective. In the implementation, the output is parsed as a list of rewrites and constrained to the allowed range of one to three rewrites. Therefore, the adaptive behaviour is prompt-steered rather than based on an explicit rule-based condition check.

We use the four strategies as prompt-design categories rather than as separate trained rewriting models. Prior work on query rewriting, query expansion, diverse multi-query rewriting, and LLM-generated pseudo-documents for retrieval motivates these strategies [20, 11, 18, 29, 7].

- **GQR (General Query Rewrite):** an information-preserving rewrite that stays close to the original question. It keeps the full meaning, conditions, negations, symptoms, and important entities. This strategy is used as the default first rewrite.
- **KWR (Keyword-focused Rewrite):** a keyword-oriented rewrite that extracts the most important terms for lexical matching, such as entities, product names, codes, acronyms, symptoms, actions, and domain-specific terms.
- **CCE (Core Content Extraction):** a compressed rewrite that keeps the central retrieval intent, usually the main action and the main object. This is useful for task-oriented questions such as setup, configuration, reset, transfer, activation, or locating information.
- **PAR (Pseudo-answer Rewrite):** a short, neutral, document-like phrase resembling how a knowledge base article, FAQ, troubleshooting guide, policy page, or procedure might describe the topic. This strategy is only used as an auxiliary rewrite and is explicitly constrained not to invent solutions or facts.

The first rewrite is usually generated as a GQR rewrite because the prompt explicitly instructs the model to keep one information-preserving formulation close to the original question. The generation of additional rewrites is also steered through the prompt rather than through a separately trained classifier.

3.3.1 Prompt template

The full prompt template is provided in Appendix A. The query rewriting prompt is designed to generate one to three retrieval-oriented rewrites while preserving the original user intent. It instructs the model not to answer the question, to keep important entities and constraints unchanged, and to avoid near-duplicate rewrites. The prompt also defines four rewrite strategies: general query rewriting, keyword-focused rewriting, core content extraction, and pseudo-answer rewriting.

3.4 Multi-query Retrieval and Weighted Fusion

The proposed method extends the baseline hybrid retrieval system by replacing the single original query with multiple LLM-generated query rewrites. This design follows the general idea of rank fusion, where rankings produced by different retrieval runs or query formulations are combined into a single final ranking [5, 26]. Rank fusion has been used in information retrieval settings to combine evidence from multiple ranked lists, including applications where different query formulations or matching strategies produce complementary rankings [26].

The underlying hybrid retriever remains the same as in the baseline: each query is searched over the chunked corpus using both BM25 and dense retrieval, and the two retrieval signals are combined into a ranked chunk list. Therefore, the

[1]The model was accessed through a KPN-approved enterprise LLM environment. The use of this model for the experiments followed KPN's internal data-protection procedures and applicable supplier agreements. No personal customer data was intentionally included in the experimental prompts.

main difference between the baseline and the proposed method is not the retrieval model itself, but the use of multiple retrieval-oriented query formulations and their subsequent fusion.

Let q denote the original user question. After adaptive query rewriting, the system obtains a set of rewrites:

$$R(q) = \{q_1, q_2, \ldots, q_n\},$$

where $1 \leq n \leq 3$. Each rewrite q_i is submitted independently to the same hybrid retrieval pipeline described in the baseline section. This produces one ranked chunk list for each rewrite.

The motivation for this multi-query setup is that different rewrites may retrieve different relevant chunks. The first rewrite usually preserves the original question structure, while later rewrites may focus on keywords, core actions, entities, or document-like formulations. As a result, the different ranked lists provide complementary retrieval evidence.

To combine the ranked lists from the different rewrites, we use weighted Reciprocal Rank Fusion (weighted RRF). The weights control the relative importance of the ranked list produced by each rewrite. They are therefore treated as hyperparameters of the fusion step rather than as learned model parameters. A higher weight gives the corresponding rewrite more influence in the final fused ranking, while equal weights treat all rewrites as equally important retrieval views of the same information need.

In this thesis, we test a small manually defined weight grid for the three generated rewrites:

$$(1.0, 0.8, 0.6),\ (1.0, 0.7, 0.5),\ (1.0, 0.6, 0.4),\ (1.0, 0.5, 0.3),\ (0.9, 0.6, 0.3),\ (0.8, 0.6, 0.4),$$

$$(1.2, 0.8, 0.5),\ (1.2, 0.7, 0.3),\ (1.0, 0.4, 0.2),\ (1.0, 0.3, 0.1),\ (1.0, 0.0, 0.0),\ (1.0, 1.0, 1.0).$$

The grid is evaluated on the labeled `All_Questions` dataset in the rewrite-weight ablation. Since the downstream QA system uses a top-five context set, Recall@5 is used as the main selection criterion, while Hit@5 and NDCG@5 are used as supporting metrics. The equal-weight setting achieves the highest Recall@5 and Hit@5 in this grid, while also keeping the fusion strategy simple and reproducible. Therefore, the final proposed method uses:

$$w_1 = 1.0, \quad w_2 = 1.0, \quad w_3 = 1.0.$$

For each chunk c, the final weighted RRF score is computed as:

$$S(c) = \sum_{i=1}^{n} w_i \cdot \frac{1}{k + r_i(c)}$$

where w_i is the weight assigned to rewrite q_i, $r_i(c)$ is the rank of chunk c in the ranked list produced by rewrite q_i, and k is the RRF constant. If a chunk is not retrieved by a rewrite, it receives no contribution from that rewrite.

The original query is not included in the final fusion step. This choice reflects the goal of evaluating whether LLM-generated retrieval-oriented rewrites can serve as the main retrieval input rather than only as auxiliary expansions of the original query. The three rewrites are treated as complementary retrieval views of the same information need. Equal weighting avoids manually prioritizing one rewrite type over another and makes the final fusion strategy simple and reproducible.

The output of equal-weight RRF is a fused chunk-level ranking. As in the baseline, this ranking is mapped back to document identifiers using the stored chunk-to-document mapping. Duplicate document identifiers are removed while preserving rank order, so that each document appears at most once in the final output. The final retrieval result consists of the top five unique document identifiers.

3.5 Evaluation procedure

The retrieval pipeline outputs a ranked list of chunks for each input query or rewrite. Since the ground-truth labels are provided at document level, the chunk-level ranking is mapped back to document identifiers before evaluation. Duplicate document identifiers are removed while preserving rank order, and the final top-five unique documents are used for evaluation.

The evaluation metrics and dataset-specific evaluation settings are described in Section 4.5. This section therefore focuses only on how the retrieval pipeline produces the ranked document list that is later evaluated.

3.6 Prompt-based retrieval quality judge

LLM-based evaluators have recently been used as scalable judges for open-ended language tasks. G-Eval uses GPT-based evaluation to improve alignment with human judgments in natural language generation evaluation [19]. Zheng et al. [34] study LLM-as-a-judge evaluation in MT-Bench and Chatbot Arena, showing that LLM judges can be useful but may also introduce bias and inconsistency. For this reason, the prompt-based judge in this thesis is used only as a complementary quality estimate for unlabeled retrieval logs, not as a replacement for ground-truth evaluation.

For datasets without manually annotated relevance labels, such as the *refined_chat* query log dataset, standard retrieval metrics such as Recall@5, NDCG@5, and Hit@5 cannot be computed directly. To obtain an additional quality estimate for such unlabeled retrieval results, we use a prompt-based retrieval quality judge. This judge is not used as a replacement for ground-truth evaluation on labeled datasets, but as a complementary analysis method for retrieval settings where no reference document identifiers are available.

The judge is implemented as a pure prompting method. For each original user question, the prompt receives the original question, the rewritten queries used by the retrieval pipeline when available, and the final retrieved documents. The judge is explicitly instructed to evaluate only the original question and the retrieved documents, without using outside knowledge. This constraint is important because the goal is to assess whether the retrieved evidence itself is sufficient to support an answer.

The prompt asks the judge to evaluate five aspects of retrieval quality. The first score is an overall retrieval quality score, which reflects the general usefulness of the complete retrieved document set. The second score measures the relevance of the top-ranked document, since the first result is especially important in ranked retrieval. The third score measures coverage, indicating whether the retrieved documents contain the information needed to answer the question. The fourth score measures answerability, which evaluates whether a grounded answer could be generated from the retrieved documents. The fifth score measures evidence consistency, capturing whether the useful retrieved documents support a coherent answer rather than providing conflicting or confusing evidence.

