MASTER'S THESIS 2026

When Does Graph RAG Pay Off?

A Systematic Comparison of LightRAG and Chunk-Based RAG Pipelines

Erik Lundberg

ISSN 1650-2884
LU-CS-EX: 2026-21

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE

Datavetenskap

LU-CS-EX: 2026-21

When Does Graph RAG Pay Off?

A Systematic Comparison of LightRAG and Chunk-Based RAG Pipelines

När lönar sig Graph RAG? En systematisk jämförelse av LightRAG och chunkbaserade RAG-pipelines

Erik Lundberg

When Does Graph RAG Pay Off?

A Systematic Comparison of LightRAG and Chunk-Based RAG Pipelines

Erik Lundberg
erik.lundberg32@gmail.com

June 10, 2026

Master's thesis work carried out at Backtick Technologies.

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
Birger Kleve, birger@backtick.se, Fredrik Olsson, fredrik@backtick.se

Examiner: Elin A. Topp, elin_a.topp@cs.lth.se

Abstract

Prior work on RAG component ablations has largely evaluated retrieval strategies in isolation or on open-domain benchmarks, leaving unanswered which combinations actually improve answer quality and when graph-based retrieval justifies its complexity over simpler chunk-based alternatives. We ablate nine pipelines, six chunk-based (Dense, Hybrid, Query Expansion, HyDE, Reranking, Combined) and three LightRAG variants that progressively layer dense search and reranking on graph retrieval, across two GraphRAG-Bench domains and a Swedish water treatment corpus from Svenskt Vatten (109 expert-validated questions), evaluated with LLM-judge scoring averaged over five runs. Across all three corpora, cross-encoder reranking is the single most impactful component; stacking further enhancements yields diminishing returns. LightRAG outperforms chunk-based pipelines on creative and relational tasks but underperforms on factoid questions. On the water treatment corpus, LightRAG+Mix+Rerank achieves the highest overall correctness (7.21) and leads on all three question categories, but the gain is modest and most of it comes from the reranker rather than graph traversal: the fully stacked Combined chunk-based pipeline trails by only 0.20 correctness points at $k=5$ while using roughly a quarter of the input tokens. These findings give practitioners concrete guidance: prioritize reranking, and use LightRAG only when its modest additional gain over a fully stacked chunk-based pipeline is worth the complexity.

Keywords: RAG, LightRAG, knowledge graph, retrieval, reranking

Acknowledgements

I would like to thank my supervisor Pierre Nugues at Lund University for his expertise, enthusiasm, and guidance throughout this project. I would also like to thank the team at Backtick Technologies for their support in the form of compute resources, technical skill, and valuable discussions that helped shape the direction of this work. Furthermore, I would like to thank Ulf Thysell at Svenskt Vatten, whose deep knowledge of the water treatment domain was invaluable when constructing the evaluation set for the water treatment corpus.

Contents

Chapter 1

Introduction

Retrieval-augmented generation (RAG) has become the dominant architecture for grounding large language models (LLMs) in external knowledge, enabling responses that are up-to-date, domain-specific, and verifiable without retraining the underlying model [Lewis et al., 2020]. A canonical RAG pipeline retrieves text chunks from a corpus by vector similarity and conditions generation on the retrieved context, and this basic recipe underlies most production deployments today. To improve retrieval quality, researchers and practitioners have introduced a growing catalogue of components: sparse lexical search fused with dense embeddings via reciprocal rank fusion [Cormack et al., 2009], LLM-based query rewriting [Raudaschl, 2025; Wang et al., 2023] and hypothetical-document generation [Gao et al., 2023], and cross-encoder reranking over candidate pools [Nogueira and Cho, 2019]. More recently, graph-based approaches such as GraphRAG [Edge et al., 2024] and LightRAG [Guo et al., 2025] replace or augment the flat chunk index with a knowledge graph extracted from the source corpus, aiming to surface information across document boundaries through entity and relationship traversal.

However, practitioners deploying RAG in production face open questions about which of these components actually matter and when their added complexity is worth the cost. First, existing ablations tend to vary one component at a time or evaluate entire RAG paradigms as monolithic systems, making it difficult to attribute performance differences to specific design choices or to understand how components interact when stacked. Second, graph-based RAG has been evaluated predominantly on purpose-built benchmarks, leaving open the question of whether its advantages transfer to the specialized industrial domains where practitioners most want to deploy it. For example, an engineer at a Swedish water utility asking *how does changing the upstream carbon source affect phosphorus removal downstream?* needs a system that can traverse causal chains across treatment stages, the kind of relational reasoning graph retrieval is theoretically well suited for. But whether benchmark-derived results predict

performance on such queries in a Swedish-language technical corpus is unclear.