Each score is assigned on a scale from 0.0 to 10.0 with exactly one decimal digit. The judge also assigns a categorical verdict based on the overall score and answerability score. A retrieval result is labeled as *good* when both the overall score and answerability score are at least 8.0. It is labeled as *acceptable* when both scores are at least 5.0. It is labeled as *poor* when either the overall score or the answerability score is below 5.0. In addition to the numeric scores and verdict, the judge outputs a short reason explaining the main retrieval issue or strength.

The output format is constrained to a single tab-separated row with seven columns: overall score, top-document relevance score, coverage score, answerability score, evidence consistency score, verdict, and main reason. This structured output makes the judge results easy to parse and compare across retrieval configurations. The full judge prompt is provided in Appendix B.

Since this method relies on an LLM-based judgment rather than human annotations, the resulting scores should be interpreted as approximate quality indicators. They are useful for comparing retrieval outputs on unlabeled data and for identifying common retrieval failure modes, but the labeled evaluation on *All_Questions* remains the primary quantitative evaluation in this thesis.

4 Datasets and evaluation setup

This study uses three KPN-internal datasets with different roles in the retrieval pipeline. The first dataset, `KB(knowledge base)`, forms the document corpus from which relevant knowledge base articles are retrieved. The second dataset, `All_Questions`, contains manually prepared user questions with ground-truth document identifiers and is used for quantitative evaluation. The third dataset, `refined_chat`, contains historical or production-style retrieval interactions without ground-truth labels. This dataset is not used for direct metric-based evaluation, but provides insight into realistic query behaviour and retrieval usage patterns.tri

The retrieval implementation uses `answerId`(ID of the query) from `All_Questions` as the document identifier, `cleaned_content` from `KB` as the document text, and `ground_truth_context_id` from `All_Questions` as the ground-truth field for labeled questions. These fields are also used in the retrieval code for corpus preparation, query evaluation, and metric calculation.

In addition to the company data, we also evaluate the method on the open-source FIQA dataset to test whether the retrieval pipeline generalizes beyond the enterprise QA setting.

Property	Value
Number of documents (documentID values)	2,534
Missing `cleaned_content` values	0
Empty `cleaned_content` values	0
Mean document length	3,469 characters
Median document length	2,326 characters
Mean document length	499 words
Median document length	332 words
Shortest document	25 characters
Longest document	292,059 characters

Table 1: Summary statistics of the `KB` corpus.

4.1 KB knowledge base corpus

The `KB` dataset is used as the main retrieval corpus. It contains the prepared knowledge base documents that the retrieval system searches over. Each row corresponds to one knowledge base article or information page. The most important fields are `documentId`, which uniquely identifies a document, and `cleaned_content`, which contains the cleaned textual content of the corresponding article.

The dataset contains **2,534 documents**. All `documentId` values are unique, and no missing or empty values were found in the `cleaned_content` field. This makes the dataset suitable as a document-level retrieval corpus.

The document lengths vary considerably. The mean document length is **3,469 characters**, with a median of **2,326 characters**. In terms of whitespace-separated words, the mean document length is **499 words**, while the median document contains **332 words**. The shortest document contains only **25 characters**, whereas the longest document contains more than **292,000 characters**. This large variation indicates that the corpus contains both short reference pages and long procedural or informational documents.

Because of this variation in document length, retrieval is not performed directly on full documents. Instead, we split documents into overlapping chunks before indexing. Each chunk keeps a mapping to its original `documentID`, allowing the system to retrieve at chunk level while evaluating and returning results at document level. This is important because long documents may contain several different topics or procedures, and chunk-level retrieval allows the retriever to match more specific passages.

A summary of the corpus is shown in Table 1.

The corpus mainly consists of Dutch customer-support and internal knowledge base material. It is heterogeneous in terms of specific support topics, including billing, internet connectivity, TV services, modem troubleshooting, eSIM activation, subscriptions, and internal support procedures. At the same time, it is relatively homogeneous at the broader domain level, since all documents belong to the KPN customer-support and internal knowledge-base setting. Many documents are structured using headings, instructions, bullet points, and links to related knowledge articles. This structure makes the corpus suitable for retrieval-based question answering, but also creates challenges. User questions may use conversational wording, while documents often use procedural titles, internal terminology, product names, or service-specific language. This motivates the use of query rewriting and hybrid retrieval in the later stages of the system.

4.2 All_Questions evaluation dataset

The `All_Questions` dataset is the labeled evaluation set used to measure retrieval performance. It contains **100 user(internal employees) questions**. Each question is associated with one or more ground-truth document identifiers in the `ground_truth_context_id` column. These ground-truth identifiers specify which knowledge base documents should be retrieved for the question. This dataset is a KPN-internal dataset and all questions were formulated and annotated by internal employees. The questions are therefore used as user questions in this study, because they reflect the type of support-related information needs that employees may submit to the enterprise QA system. The ground-truth document identifiers were manually assigned by internal annotators, indicating which KB knowledge base articles should be considered relevant for each question.

The questions are mainly task-oriented support questions rather than open-ended or abstractive questions. In most cases, the information need can be addressed by retrieving one relevant knowledge base article, making the task largely single-hop from a retrieval perspective. However, the dataset also contains multi-document cases: some questions are

associated with several ground-truth document identifiers, indicating that more than one knowledge base article may be needed to cover the complete support procedure or explanation. The questions mostly extractive and single-hop retrieval questions, with a smaller number of compositional or multi-document retrieval cases.

The key fields in this dataset are `question`, `ground_truth_answer`(generation answer), and `ground_truth_context_id`(document id). The `question` field contains the real user(KPN employee) question. The `ground_truth_answer` field contains the expected answer or reference answer, we do not use it to evaluate the retrieval performance. The `ground_truth_context_id` field contains the relevant document identifiers. This field is used as the main evaluation target for retrieval metrics such as Hit@1, Hit@5, Recall@5, and NDCG@5.

All 100 questions have a non-empty ground-truth context field. The number of relevant documents per question ranges from **1 to 5**. On average, each question has **1.76** ground-truth document identifiers. The distribution shows that **51 questions** have one ground-truth document, **32 questions** have two ground-truth documents, **10 questions** have three, **4 questions** have four, and **3 questions** have five. This means that the evaluation task is not always single-label retrieval. For many questions, multiple documents can be considered relevant.

Across all questions, there are **106 unique ground-truth document identifiers**. Of these, **105 identifiers** are present in the `KB` corpus, while **1 identifier** could not be matched to the corpus because of 1 missing document in the corpus file.

The questions themselves are relatively short and resemble real customer-support or agent-support queries. Table 2 shows examples from the `question` field, together with English translations.

Original Dutch question	**English translation**
Wat is KPN Pay?	*What is KPN Pay?*
Factuur klant klopt niet	*The customer's invoice is incorrect.*
hoe reset ik een 4g modem?	*How do I reset a 4G modem?*
welke knop bij het opnemen	*Which button should be used when recording?*
Hoe activeer ik een e-sim?	*How do I activate an eSIM?*

Table 2: Example questions from the `All_Questions` dataset with English translations.

The average question length is approximately **54 characters**, with a median of **48 characters**. In terms of words, the average question contains approximately **10 words**, with a median of **9 words**. The shortest question contains **4 words**, while the longest contains **31 words**. This reflects the short and often underspecified nature of practical support questions.

The dataset also contains columns from a previous retrieval or answer generation run, including retrieved context identifiers, model responses, latency, context recall, faithfulness, answer correctness, and helpfulness scores. These columns are useful for analysis and comparison, but the main retrieval evaluation in this study uses the `question` and `ground_truth_context_id` fields.

A summary of the evaluation dataset is shown in Table 3.

This dataset is used as the primary benchmark for retrieval experiments. For each question, the system retrieves the top-ranked documents and compares their `answerId` values against the ground-truth context identifiers. Because the dataset contains multiple relevant documents for some questions, Recall@5 and NDCG@5 are especially important. Recall@5 measures the fraction of ground-truth documents that appear within the top five retrieved documents, while NDCG@5 additionally rewards relevant documents that appear earlier in the ranking. The metric definitions are provided in Section 4.5. The KPN RAG system retrieves five documents for downstream answer generation, so the @5 is a relevant cut-off.

4.3 refined_chat query log dataset

The `refined_chat` dataset contains unlabeled query and retrieval interaction logs from the existing retrieval system. Unlike `All_Questions`, this dataset does not contain manually annotated ground-truth document identifiers. It contains conversational questions, short keyword-style searches, incomplete formulations, and repeated retrieval events within the same conversation. This makes it useful for understanding the types of queries that occur in practice and for motivating the need for robust query rewriting. However, because it lacks manually annotated relevance labels, it is not used as the primary dataset for quantitative evaluation of the proposed retrieval method. Therefore, it cannot be used

Property	Value
Number of questions	100
Questions with non-empty ground truth	100
Mean number of ground-truth documents	1.76
Median number of ground-truth documents	1
Minimum number of ground-truth documents	1
Maximum number of ground-truth documents	5
Unique ground-truth document IDs	106
Ground-truth IDs present in corpus	105
Ground-truth IDs missing from corpus	1
Mean question length	54 characters
Median question length	48 characters
Mean question length	10 words
Median question length	9 words

Table 3: Summary statistics of the `All_Questions` evaluation dataset.

directly for supervised retrieval evaluation with Hit, Recall, or NDCG. Instead, it provides a larger and more realistic view of how users or agents formulate queries in practice and how the original system responded to those queries.

A summary of the `refined_chat` dataset is shown in Table 4.

It is important to note that the retrieved documents and latency values in this dataset are outputs of the original retrieval system, not outputs of the retrieval pipeline proposed in this study. The fields `docs_retrieved`, `doc_titles`, `n_docs_retrieved`, `search_decision_latency`, and `retrieval_latency` describe the behaviour of the previous system at the time the logs were collected. They are therefore used only for descriptive analysis and for understanding the existing retrieval setting, not as evaluation results of the proposed method.

The dataset contains **7,660 retrieval records** from **5,257 unique conversations**. Each row represents one retrieval event. The most important fields are `conversation_id`, `triggering_query`, , `docs_retrieved`, `doc_titles`, and `n_docs_retrieved`. The `triggering_query` field contains the user query that triggered retrieval. The `docs_retrieved` field contains the document identifiers retrieved by the original system, and `doc_titles` contains the corresponding document titles.

Most retrieval events return five documents. 61 queries have empty **docs_retrieved**. In total, the dataset contains **1,215 unique retrieved document identifiers**. Of these, **1,207** are present in the `KB` corpus due to corpus file missing in KB dataset.

The query text in `refined_chat` is short and practical. The average triggering query length is approximately **55 characters**, with a median of **45 characters**. In terms of words, the average triggering query contains approximately **10 words**, with a median of **8 words**. Some rows have missing triggering or search queries: **71 rows** have missing triggering queries. These missing values reflect the noisier nature of production-style logs compared with the curated evaluation set.

The dataset also includes latency fields from the original system. The `search_decision_latency` field measures the time used by the original system to decide whether a search step should be triggered and how the search query should be formulated. The `retrieval_latency` field measures the time used by the original system to retrieve documents after the search query has been produced. The average search-decision latency is approximately **0.87 seconds**, while the average retrieval latency is approximately **2.04 seconds**. These values are useful for describing the behaviour of the original system, but they are not directly comparable to the latency of the proposed method unless both systems are evaluated under the same runtime conditions.

4.4 FIQA open-source financial QA dataset

To further evaluate whether the proposed query rewriting and retrieval pipeline generalizes beyond the Dutch enterprise QA setting, an additional experiment is conducted on the FIQA dataset. In contrast to the KPN datasets, FIQA is an English financial question-answering retrieval dataset. It contains natural-language financial questions, a document corpus consisting of financial answer passages, and relevance judgments that map each query to one or more relevant corpus documents. FIQA is used through the BEIR benchmark, which provides heterogeneous retrieval datasets across multiple tasks and domains [28].

Property	Value
Number of retrieval records	7,660
Unique conversations	5,257
Missing triggering queries	71
Missing search queries	129
Mean retrieved documents per record	4.97
Median retrieved documents per record	5
Minimum retrieved documents per record	1
Maximum retrieved documents per record	5
Unique retrieved document IDs	1,215
Retrieved IDs present in corpus	1,207
Retrieved IDs missing from corpus	8
Mean triggering query length	55 characters
Median triggering query length	45 characters
Mean triggering query length	10 words
Median triggering query length	8 words
Mean search-decision latency from original system	0.87 seconds
Mean retrieval latency from original system	2.04 seconds

Table 4: Summary statistics of the `refined_chat` query log dataset.

Property	Value
Number of corpus documents	57,638
Empty corpus documents	38
Non-empty corpus documents used for retrieval	57,600
Document identifier field	`corpus_id`
Document text field	`corpus_text`
Query identifier field	`query_id`
Query text field	`query_text`
Relevance label table	`qrels_all_csv`
Mean document length	132.9 words
Median document length	90 words
75th percentile document length	159 words
90th percentile document length	270 words
95th percentile document length	368 words
99th percentile document length	642 words
Maximum document length	2,973 words
Documents longer than 512 words	1,165

Table 5: Summary statistics of the FIQA corpus used for the open-source financial QA retrieval experiment.

In the prepared version used in this study, the corpus is stored in `corpus_fiqa`. Each document is identified by a `corpus_id`, and the document content is stored in the `corpus_text` field. The queries are stored in `queries_fiqa`, using `query_id` as the query identifier and `query_text` as the query text. The relevance labels are stored in `qrels_all_csv`, which combines the train, development, and test relevance files into a single table. Each row in this table links a `query_id` to a relevant `corpus_id` and includes a `split` field indicating the original data split.

The FIQA corpus contains 57,638 documents. Of these, 38 documents are empty and are removed before retrieval, leaving 57,600 non-empty corpus documents. The corpus is substantially different from the KB knowledge base corpus. While the KB corpus contains long Dutch knowledge base articles and procedural documentation, the FIQA corpus mainly consists of shorter English financial answer passages. The median document length is 90 words, and the mean document length is 132.9 words. The 95th percentile is 368 words, indicating that most documents are relatively short. Only around 2.0% of the documents are longer than 512 words.

A summary of the FIQA corpus is shown in Table 5.

4.5 Retrieval evaluation setup and metrics

The evaluation design is influenced by recent work on retrieval and retrieval-augmented generation evaluation. RAGAS proposes automated evaluation methods for RAG systems, including criteria related to context relevance, faithfulness, and answer quality [6]. ARES provides an automated evaluation framework for retrieval-augmented generation systems, including evaluation dimensions such as context relevance, answer faithfulness, and answer relevance [24]. The RGB benchmark studies retrieval-augmented generation under conditions such as noisy retrieved documents and the need to integrate retrieved evidence [4]. These works show that evaluating retrieval-augmented QA systems can involve both retrieval quality and downstream answer usefulness.

This thesis focuses on the retrieval stage rather than full downstream answer generation. This choice follows from the research focus on query rewriting: the proposed method changes the input to the retriever and the resulting retrieved document set, but it does not modify the answer generator itself. In addition, the production KPN QA system is accessed through an end-to-end API, which makes it difficult to isolate and evaluate the generation component independently under controlled conditions. Therefore, the main labeled evaluation in this thesis uses document-level retrieval metrics. The goal is to measure whether query rewriting places more relevant knowledge base articles in the top-five retrieved context set. For the unlabeled `refined_chat` logs, a prompt-based retrieval quality judge is used only as a complementary quality estimate.

The retrieval experiments are evaluated at document level. Although retrieval is performed over chunks, each retrieved chunk is mapped back to its source document identifier before evaluation. Duplicate document identifiers are removed while preserving the ranking order, and the final output for each query consists of the top five unique document identifiers.