To address these gaps, this thesis systematically ablates nine retrieval pipelines: six chunk-based variants that layer hybrid search, query expansion, HyDE, and cross-encoder reranking on a dense baseline, and three LightRAG variants that progressively add dense retrieval and reranking to graph retrieval. We evaluate all nine across two GraphRAG-Bench [Xiang et al., 2025] domains covering clinical guidelines and literary prose, and on a Swedish water treatment corpus from Svenskt Vatten whose 20 technical documents and 109 expert-validated questions represent the kind of specialized industrial material where RAG adoption is most motivated. All evaluations use LLM-judge scoring for internal consistency, and each pipeline is also profiled for input and output token usage, enabling a direct cost–quality comparison that spans both benchmark and real-world settings.

The contributions of this thesis are threefold:

- **Component ablation.** A systematic component-level comparison of nine RAG pipelines on a common evaluation protocol (summarised in Table 3.1 and detailed in Chapter 3), identifying cross-encoder reranking as the single most impactful retrieval enhancement and showing that further stacking yields diminishing returns. The reranker is also what makes graph-based retrieval competitive: without it, pure LightRAG underperforms a dense baseline.
- **Question-type specialization.** A mapping of the question types where each pipeline family excels: LightRAG leads on creative generation and relational multi-hop queries, chunk-based pipelines dominate factoid retrieval, and finer-grained patterns emerge between variants (for example, query expansion edges out LightRAG+Mix+Rerank on specific multi-hop questions in the water treatment domain).
- **Practitioner-relevant cost–quality finding.** On a specialized industrial corpus (a Swedish water treatment collection from Svenskt Vatten, evaluated against a new 109-question expert-validated test set, the first systematic RAG ablation on a Swedish specialized domain to our knowledge), the fully stacked chunk-based pipeline reaches within 0.20 correctness points of LightRAG+Mix+Rerank at roughly a quarter of the input token cost. The reranker, not graph traversal, is doing most of the heavy lifting, meaning teams deploying RAG on specialized domains can capture nearly all of the available quality without the graph indexing overhead.

1.1 Related Work

Several recent studies compare graph-based and chunk-based RAG. GraphRAG-Bench [Xiang et al., 2025] benchmarks six graph RAG systems against chunk-based baselines across four question types, finding that no single system dominates: graph methods excel on multi-hop and summarization queries while chunk-based RAG remains competitive on factoid lookups. Han et al. [2025] reach a similar conclusion under a unified evaluation protocol, reporting that embedding-based RAG outperforms GraphRAG by 13.4% on factoid benchmarks but

underperforms on query-based summarization. Zhou et al. [2025] implement 12 graph RAG methods in a shared testbed and confirm this task-dependent pattern. Notably, Zeng et al. [2025] show that LightRAG's original evaluation suffered from length and position bias; after correction, its claimed advantage over naive RAG largely disappears, motivating the use of more controlled evaluation frameworks. Although these studies establish that graph and chunk-based RAG have complementary strengths, they treat each paradigm largely as a monolithic system, leaving unclear which individual components within each paradigm drive the observed differences or how components behave when stacked. Our work decomposes both paradigms into their retrieval components and ablates them jointly on a single protocol.

At the component level, Li et al. [2025] systematically vary RAG parameters including chunk size, query expansion, and retrieval stride, providing guidance on which design choices matter most. Yu et al. [2024] ablate reranking within a unified RAG model and show it contributes substantial accuracy gains across nine benchmarks. However, these component-level studies vary one or two components at a time within purely chunk-based pipelines and do not extend the ablation to graph-based alternatives. We bridge these two lines of work by applying a common component-level ablation protocol across both chunk-based variants (Dense, Hybrid, QE, HyDE, Rerank, Combined) and three LightRAG variants.

RAG is increasingly applied to specialized domains. Xiong et al. [2024] benchmark 41 retriever-LLM combinations on medical QA, finding that combining diverse corpora with different retrievers yields the best results and that retrieval design choices matter more than LLM scale in domain-specific settings. Yet existing domain-specific evaluations are largely English-centric and do not contrast chunk-based and graph-based retrieval on the same specialized corpus. To our knowledge, this thesis is the first to evaluate the full chunk-versus-graph spectrum on a Swedish water treatment corpus with expert-validated questions.