For the internal *All_Questions* dataset, the retrieved document identifiers are compared with the manually provided ground-truth identifiers in the `ground_truth_context_id` field. Since a question can have more than one relevant document, the evaluation uses metrics that account for both retrieval coverage and ranking quality. The main metrics are Hit@1, Hit@5, Recall@5, and NDCG@5.

Hit@k measures whether at least one relevant document is retrieved within the top k results:

$$\text{Hit@}k = \begin{cases} 1, & \text{if } R_k \cap G \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

where R_k is the set of retrieved documents in the top k positions and G is the set of ground-truth relevant documents. Hit@1 evaluates whether the first retrieved document is relevant, while Hit@5 evaluates whether at least one relevant document appears anywhere in the top five results.

Recall@5 measures the fraction of all relevant documents that are retrieved in the top five results:

$$\text{Recall@5} = \frac{|R_5 \cap G|}{|G|}$$

This metric is important because several questions in *All_Questions* have multiple relevant ground-truth documents. A higher Recall@5 means that the retrieved top-five context set contains a larger proportion of the relevant documents.

NDCG@5 evaluates ranking quality by giving higher weight to relevant documents that appear earlier in the retrieved list. It is computed as:

$$\text{DCG@5} = \sum_{j=1}^{5} \frac{rel_j}{\log_2(j+1)}$$

where $rel_j = 1$ if the document at rank j is relevant and 0 otherwise. The normalized score is then computed as:

$$\text{NDCG@5} = \frac{\text{DCG@5}}{\text{IDCG@5}}$$

where IDCG@5 is the ideal DCG score for the query. NDCG@5 is useful because the order of retrieved documents matters in retrieval-augmented question answering: a relevant document ranked first is more useful than a relevant document ranked fifth.

Precision@5 is not used as a main metric for the internal *All_Questions* evaluation because the number of relevant documents differs across questions. Since the system always returns five documents, Precision@5 would be strongly affected by how many ground-truth documents are available for each query. For this reason, Hit@k, Recall@5, and NDCG@5 are more informative for the internal labeled evaluation.

For the FIQA experiment, the same top-five retrieval setting is used, but additional benchmark-style metrics are reported. These include Precision@5 and MRR@5, in addition to Recall@5, NDCG@5, and Hit@5. Precision@5 measures the proportion of the top-five retrieved documents that are relevant:

$$\text{Precision@5} = \frac{|R_5 \cap G|}{5}$$

MRR@5 measures the reciprocal rank of the first relevant document within the top five results:

$$\text{MRR@5} = \begin{cases} \frac{1}{r}, & \text{if the first relevant document appears at rank } r \leq 5 \\ 0, & \text{otherwise} \end{cases}$$

The scores are averaged over all evaluated queries. For *All_Questions* and FIQA, the main evaluation is based on ground-truth relevance labels. For *refined_chat*, manually annotated relevance labels are not available, so standard retrieval metrics cannot be computed directly. This dataset is therefore evaluated separately using the prompt-based retrieval quality judge described in Appendix B.

5 Experiments and results

5.1 Experimental setup and baseline configuration

The experiments are designed to evaluate whether LLM-based query rewriting improves retrieval performance over the existing hybrid retrieval pipeline. The baseline and the proposed method use the same document corpus, chunking strategy, BM25 index, dense vector index, and document-level evaluation procedure. This ensures that differences in performance can be attributed mainly to the query rewriting and rank fusion strategy rather than to changes in the underlying retrieval backend.

The baseline represents the current retrieval setting without query rewriting. For each question, the original user query is used directly as the only retrieval input. Candidate chunks are retrieved using the hybrid retriever, which combines BM25 and dense retrieval scores. The resulting chunk ranking is mapped back to document identifiers, duplicate document identifiers are removed while preserving rank order, and the top five unique documents are returned for evaluation.

The proposed method uses the same hybrid retrieval backend, but replaces the single original query with one to three LLM-generated rewrites. Each rewrite is retrieved independently, producing one ranked chunk list per rewrite. These ranked lists are then combined using reciprocal rank fusion. Based on the ablation results, the final selected configuration uses equal weights for all three rewrites and does not include the original query in the final fusion step.

Table 6 summarizes the main retrieval configurations evaluated in this chapter. The first configuration is used as the baseline. The second configuration is the final proposed method. The remaining configurations are used for ablation analysis to study the effect of rewrite weights, retrieval backend choices, the number of rewrites, and the inclusion of the original query.

All experiments on the internal labeled dataset are evaluated on *All_Questions*. For each question, the top five retrieved document identifiers are compared with the ground-truth document identifiers. The main reported metrics are Hit@1, Hit@5, Recall@5, and NDCG@5. Additional experiments are conducted on *refined_chat* using the prompt-based retrieval quality judge and on FIQA using the same final rewrite-only equal-weight configuration.

Configuration	Purpose
Original-query hybrid retrieval	Baseline system. The original user question is used directly as a single retrieval query without rewriting.
Rewrite-only equal-weight retrieval	Final proposed method. Generated rewrites are retrieved independently and fused with equal weights.
Rewrite-weight variants	Ablation setting used to compare different reciprocal rank fusion weights for the generated rewrites.
Backend variants	Ablation setting used to compare BM25-only, dense-only, and hybrid retrieval backends.
Original-query inclusion variants	Ablation setting used to test whether adding the original query to the generated rewrites improves the fused ranking.

Table 6: Overview of the retrieval configurations evaluated in the experiments.

Method	Hit@1	Hit@5	Recall@5	NDCG@5
Baseline hybrid retrieval	0.616	0.869	0.746	0.662
Our method	0.586	0.929	0.808	0.692
Absolute change	**-0.030**	**+0.060**	**+0.062**	**+0.030**
Relative change	**-4.9%**	**+7.0%**	**+8.3%**	**+4.5%**

Table 7: Retrieval performance of the baseline and proposed method on *All_Questions*.

5.2 Comparison with the baseline on internal dataset

This experiment evaluates whether the proposed LLM-based multi-query rewriting method improves retrieval performance compared with the original-query hybrid retrieval baseline. The experiment is conducted on the labeled *All_Questions* dataset, which contains manually provided ground-truth document identifiers. This dataset is used as the primary benchmark because it allows direct metric-based comparison between retrieval configurations.

The comparison is designed as a controlled experiment. Both the baseline and the proposed method use the same document corpus, the same chunking procedure, the same BM25 and dense retrieval indexes, and the same document-level evaluation procedure. The only difference is the query formulation strategy. The baseline uses the original user question directly as a single retrieval query. The proposed method uses the three LLM-generated rewrites as retrieval inputs, retrieves results independently for each rewrite, and combines the resulting ranked lists using equal-weight reciprocal rank fusion. The original query is not included in the final fusion step.

For each question, both systems return the top five unique document identifiers after mapping retrieved chunks back to their source documents. These document identifiers are compared against the ground-truth document identifiers in *All_Questions*. The main objective of this experiment is to test whether replacing the original query with multiple retrieval-oriented rewrites improves the quality of the top-five retrieved context set.

Table 7 reports the results of this comparison. The proposed method improves the top-five retrieval quality compared with the baseline. Hit@5 increases from 0.869 to 0.929, corresponding to a relative improvement of 7.0%. Recall@5 increases from 0.746 to 0.808, corresponding to a relative improvement of 8.3%. NDCG@5 also increases from 0.662 to 0.692, corresponding to a relative improvement of 4.5%. These results show that using multiple retrieval-oriented rewrites improves both retrieval coverage and ranking quality within the top five results.

At the same time, Hit@1 decreases from 0.616 to 0.586. This indicates that the proposed method does not always improve the first retrieved document. Instead, its main benefit is that it retrieves relevant documents more often within the top-five result set. This trade-off is important for retrieval-augmented question answering, where several retrieved documents are usually passed to the downstream answer generation step. In this setting, improving the coverage and usefulness of the top-five context set can be more valuable than optimizing only the first retrieved document.