1.2 Limitations

All pipelines use Claude Haiku 4.5 for both generation and LLM-judge scoring. GPT-4o-mini was considered but API rate limits made running the full evaluation volume infeasible. Because the GraphRAG-Bench baseline results use a different model for scoring, direct numerical comparison with those results is not meaningful; the benchmarks serve here as a controlled setting for comparing pipelines relative to each other rather than against external leaderboard numbers.

The evaluation covers three corpora: two GraphRAG-Bench domains and one Swedish water treatment collection. While these span distinct knowledge structures, three corpora is a narrow empirical base. Therefore findings, especially ones on when graph-based retrieval pays off, should be treated as indicative rather than definitive.

1.3 Use of AI

Generative AI tools, specifically Claude and Gemini, were utilized as collaborative assistants in both the technical and writing phases of this thesis.

- **Software Development:** AI was used to generate code, including scripts and implementations of different RAG architectures.
- **Thesis Documentation:** AI was used to structure and write the report, including LaTeX formatting.

All AI-generated outputs were manually reviewed, verified for technical accuracy, and edited by the author to ensure academic integrity.

Chapter 2

Theoretical Background

2.1 Large Language Models

Large Language Models (LLMs) are neural networks trained on large text corpora to predict the next token in a sequence. Built on the Transformer architecture [Vaswani et al., 2017], which uses self-attention mechanisms to capture dependencies across an input sequence in parallel, modern LLMs have scaled to billions of parameters and demonstrate general-purpose capabilities including instruction following, reasoning, and coherent text generation.

Two properties of LLMs are particularly relevant to RAG. First, the *context window*, the maximum number of tokens the model can process in a single forward pass, determines how much retrieved content can be included in the prompt. Modern models support context windows of tens to hundreds of thousands of tokens, but RAG system design still involves trade-offs between providing more context and diluting relevance. Second, LLMs encode knowledge only from their training data, meaning they cannot access information that is domain-specific, proprietary, or published after the training cutoff. This limitation motivates retrieval-augmented approaches that supply relevant documents at inference time.

2.2 Retrieval Augmented Generation

Retrieval-augmented generation (RAG) is an architectural pattern that addresses the core limitations of purely parametric language models by conditioning generation on documents retrieved at inference time [Lewis et al., 2020]. Rather than relying solely on knowledge encoded in model weights during training, a RAG system augments the LLM's input with

relevant passages fetched from an external corpus, grounding responses in up-to-date and verifiable sources. The canonical pipeline introduced by Lewis et al. [2020] consists of two components: a retriever, which selects relevant documents given a query, and a generator, which produces a response conditioned on both the query and the retrieved context. Figure 2.1 illustrates the standard flow.

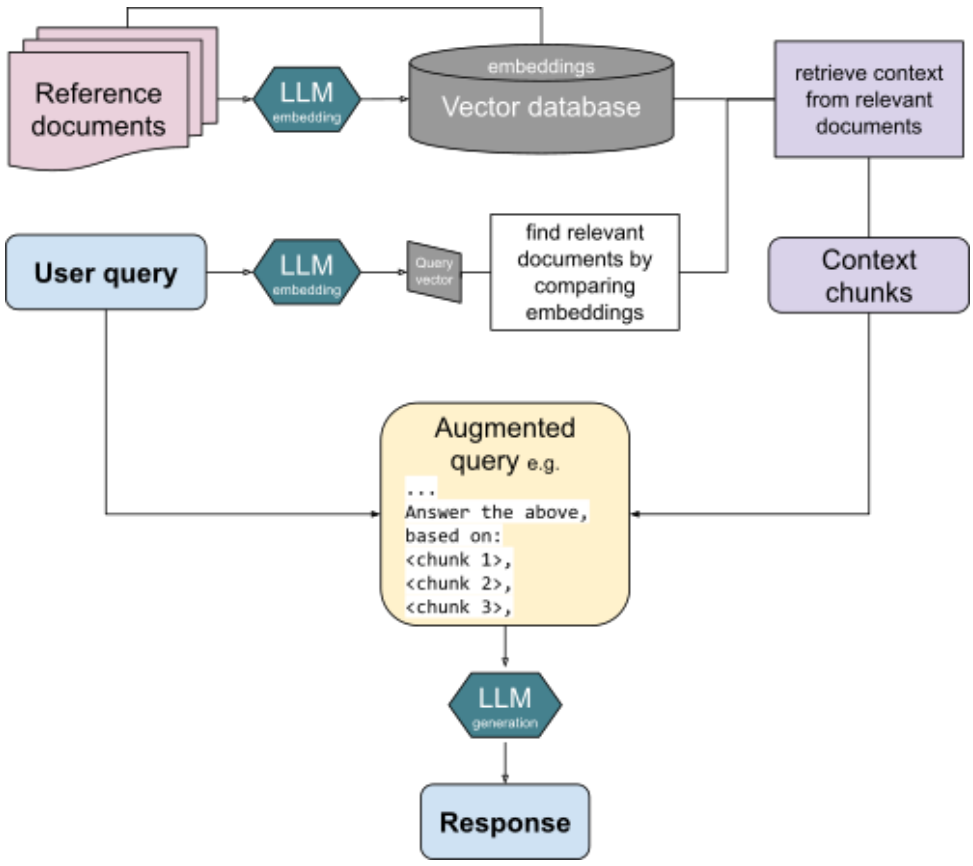


Figure 2.1: Schematic of a standard retrieval-augmented generation pipeline. Reference documents are embedded and stored in a vector database; at query time, the user query is embedded, the most similar context chunks are retrieved, and the augmented prompt (query plus retrieved chunks) is passed to a language model that produces the final response. *Source*: Wikipedia, "Retrieval-augmented generation".

Since this formalization, the retrieval component has evolved considerably, from sparse lexical methods such as Best Match 25 (BM25) [Robertson et al., 1994], through dense vector retrieval, to structured knowledge graphs that support multi-hop and relational reasoning [Edge et al., 2024]. The following subsections trace this progression.

2.2.1 BM25 and Lexical Retrieval

The foundation of classical keyword search is the *inverted index*, a data structure that maps each unique term in a corpus to the set of documents containing it. Given a query, the index enables sub-linear lookup of candidate documents, making it the backbone of virtually all large-scale text retrieval systems.

Early ranking functions such as TF-IDF scored documents by the product of term frequency and inverse document frequency, but suffered from unbounded term-frequency growth in long documents. BM25, introduced by Robertson et al. [1994], addresses these shortcomings with two key refinements: a saturating term-frequency component controlled by the parameter k_1, which dampens the impact of repeated terms, and a document-length normalization factor governed by b, which penalizes or rewards documents relative to the average corpus length. The score of a document D given a query Q is computed as

$$\mathrm{BM25}(D, Q) = \sum_{q \in Q} \mathrm{IDF}(q) \cdot \frac{f(q, D)\,(k_1 + 1)}{f(q, D) + k_1 \left(1 - b + b\,\frac{|D|}{\mathrm{avgdl}}\right)}, \tag{2.1}$$

where $f(q, D)$ is the frequency of term q in D, $|D|$ is the length of D in tokens, avgdl is the average document length in the corpus, and $\mathrm{IDF}(q)$ is the inverse document frequency of q. Together, these parameters yield a probabilistic ranking function that is both theoretically grounded and empirically robust.

Despite the rapid progress of neural retrieval methods, BM25 remains a strong baseline, particularly on queries containing rare or domain-specific terminology where exact lexical matching outperforms embedding similarity [Thakur et al., 2021]. This makes it a valuable component in hybrid pipelines that fuse sparse and dense retrieval signals.

2.2.2 Dense Retrieval

Where BM25 matches on exact terms, dense retrieval learns to map queries and passages into a shared embedding space so that semantically similar texts lie close together even when they share no words. Karpukhin et al. [2020] showed that a bi-encoder trained on question–passage pairs substantially outperforms BM25 on open-domain question answering, establishing dense passage retrieval as the dominant paradigm. Efficient sentence-level embeddings, such as those produced by Sentence-BERT [Reimers and Gurevych, 2019], further lowered the barrier by enabling fast approximate nearest-neighbor search over millions of passages.

A standard dense RAG pipeline operates in four stages: the source corpus is split into fixed-length chunks, each chunk is encoded into a vector by a bi-encoder (a model that encodes each text independently into a fixed vector; see Section 2.2.5), the vectors are indexed in a vector store, and at query time the top-k most similar chunks are retrieved and prepended to the LLM prompt as context. Because dense and sparse signals are complementary (embeddings capture paraphrases and synonyms while BM25 captures exact terminology), production systems often fuse both into a hybrid retrieval step, combining their ranked lists before generation.