5.3 Ablation study on internal dataset

The main comparison in the previous section evaluates the overall effect of the final rewrite-only equal-weight method. To better understand which components contribute to this improvement, we conduct a set of ablation experiments on the

Weight set	w_1	w_2	w_3	Hit@1	Hit@5	Recall@5	NDCG@5
w1	1.0	0.8	0.6	**0.586**	**0.929**	0.804	0.689
w2	1.0	0.7	0.5	**0.586**	**0.929**	0.804	0.689
w3	1.0	0.6	0.4	0.576	**0.929**	0.806	0.689
w4	1.0	0.5	0.3	**0.586**	**0.929**	0.804	**0.693**
w5	0.9	0.6	0.3	0.576	**0.929**	0.804	0.688
w6	0.8	0.6	0.4	**0.586**	**0.929**	0.804	0.689
w7	1.2	0.8	0.5	0.576	**0.929**	0.804	0.686
w8	1.2	0.7	0.3	**0.586**	**0.929**	0.804	**0.693**
w9	1.0	0.4	0.2	**0.586**	0.919	0.786	0.686
w10	1.0	0.3	0.1	**0.586**	0.919	0.781	0.683
GQR-only / w11	1.0	0.0	0.0	0.576	0.889	0.766	0.667
w12	**1.0**	**1.0**	**1.0**	**0.586**	**0.929**	**0.808**	0.692

Table 8: Rewrite-weight ablation on the *All_Questions* dataset.

All_Questions dataset. Each ablation changes one part of the retrieval pipeline while keeping the remaining components fixed as much as possible.

The ablation study focuses on three aspects of the proposed method. First, we evaluate different rewrite-weight configurations to examine how sensitive reciprocal rank fusion is to the weights assigned to the generated rewrites. Second, we compare different retrieval backends to study the contribution of lexical retrieval, dense retrieval, and hybrid retrieval. Third, we analyze the effect of the number of rewrites and whether the original query should be included in the final fused ranking.

Together, these experiments are designed to identify whether the improvement comes mainly from the use of multiple rewrites, the hybrid retrieval backend, the fusion weights, or the exclusion of the original query from the final fusion step.

5.3.1 Rewrite-weight exploration

This experiment evaluates how sensitive the multi-query retrieval pipeline is to the weights assigned to the generated rewrites. All configurations use the same labeled dataset, document corpus, chunking procedure, hybrid retrieval backend, and evaluation metrics. The only variable changed in this experiment is the set of weights used in reciprocal rank fusion.

The weights w_1, w_2, and w_3 correspond to the first, second, and third generated rewrite respectively. The tested configurations include gradually decreasing weights, manually designed alternatives, a GQR-only setting, and an equal-weight setting. The GQR-only setting uses only the first rewrite and acts as a control condition for testing whether the second and third rewrites provide additional retrieval evidence.

Table 8 reports the results of this experiment. The GQR-only setting obtains a Recall@5 of 0.766 and an NDCG@5 of 0.667, which is lower than the multi-rewrite configurations. This shows that using only the first rewrite is not sufficient to obtain the strongest retrieval performance.

The equal-weight configuration, which assigns $w_1 = 1.0$, $w_2 = 1.0$, and $w_3 = 1.0$, achieves the highest Recall@5 score of 0.808 and reaches the highest Hit@5 score of 0.929. This indicates that the three generated rewrites provide complementary retrieval evidence and do not need to be manually prioritized through different weights. Since the downstream QA system can use multiple retrieved documents as context, this equal-weight setting is selected as the final rewrite-weight configuration.

5.3.2 Retrieval backend ablation

This experiment evaluates whether the improvement of the proposed method depends on the use of hybrid retrieval. The query rewriting and fusion procedure are kept fixed, while the retrieval backend is changed. The tested backends are BM25-only retrieval, dense-only retrieval, and hybrid retrieval with different values of α. The parameter α controls the contribution of BM25 in the hybrid retrieval score, where larger values assign more weight to lexical matching.

The purpose of this ablation is to isolate the effect of the retrieval backend. If BM25-only or dense-only retrieval performs similarly to hybrid retrieval, then the improvement would mainly come from the query rewriting and fusion

Backend setting	Hit@1	Hit@5	Recall@5	NDCG@5
BM25-only	0.444	0.838	0.708	0.584
Dense-only	0.505	0.818	0.724	0.620
Hybrid, $\alpha = 0.25$	0.606	0.879	0.769	0.676
Hybrid, $\alpha = 0.50$	**0.616**	**0.929**	0.794	**0.695**
Hybrid, $\alpha = 0.60$	0.586	**0.929**	**0.804**	0.693
Hybrid, $\alpha = 0.75$	0.535	0.899	0.780	0.660

Table 9: Retrieval backend ablation on the *All_Questions* dataset. The parameter α controls the BM25 weight in the hybrid retrieval score.

Configuration	Hit@1	Hit@5	Recall@5	NDCG@5
Original query only	0.616	0.869	0.746	0.662
Rewrite 1 only	0.576	0.889	0.766	0.667
Rewrite 1 + 2, equal weights	0.545	0.919	0.792	0.673
Rewrite 1 + 2 + 3, equal weights	0.586	**0.929**	**0.808**	0.692
Original + rewrite 1	**0.636**	0.869	0.747	0.674
Original + rewrite 1 + 2	0.596	0.899	0.777	0.689
Original + rewrite 1 + 2 + 3, equal weights	0.616	0.919	0.795	**0.697**
Original + rewrite 1 + 2 + 3, w8	0.626	0.899	0.775	0.689

Table 10: Ablation results for different combinations of the original query and generated rewrites on the *All_Questions* dataset.

procedure. If hybrid retrieval performs better, this means that the retrieval backend itself also contributes to final performance by combining lexical and semantic evidence.

Table 9 reports the backend ablation results. BM25-only and dense-only retrieval are both weaker than hybrid retrieval. BM25-only obtains a Recall@5 of 0.708 and an NDCG@5 of 0.584, while dense-only obtains a Recall@5 of 0.724 and an NDCG@5 of 0.620. This shows that neither lexical nor dense retrieval alone is sufficient for the best performance.

The hybrid settings perform better because they combine exact lexical matching with semantic similarity. The hybrid setting with $\alpha = 0.50$ obtains the highest NDCG@5 score of 0.695, while $\alpha = 0.60$ obtains the highest Recall@5 score of 0.804. This means that both sparse and dense retrieval signals are useful, and that the best balance depends on whether ranking quality or top-five coverage is prioritized.

5.3.3 Rewrite count and original-query ablation

This experiment evaluates how the number of query formulations affects retrieval performance. The retrieval backend and evaluation procedure are kept fixed, while the set of queries included in the fused ranking is changed. The tested configurations include the original query alone, one rewrite, two rewrites, three rewrites, and combinations that add the original query back into the fusion step.

The purpose of this ablation is to separate two questions. The first question is whether adding more generated rewrites improves retrieval performance. The second question is whether the original query should be retained as part of the final fused ranking. This is important because the original query preserves the user's initial wording, while the generated rewrites may introduce more retrieval-oriented formulations.

Table 10 reports the results of this experiment. The original query alone obtains a Recall@5 of 0.746 and an NDCG@5 of 0.662. Using only the first rewrite improves Recall@5 to 0.766 and NDCG@5 to 0.667, but reduces Hit@1 from 0.616 to 0.576. This means that the first rewrite can improve top-five coverage, but it may also move the most relevant document away from the first rank.

Adding more rewrites further improves top-five retrieval quality. The configuration with three equally weighted rewrites obtains the highest Recall@5 score of 0.808 and the highest Hit@5 score of 0.929. This supports the idea that the three rewrites provide complementary retrieval perspectives. The first rewrite preserves the original intent, while the second and third rewrites provide additional lexical, core-content, or document-like formulations.

Method	Average judge score	Normalized judge score
Baseline hybrid retrieval	5.4	0.537
Our method	5.8	0.576
Absolute change	**+0.4**	**+0.039**
Relative change	**+7.3%**	**+7.3%**

Table 11: Prompt-based judge comparison on the *refined_chat* dataset.

Including the original query can improve ranking stability. The configuration that combines the original query with all three rewrites using equal weights obtains the highest NDCG@5 score of 0.697 and preserves the baseline Hit@1 score of 0.616. However, its Recall@5 and Hit@5 are lower than the rewrite-only equal-weight configuration. Since the goal of the final method is to improve the top-five context set for downstream QA, the rewrite-only equal-weight configuration is selected as the final proposed method.