2.2.3 Hybrid Retrieval and Reciprocal Rank Fusion

Combining the outputs of a lexical retriever (e.g. BM25) with those of a dense retriever requires a way to merge their ranked lists into a single ranking. The two retrievers produce incomparable scores: BM25 is an unbounded relevance score, while dense retrievers typically report cosine similarities bounded between -1 and 1, so naively summing scores is ill-defined. Reciprocal Rank Fusion (RRF) [Cormack et al., 2009] sidesteps this by scoring each document purely on its rank position within each list:

$$\mathrm{RRF}(d) = \sum_{r \in R} \frac{1}{k + \mathrm{rank}_r(d)},$$

where R is the set of ranked lists contributing to the fusion and k is a damping constant (typically 60) that softens the contribution of top ranks. RRF requires no per-retriever tuning or score calibration and is strictly rank-based, which makes it a robust default for combining heterogeneous retrievers whose underlying scores are not directly comparable.

2.2.4 Query Expansion and HyDE

The vocabulary a user employs to phrase a question is often different from the vocabulary used in the source documents, and both BM25 and dense retrieval can underperform when this gap is large. Query expansion addresses this by enriching the query before retrieval. Classical approaches relied on thesauri or pseudo-relevance feedback, but recent work uses LLMs to generate expansion content for the query, whether a single pseudo-document that stands in for the query (Query2doc [Wang et al., 2023]) or several alternative phrasings issued as parallel queries and merged via RRF (RAG-Fusion [Raudaschl, 2025]). Both strategies improve recall for questions whose most natural phrasing does not align with the corpus vocabulary. Figure 2.2 illustrates the multi-query rewrite variant.

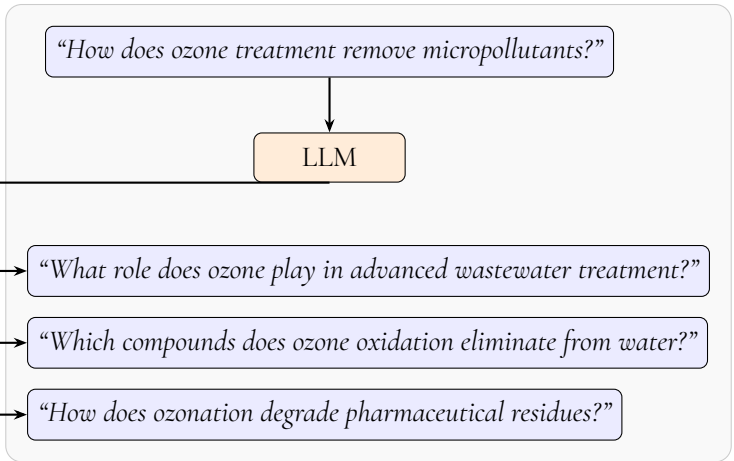


Figure 2.2: Query expansion. An LLM generates several semantically equivalent rewrites of the original query, broadening the lexical surface used for retrieval. Each rewrite is issued as a separate query, and their ranked results are fused (e.g., via RRF) into a single ranking.

HyDE (Hypothetical Document Embeddings) [Gao et al., 2023] takes a different route. Rather than generating alternative queries, it prompts an LLM to produce a hypothetical passage that would answer the question, then embeds and retrieves against this passage instead of the query. For example, given the question *"How does ozone treatment remove micropollutants?"*, an LLM might produce a synthetic answer along the lines of *"Ozone is a strong oxidant that breaks down pharmaceutical residues and other micropollutants by attacking double bonds in their molecular structure"*; this passage is then used as the retrieval query in place of the original question. By construction, the hypothetical passage uses vocabulary and structure closer to corpus documents than a short interrogative query, which typically improves dense-retrieval recall without requiring any labelled training data. The trade-off is one additional LLM call per query at retrieval time.

2.2.5 Cross-Encoder Reranking

Dense retrievers use a *bi-encoder* architecture: the query and each candidate passage are encoded independently into fixed vectors, so retrieval reduces to approximate nearest-neighbour search. This factorisation is what makes corpus-scale dense retrieval computationally feasible, but it also means the model never sees the query and passage together and can only model their interaction through vector similarity. *Cross-encoders* [Nogueira and Cho, 2019] remove this restriction by feeding the concatenated query–passage pair through a Transformer and predicting a relevance score conditioned on both. The resulting relevance estimates are substantially stronger than cosine similarity, at the cost of being too expensive to run over an entire corpus. Figure 2.3 contrasts the two architectures.

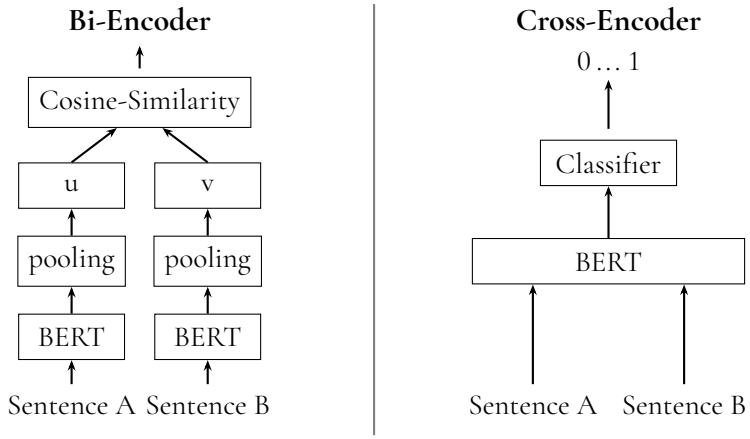


Figure 2.3: Bi-encoder vs. cross-encoder architectures. The bi-encoder (left) encodes each sentence independently into a fixed vector and compares them with cosine similarity, enabling fast nearest-neighbour search at corpus scale. The cross-encoder (right) jointly encodes the two sentences and produces a single relevance score, yielding more accurate judgements at the cost of higher per-pair latency. Adapted from the sentence-transformers documentation [Sentence-Transformers, 2024].

In practice, cross-encoders are therefore used as rerankers over a small candidate pool. A fast first-stage retriever (BM25, dense, or hybrid) returns a shortlist of typically 50–100 passages, and the cross-encoder rescores this shortlist to produce the final ordering. This two-stage retrieve-then-rerank pattern has become standard in modern RAG systems and is the basis of the rerank and combined pipelines evaluated in this thesis.

2.2.6 Knowledge Graphs

A knowledge graph (KG) is a graph-structured data model in which nodes represent entities and directed, labelled edges represent the relationships between them [Hogan et al., 2021]. Formally, a KG consists of a set of triples of the form *(subject, predicate, object)*, such as *(water pump, connected to, filter unit)*, enabling machines to represent and reason over real-world facts. Figure 2.4 shows a small illustrative example. Unlike relational databases, which enforce rigid tabular schemas, or NoSQL stores, which optimize for document or key-value access patterns, KGs are schema-flexible and designed for traversal, making them well-suited to domains where entity relationships are first-class information rather than foreign-key constraints. In the context of RAG, knowledge graphs serve as a structured retrieval layer: rather than searching over flat text chunks, the system can traverse entity neighborhoods and relationship paths to gather context that spans multiple source documents, motivating the graph-based approaches described below.

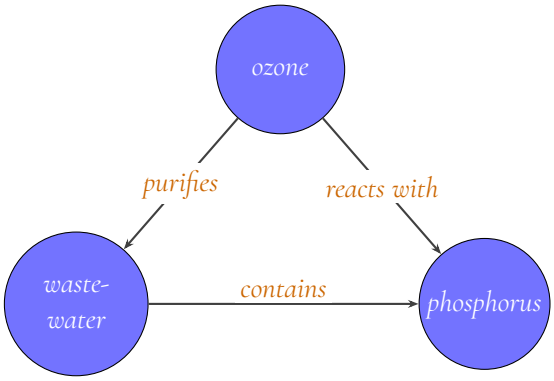


Figure 2.4: A small illustrative knowledge graph from the water treatment domain. Nodes represent entities; labelled edges represent named relationships between them.

2.2.7 GraphRAG and LightRAG

Traditional RAG pipelines treat each chunk independently, which limits their ability to answer queries that require reasoning across distant parts of a corpus. Graph-based RAG addresses this by constructing a knowledge graph from the source documents and retrieving over its structure rather than, or in addition to, a flat vector index.

GraphRAG, introduced by Edge et al. [2024], uses an LLM to extract entities and relationships from every chunk, assembling them into a corpus-wide knowledge graph. The graph is

then partitioned into hierarchical communities using the Leiden algorithm [Traag et al., 2019] (a community-detection method that partitions a graph into densely connected subgraphs by optimising modularity), and each community is pre-summarized by the LLM. At query time, these community summaries enable *global* queries that synthesize information spread across the entire corpus, a capability that chunk-level retrieval fundamentally lacks. The trade-off is a costly indexing phase: every chunk must pass through the LLM for extraction, and every community requires a summary generation step.