5.4 Comparison on additional internal dataset

This experiment evaluates the retrieval pipeline on production-style query logs. Unlike *All_Questions*, the *refined_chat* dataset does not contain manually annotated ground-truth document identifiers. Therefore, standard retrieval metrics such as Hit@5, Recall@5, and NDCG@5 cannot be computed directly. Instead, this experiment uses the prompt-based retrieval quality judge described in Appendix B to obtain an approximate quality estimate.

The purpose of this experiment is to test whether the proposed query rewriting method also improves retrieval quality on realistic unlabeled queries. The baseline and the proposed method are evaluated on the same query log data and are judged using the same evaluation prompt. The baseline uses the original query with hybrid retrieval, while the proposed method uses LLM-generated rewrites and reciprocal rank fusion. For each query, the judge receives the original user question and the retrieved documents, and assigns an overall retrieval quality score from 0.0 to 10.0.

Table 11 reports the average judge scores. The baseline obtains an average overall judge score of 5.4, while the proposed method obtains a higher score of 5.8. This corresponds to an absolute improvement of 0.4 points on the 0–10 scale, or a relative improvement of 7.3%. The normalized judge score also increases from 0.537 to 0.576.

These results suggest that the proposed query rewriting method also improves retrieval quality on unlabeled production-style queries. Since the judge evaluates whether the retrieved documents are useful for answering the original user question, the higher score indicates that the rewritten queries tend to retrieve documents that are more semantically relevant and more useful as answer evidence.

However, this experiment should be interpreted more cautiously than the labeled evaluation on *All_Questions*. The scores are produced by an LLM-based judge rather than by human relevance annotations. They are therefore used as a complementary quality estimate, not as a replacement for ground-truth retrieval metrics.

5.5 Comparison on open-source dataset

This experiment evaluates whether the final rewrite-only equal-weight method generalizes beyond the internal KPN enterprise QA setting. While *All_Questions* contains Dutch support-style questions and document-level knowledge base identifiers, FIQA is an English financial question-answering retrieval dataset. This makes it useful for testing whether the proposed multi-query rewriting approach also improves retrieval in a different language, domain, and document style.

The experiment compares the same two retrieval configurations used in the main internal evaluation. The baseline uses the original financial question as a single retrieval query with the hybrid retrieval backend. The proposed method uses the three LLM-generated rewrites as retrieval inputs, retrieves results independently for each rewrite, and combines the ranked lists using equal-weight reciprocal rank fusion. The original query is not included in the final fusion step.

The evaluation is conducted using the relevance judgments provided in FIQA. For each query, the retrieved top-five corpus documents are compared with the relevant corpus identifiers in the qrels table. In addition to Recall@5, NDCG@5, and Hit@5, this experiment also reports Precision@5 and MRR@5, which are commonly used in benchmark-style retrieval evaluation.

Method	Precision@5	Recall@5	NDCG@5	MRR@5	Hit@5
Baseline hybrid retrieval	0.146	0.339	0.320	0.401	0.527
Our method	0.172	0.398	0.365	0.444	0.596
Absolute change	**+0.026**	**+0.059**	**+0.045**	**+0.043**	**+0.069**
Relative change	**+17.9%**	**+17.4%**	**+14.0%**	**+10.6%**	**+13.2%**

Table 12: Comparison between the baseline hybrid retrieval system and our method on FIQA.

Table 12 reports the FIQA results. The proposed method improves retrieval performance across all reported metrics. Precision@5 increases from 0.146 to 0.172, corresponding to a relative improvement of 17.9%. Recall@5 increases from 0.339 to 0.398, corresponding to a relative improvement of 17.4%. NDCG@5 improves from 0.320 to 0.365, corresponding to a relative improvement of 14.0%. This shows that the generated rewrites help the retriever find more relevant financial passages and rank them more effectively within the top five results.

To contextualize these results, previous BEIR experiments on *FiQA-2018* report several retrieval baselines using nDCG@10 rather than the @5 metrics used in this thesis. Thakur et al. [28] report an nDCG@10 score of 0.236 for BM25 on FiQA-2018, while stronger retrieval and reranking methods such as ColBERT and BM25 with cross-encoder reranking reach 0.317 and 0.347 respectively. These numbers are not directly comparable to the results in this thesis because the cut-off, preprocessing, and retrieval setup differ. However, they provide useful context: FIQA is a challenging retrieval benchmark, and the improvement from 0.320 to 0.365 in NDCG@5 suggests that the proposed query-rewriting setup provides a meaningful gain over the thesis baseline on this dataset.

The proposed method also improves MRR@5 from 0.401 to 0.444, corresponding to a relative improvement of 10.6%. Hit@5 increases from 0.527 to 0.596, corresponding to a relative improvement of 13.2%. This means that a relevant financial passage is retrieved within the top five results for more queries.

Overall, the FIQA experiment provides external evidence for the usefulness of the proposed method. The improvement on FIQA means that the rewrite-only equal-weight strategy is not limited to the Dutch KPN knowledge base. Instead, treating the generated rewrites as complementary retrieval views can also improve retrieval performance on an open-source English financial QA benchmark.

6 Discussion

6.1 Reflection on results

The results show that LLM-based adaptive query rewriting can improve retrieval performance, but the improvement mainly comes from using multiple complementary rewrites rather than from rewriting the query only once. The GQR-only setting provides only a small improvement over the baseline in top-five retrieval metrics. This means that a single information-preserving rewrite can help make the original question slightly more retrieval-friendly, but it does not fully address the vocabulary mismatch between user questions and enterprise documentation.

The stronger improvement is obtained when multiple rewritten queries are combined through weighted reciprocal rank fusion. This indicates that different rewrite strategies capture different aspects of the same information need. The first rewrite preserves the full user intent, while later rewrites can emphasize keywords, core actions, entities, or document-like formulations. These alternative formulations help retrieve relevant documents that may not be matched by the original query or by a single general rewrite. In an enterprise knowledge base, where documents often use procedural titles, internal terminology, product names, or service-specific wording, this additional retrieval diversity is especially useful.

At the same time, the results also show a trade-off between first-rank accuracy and broader top-five retrieval quality. On the internal evaluation dataset, the proposed method improves Hit@5, Recall@5, and NDCG@5, but Hit@1 remains lower than the baseline. This means that the method retrieves relevant documents more often within the top five results, but it does not always place a relevant document at the first position. This trade-off is important for interpreting the results. If a system only displays or uses the top-ranked document, the lower Hit@1 would be a concern. However, in a retrieval-augmented QA setting, several retrieved documents are typically passed to the answer generation step. In that context, improving the quality and coverage of the top-five context set can be more valuable than optimizing only the first retrieved document.

The ablation results show that the equal-weight configuration is a strong and simple final setting. Among the rewrite-only configurations, it achieves the highest Recall@5 and reaches the highest Hit@5. This means that the three generated rewrites provide complementary retrieval evidence and do not need to be manually prioritized through different weights. Although including the original query with the rewrites gives a slightly higher NDCG@5, the rewrite-only equal-weight configuration gives the strongest top-five coverage. Since the downstream QA system can use multiple retrieved documents as context, improving top-five coverage is especially important for this setting.

The experiment on the *refined_chat* dataset provides additional evidence from production-style query logs. Since this dataset does not contain manually annotated relevance labels, it cannot be evaluated with standard retrieval metrics. The prompt-based retrieval quality judge instead provides an approximate quality estimate. The higher average judge score for the proposed method suggests that query rewriting also improves retrieval quality on realistic unlabelled queries. However, this result should be interpreted more cautiously than the labeled evaluation, because the scores are produced by an LLM-based judge rather than by human relevance annotations. The judge-based evaluation is therefore useful as a complementary signal, but not as a replacement for manually labelled retrieval evaluation.

The FIQA experiment further supports this interpretation. The same rewrite-only equal-weight method improves all reported metrics on an English financial QA dataset, including Recall@5, NDCG@5, MRR@5, and Hit@5. This means that the benefit of multi-query rewriting is not limited to the Dutch KPN knowledge base. Instead, the method appears to transfer to another domain where queries and documents differ in language, style, and terminology.