LightRAG [Guo et al., 2025] retains the LLM-based entity and relationship extraction of GraphRAG but replaces the global community summarisation with a lightweight, keyword-driven dual-level retrieval procedure. During indexing, each chunk is processed by an LLM that extracts entities (with names, types, and descriptions) and undirected relationships connecting them. Entity and relationship descriptions are embedded separately, and each graph element also carries *provenance* links back to the chunks from which it was extracted, so source text can be recovered by following these links rather than by retrieving chunks directly.

At query time, LightRAG first extracts two kinds of keywords from the user query: *low-level keywords* denoting specific entities or technical terms, and *high-level keywords* denoting abstract themes or concepts. These drive two parallel searches: low-level keywords are matched against entity embeddings to retrieve specific entities together with their direct neighbourhoods (connecting relationships), while high-level keywords are matched against relationship embeddings to retrieve broad thematic connections. Entities surfaced only through the global path are then expanded with their own neighbourhoods, bridging the two paths, and the combined entities and relationships form the graph context supplied to the generator. Because retrieval is driven by embeddings over compact entity and relationship descriptions rather than over raw chunks, LightRAG can surface related material that a chunk-level vector search would miss, without the expensive pre-computed community summaries that GraphRAG requires. LightRAG also supports incremental indexing, so new documents can be incorporated without rebuilding the graph.

2.3 Evaluation

Evaluating RAG systems requires measuring both the quality of retrieved context and the correctness of generated answers. Open-ended generation makes this challenging: determining whether an answer is "correct" is itself non-trivial, and human evaluation is expensive to scale. This has motivated automated evaluation frameworks tailored to RAG, described below.

2.3.1 LLM as a Judge

Human evaluation of open-ended text is expensive and difficult to scale. Zheng et al. [2023] showed that a strong LLM can serve as a proxy judge, achieving over 80% agreement with human annotators, comparable to inter-annotator agreement itself. In practice, the judge receives a question, one or more candidate answers, and optionally a reference answer, then

returns a score or pairwise preference. This approach makes it feasible to evaluate hundreds of generated answers at low cost, though it inherits the biases of the judge model, such as a tendency to favor longer or more verbose responses [Zeng et al., 2025].

2.3.2 RAGAS

RAGAS (retrieval-augmented generation assessment) [Es et al., 2024] is an evaluation framework designed specifically for RAG pipelines. It decomposes quality into component-level metrics computed via LLM calls, separately assessing retrieval quality (whether the right context was fetched) and generation quality (whether the answer is correct, relevant, and grounded in the retrieved context). By evaluating these stages independently, RAGAS helps pinpoint whether errors originate from poor retrieval or poor generation. RAGAS also provides a synthetic test-set generation pipeline that produces question–answer (QA) pairs from source documents, which we use to generate evaluation data for the water treatment corpus (Section 3.1.2).

2.3.3 GraphRAG-Bench

GraphRAG-Bench [Xiang et al., 2025] is a benchmark designed to evaluate graph-based RAG systems. Unlike end-to-end benchmarks that only score final answers, it evaluates retrieval and generation as separate stages, making it possible to diagnose whether failures originate from poor context selection or poor answer synthesis. The benchmark organizes questions into four types of increasing difficulty: *fact retrieval* (single-hop lookup), *complex reasoning* (multi-hop inference across documents), *contextual summarization* (synthesis of fragmented information), and *creative generation* (extrapolation beyond the retrieved content). It includes multiple domain-specific datasets; the two used in this thesis are clinical oncology guidelines (NCCN), providing tightly structured hierarchical knowledge, and pre-20th-century novels from Project Gutenberg, providing loosely organised narrative knowledge. The benchmark also provides baselines for both chunk-based and graph-based systems including MS-GraphRAG, HippoRAG, LightRAG, and RAPTOR.

Chapter 3

Method

This study employs an ablation design. We instantiate nine retrieval pipelines, six chunk-based variants that progressively layer components (hybrid search, query expansion, HyDE, and cross-encoder reranking) on a dense baseline, and three LightRAG variants that progressively add dense retrieval and reranking to graph-based retrieval. All pipelines share the same generation model and prompt, and are evaluated on the same questions using the same LLM judge. Each pipeline is run five times and results are reported as mean ± standard deviation. The following sections describe the evaluation corpora (Section 3.1), the nine pipeline implementations (Section 3.2), and the evaluation metrics and procedures (Section 3.3).