Overall, the findings suggest that LLM-based query rewriting is most useful when it is integrated into a multi-query retrieval pipeline. The method improves retrieval performance by creating several retrieval-oriented queries for the same user question, and weighted fusion helps control the influence of each rewrite. For enterprise QA systems, this is practically relevant because user questions are often short, informal, or different from the terminology used in internal documentation. The proposed method does not remove the need for careful evaluation and tuning, but it provides a practical way to improve the quality of the retrieved context used for downstream answer generation.

6.2 Limitations

This study has several limitations. The labeled internal evaluation dataset is relatively small. The *All_Questions* dataset contains 100 questions, which is useful for controlled evaluation but may not cover the full variety of queries that occur in a production enterprise QA system. A larger manually annotated evaluation set would make the results more reliable and would allow more detailed analysis across query types, products, document categories, and user intents.

The final rewrite fusion strategy is selected based on the available evaluation results. The equal-weight rewrite-only configuration is simple and does not require manually tuning different weights for each rewrite. However, the choice is still evaluated on the same internal dataset used for the ablation study. Therefore, the selected configuration should be interpreted as an ablation-based choice rather than as a fully independent hyperparameter optimization result.

The evaluation is performed mainly at document level, while retrieval itself is performed at chunk level. This is appropriate for matching retrieved chunks back to knowledge base articles, but it may hide differences in passage-level quality. A retrieved document can be counted as relevant even if the specific retrieved chunk is not the most useful part of that document.

The *refined_chat* dataset does not contain human relevance labels. The prompt-based retrieval quality judge provides a useful complementary quality estimate for unlabeled production-style query logs, but it is not equivalent to manual annotation. Since the judge is also based on an LLM, its scores may be affected by model bias, prompt sensitivity, and inconsistency across runs.

This study does not include a controlled latency and cost analysis of the full proposed pipeline. Query rewriting requires an additional LLM call, and multi-query retrieval requires several retrieval operations before rank fusion. These steps can increase response time and computational cost. Although some datasets contain latency-related fields from previous systems, they are not directly comparable to the proposed method under the same runtime conditions. In addition, the `llm_raw_response` field contains token usage and estimated cost information, but does not contain direct runtime measurements for the LLM call. Therefore, LLM generation latency cannot be measured directly from the available data.

The current method uses a fixed prompt and a manually designed set of rewrite strategies. Although the results show that these strategies are useful, the prompt itself may not be optimal for every domain or query type. The method also evaluates retrieval quality rather than the complete end-to-end answer quality of the QA system. Better retrieval

performance is expected to support answer generation by providing a stronger evidence set, but this relationship is not directly measured in the current experiments.

6.3 Future work

Future work should validate the selected rewrite-only equal-weight configuration on a larger held-out test set or through cross-validation. This would provide a more robust estimate of how well the selected strategy generalizes to unseen enterprise queries.

Another direction is to evaluate both document-level and passage-level relevance. Since retrieval is performed over chunks but evaluated at document level, passage-level evaluation would make it possible to assess whether the retrieved chunk itself contains the most useful evidence for answering the question.

Future work should also compare the prompt-based judge scores with human judgments. This would help determine whether the judge is reliable for enterprise retrieval evaluation. A lower-scale judge, such as a 1–5 Likert-style scale, could also be tested to make the judging task simpler and reduce the appearance of false numerical precision.

A controlled latency and cost analysis is also needed before practical deployment. Future experiments should record end-to-end latency, LLM generation time, retrieval time, fusion time, token usage, and estimated cost under the same runtime conditions.

The query rewriting method itself could also be improved. Future work could investigate automatic prompt optimization, query-type-specific rewriting policies, or adaptive selection of the number of rewrites. Another useful direction would be to decide dynamically whether rewriting is necessary at all, since some original queries may already be short, precise, and retrieval-friendly.

Finally, future work should evaluate whether the improved top-five retrieval context leads to more accurate and helpful generated answers. This would connect retrieval improvements more directly to the final user experience in an enterprise QA system.

7 Conclusion

This thesis investigated whether LLM-based query rewriting can improve document retrieval performance in an enterprise QA system. The proposed method uses an adaptive prompt-based rewriting module to generate one to three retrieval-oriented rewrites for each user question. Each rewrite is retrieved independently using a hybrid retrieval pipeline that combines BM25 and dense retrieval, and the resulting ranked lists are merged using weighted reciprocal rank fusion.

The first research question asked how multi-query rewriting affects performance compared with the baseline.The results show that the proposed method improves retrieval performance on the labeled *All_Questions* dataset. Hit@5 increases from 0.869 to 0.929, Recall@5 increases from 0.746 to 0.808, and NDCG@5 increases from 0.662 to 0.692. The comparison with the GQR-only setting shows that a single general rewrite provides only a small improvement, while the full multi-query method gives a larger gain. This indicates that the main benefit comes from combining multiple complementary rewrites.

The second research question asked how different rewrite-weight configurations affect the final fused ranking. The ablation results show that using all three generated rewrites with equal weights provides the strongest top-five coverage. Manually weighted configurations perform similarly, but the equal-weight setting is simpler and avoids manually prioritizing one rewrite type. This means that the rewrites act as complementary retrieval views and can be combined effectively with a simple fusion strategy.

The third research question asked what the practical trade-offs are when applying LLM-based query rewriting in an enterprise retrieval pipeline. The results suggest a trade-off between retrieval quality and system complexity. Query rewriting improves retrieval coverage and ranking quality, but it also requires additional LLM calls and a more complex retrieval pipeline. For unlabeled production-style logs, the prompt-based judge means that the proposed method improves average retrieval quality, but this evaluation is less reliable than evaluation with human relevance labels. Therefore, query rewriting appears promising for enterprise QA, but its deployment should consider latency, cost, and monitoring of retrieval quality.

In conclusion, this thesis shows that LLM-based adaptive query rewriting can improve document retrieval in an enterprise QA setting. The strongest improvements are obtained when multiple rewritten queries are combined through

weighted reciprocal rank fusion. The method improves performance on both internal KPN evaluation datasets and open-source FIQA dataset, suggesting that the approach can generalize beyond one specific corpus. These findings support the use of query rewriting as a practical component for improving retrieval-augmented enterprise QA systems.

Acknowledgments

To start with, I would like to thank my supervisors, Suzan Verberne and Yumeng Wang, for their guidance throughout this thesis. Their feedback, research-related insights, and suggestions helped shape the direction and presentation of this work. I would also like to thank my thesis examiner, Peter van der Putten. His detailed review and critical feedback helped me make the thesis more rigorous, precise, and better positioned within the existing literature.

In the applied part of my thesis, I would like to thank KPN for their technical support. I am especially grateful to Gianluigi Bardelloni for his support during the internship and for providing the opportunity to work on this project. KPN provided access to their data and shared their domain knowledge, which enabled further improvement of their workflows.

Lastly, I would like to thank the following people for reading my thesis and providing overall support throughout the internship: Bart Westhoff, Federico Signorelli, Maksim Starostenko and Yilan Zhang.

References

[1] Akari Asai et al. "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection". In: *International Conference on Learning Representations*. 2024.

[2] Sebastian Bruch, Siyu Gai, and Amir Ingber. "An analysis of fusion functions for hybrid retrieval". In: *ACM Transactions on Information Systems* 42.1 (2023), pp. 1–35.

[3] Chi-Min Chan et al. "Rq-rag: Learning to refine queries for retrieval augmented generation". In: *arXiv preprint arXiv:2404.00610* (2024).

[4] Jiawei Chen et al. "Benchmarking large language models in retrieval-augmented generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 16. 2024, pp. 17754–17762.

[5] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: Association for Computing Machinery, 2009, pp. 758–759. ISBN: 9781605584836. DOI: 10.1145/1571941.1572114. URL: https://doi.org/10.1145/1571941.1572114.

[6] Shahul Es et al. "Ragas: Automated evaluation of retrieval augmented generation". In: *Proceedings of the 18th conference of the european chapter of the association for computational linguistics: system demonstrations*. 2024, pp. 150–158.

[7] Luyu Gao et al. "Precise zero-shot dense retrieval without relevance labels". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2023, pp. 1762–1777.

[8] Kelvin Guu et al. "Retrieval augmented language model pre-training". In: *International conference on machine learning*. PMLR. 2020, pp. 3929–3938.

[9] Gautier Izacard and Edouard Grave. "Leveraging passage retrieval with generative models for open domain question answering". In: *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*. 2021, pp. 874–880.

[10] Gautier Izacard et al. "Atlas: Few-shot learning with retrieval augmented language models". In: *Journal of Machine Learning Research* 24.251 (2023), pp. 1–43.

[11] Rolf Jagerman et al. *Query Expansion by Prompting Large Language Models*. 2023. arXiv: 2305.03653 [cs.IR]. URL: https://arxiv.org/abs/2305.03653.

[12] Soyeong Jeong et al. "Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity". In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 2024, pp. 7036–7050.

[13] Zhengbao Jiang et al. "Active retrieval augmented generation". In: *Proceedings of the 2023 conference on empirical methods in natural language processing*. 2023, pp. 7969–7992.

[14] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs". In: *IEEE transactions on big data* 7.3 (2019), pp. 535–547.

[15] Vladimir 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)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. URL: https://aclanthology.org/2020.emnlp-main.550/.

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

[17] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in neural information processing systems* 33 (2020), pp. 9459–9474.

[18] Zhicong Li et al. *DMQR-RAG: Diverse Multi-Query Rewriting for RAG*. 2024. arXiv: 2411.13154 [cs.IR]. URL: https://arxiv.org/abs/2411.13154.

[19] Yang Liu et al. "G-eval: NLG evaluation using gpt-4 with better human alignment". In: *Proceedings of the 2023 conference on empirical methods in natural language processing*. 2023, pp. 2511–2522.

[20] Xinbei Ma et al. "Query Rewriting in Retrieval-Augmented Large Language Models". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 5303–5315. DOI: 10.18653/v1/2023.emnlp-main.322. URL: https://aclanthology.org/2023.emnlp-main.322/.

[21] Shengyu Mao et al. “RaFe: Ranking feedback improves query rewriting for RAG”. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. 2024, pp. 884–901.

[22] Nils Reimers and Iryna Gurevych. “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)*. 2019, pp. 3982–3992.

[23] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Vol. 4. Now Publishers Inc, 2009.

[24] Jon Saad-Falcon et al. “Ares: An automated evaluation framework for retrieval-augmented generation systems”. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 2024, pp. 338–354.

[25] Keshav Santhanam et al. “Colbertv2: Effective and efficient retrieval via lightweight late interaction”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2022, pp. 3715–3734.

[26] Harrisen Scells, Guido Zuccon, and Bevan Koopman. “You Can Teach an Old Dog New Tricks: Rank Fusion applied to Coordination Level Matching for Ranking in Systematic Reviews”. In: *Advances in Information Retrieval*. Ed. by Joemon M. Jose et al. Cham: Springer International Publishing, 2020, pp. 399–414. ISBN: 978-3-030-45439-5.

[27] Weijia Shi et al. “REPLUG: Retrieval-Augmented Black-Box Language Models”. In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2024, pp. 8364–8377.

[28] Nandan Thakur et al. “Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models”. In: *arXiv preprint arXiv:2104.08663* (2021).

[29] Liang Wang, Nan Yang, and Furu Wei. “Query2doc: Query expansion with large language models”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 9414–9423.

[30] Liang Wang et al. “Text embeddings by weakly-supervised contrastive pre-training”. In: *arXiv preprint arXiv:2212.03533* (2022).

[31] Yujing Wang et al. “Maferw: Query rewriting with multi-aspect feedbacks for retrieval-augmented large language models”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 24. 2025, pp. 25434–25442.

[32] Lee Xiong et al. “Approximate nearest neighbor negative contrastive learning for dense text retrieval”. In: *arXiv preprint arXiv:2007.00808* (2020).

[33] Shi-Qi Yan et al. “Corrective Retrieval Augmented Generation”. In: *arXiv preprint arXiv:2401.15884* (2024).

[34] Lianmin Zheng et al. “Judging llm-as-a-judge with mt-bench and chatbot arena”. In: *Advances in neural information processing systems* 36 (2023), pp. 46595–46623.

[35] Yutao Zhu et al. “Large language models for information retrieval: A survey”. In: *ACM Transactions on Information Systems* 44.1 (2025), pp. 1–54.

A Prompt for adaptive query rewriting

```
System
You are a query rewriting module for a document retrieval system.

Your task is NOT to answer the question.
Your task is to generate a small set of retrieval-oriented rewrites that preserve
the user's intent while improving retrieval diversity.

The goal is not to maximize the number of rewrites.
The goal is to produce 1 to 3 high-quality rewrites that are useful for parallel
retrieval.
Do not generate a rewrite unless it adds a genuinely different retrieval
perspective.

Available rewrite strategies:

GQR (General Query Rewrite)
Use for general natural-language questions and as the default first rewrite when
preserving the full meaning helps ranking. This rewrite should remain close to
the original question while making it clearer and more retrieval-friendly.

KWR (Keyword-focused Rewrite)
Use for short, keyword-like, entity-heavy, terminology-heavy, acronym-heavy,
code-heavy, product-specific, item-specific, or concept-specific queries. This
rewrite should emphasize important lexical terms that are likely to appear in
relevant documents.

CCE (Core Content Extraction)
Use for action-oriented or task-oriented queries, such as finding information,
explaining a concept, comparing items, troubleshooting a problem, configuring a
system, applying a procedure, or locating a rule or policy. This rewrite should
keep the main action and the main object of the information need.

PAR (Pseudo-answer Rewrite)
Use only as an auxiliary rewrite when a neutral document-like formulation may
help retrieval. This can resemble how a relevant document, FAQ, manual, policy
page, explanation, answer passage, or troubleshooting guide might describe the
topic.
PAR must be neutral and document-like. It must not answer the question or invent
facts.

User
Generate retrieval-friendly rewrites of the user's question for parallel
retrieval.

IMPORTANT:
All rewrites must preserve the SAME user intent.

Global rules:
Output language MUST match the language of the original user question.
Do NOT change the user's intent.
Do NOT answer the question.
Do NOT add facts that are not present or directly implied by the original query.

Preserve all important terms exactly as they appear whenever present in the
original query: numbers, names, product names, model numbers, acronyms, codes,
```

```
IDs, dates, symptoms, error messages, domain-specific terms, technical terms,
legal or policy terms, named entities, and quoted phrases.
Preserve negations, constraints, symptoms, and problem descriptions whenever
relevant.
Preserve key action verbs when the rewrite still contains that action.
Do NOT merge tokens or change formatting of entities.
For acronyms, abbreviations, or short domain codes written in lowercase, preserve
the original form, but add a common casing variant only when useful for retrieval [...]
```

B Prompt for the retrieval quality judge

```
System
You are a strict and consistent retrieval quality judge.

Your task is to evaluate the final retrieval result for an original user question
when no ground-truth answer is available.

You must judge only from the original question and the retrieved documents.

Judge based on:
1. Semantic relevance to the original question
2. Direct usefulness for answering the question
3. Ranking quality of the retrieved documents
4. Coverage of the information needed to answer the question
5. Whether the retrieved documents are sufficient to generate a grounded answer

Be strict:
- Do not give high scores only because of keyword overlap.
- Prefer documents that directly answer the original question.
- Penalize documents that are only vaguely related.
- Penalize retrieval results where useful documents appear only at lower ranks.
- If the retrieved documents do not contain enough evidence to answer the
  question, give a low answerability score.
- Do not use outside knowledge. Judge only based on the retrieved documents.

All numeric scores must be decimal numbers with exactly one digit after the
decimal point, for example 7.0 or 8.5.

Return exactly one TSV row with 7 columns.
Do not include a header.
Do not use markdown.
Do not add extra commentary.
Do not include tab characters inside explanations.

User
Evaluate the final retrieval quality for the following original question.

Original question:
{{original_question}}

Rewritten queries used to produce the final retrieval, if available:
rewrite_1: {{rewrite_1}}
rewrite_2: {{rewrite_2}}
rewrite_3: {{rewrite_3}}
```

```
Final retrieved documents:
{{retrieved_documents}}

Use the scoring rubrics below.

Overall final retrieval score, from 0.0 to 10.0:
[...]
```