3.1 Datasets

3.1.1 Benchmark Datasets

We use the two domain-specific datasets released with GraphRAG-Bench [Xiang et al., 2025], which were deliberately paired to contrast a tightly structured, hierarchical domain with a loosely organised narrative one. Together they stress-test retrieval over complementary knowledge structures: formal, explicit hierarchies on one side and implicit, context-dependent relationships on the other.

The *Medical* dataset is built from National Comprehensive Cancer Network (NCCN) clinical guidelines. NCCN documents encode standardised oncology treatment protocols, drug interaction hierarchies, and diagnostic criteria as dense conceptual networks (for example, symptom → diagnosis → treatment chains). They represent the kind of tightly structured domain expertise that graph-based retrieval is theoretically well suited to navigate.

The *Novel* dataset is constructed from pre-20th-century novels sourced from Project Gutenberg, restricted to lesser-known works to minimise the risk of pretraining contamination inflating scores through memorisation. Literary prose exhibits the opposite profile to medical text: relationships between characters, places, and events are implicit, non-linear, and rely on socio-historical context, testing whether graph-based retrieval can capture latent connections beyond those stated explicitly in the text.

For both corpora, GraphRAG-Bench generates questions by first extracting a reference knowledge graph from the text and then sampling logic chains of triples from this graph. Question complexity is controlled along two dimensions: *knowledge breadth* (the number of triples needed to answer) and *reasoning depth* (the number of inference hops between those triples). Questions are grouped into four categories of increasing complexity, detailed in Section 3.3.1 where the corresponding evaluation metrics are introduced. Each dataset contains approximately 2,000 question–answer pairs distributed across the four categories. For computational tractability across nine pipelines, we sample 300 questions per dataset, stratified proportionally.

3.1.2 Water Treatment Data

To evaluate retrieval on a real-world domain-specific corpus, we use water treatment documents published by *Svenskt Vatten*, Sweden's industry organization for water and wastewater utilities. The corpus comprises 20 PDF documents (1,063 pages, 1,328 chunks) written in Swedish with domain-specific terminology and regulatory references. Individual documents range from 28 to 75 pages (median 54). The documents span six thematic areas:

- **Wastewater treatment & pollutants** (5 docs, 323 pages): biological phosphorus removal, PFAS, micropollutants, ethanol as carbon source, natural treatment systems.
- **Drinking water treatment** (4 docs, 184 pages): chlorine dioxide analysis, water hardening, iron/manganese filtration, artificial groundwater recharge.
- **Pipe infrastructure** (4 docs, 166 pages): hydraulic analysis, concrete pipe condition assessment, damage cases, tree root intrusion.
- **Stormwater & urban drainage** (3 docs, 177 pages): local stormwater management, stormwater recycling, combined sewer overflows.
- **VA sector governance** (3 docs, 153 pages): internal control, regional cooperation, VAFORSK program evaluation.
- **Sludge management** (1 doc, 60 pages): sustainable sludge handling in northern Sweden.

Unlike the GraphRAG-Bench benchmarks, which provide standardized QA pairs for controlled comparison, the water treatment corpus represents proprietary, specialized documentation that motivates RAG adoption in practice. General-purpose LLMs are unlikely to have

encountered this material during pretraining. The domain is also well suited for graph-based retrieval: water treatment involves interconnected chains of physical, chemical, and biological steps where understanding one stage often requires knowledge of upstream and downstream processes, naturally giving rise to multi-hop queries.

The source PDFs mix running text with tables, figures, and headers. We use Azure Document Intelligence for layout-aware extraction, which outputs Markdown (a lightweight text formatting syntax) preserving heading levels, bullet lists, and table formatting. The Markdown is then split into chunks following the shared tokenization parameters described below. Since no standardized QA pairs exist for this corpus, we generate them synthetically using the RAGAS test-set generation pipeline [Es et al., 2024]. RAGAS constructs a knowledge graph from the chunks: named-entity and keyphrase extraction identify key concepts, and a relationship builder links nodes with overlapping entities. An LLM-based query synthesizer traverses this graph, generating questions that require the linked information to answer, along with reference answers grounded in the same context. Varying the number of hops produces both single-hop factual and multi-hop reasoning questions. This process yielded 161 candidate QA pairs. A domain expert at Svenskt Vatten manually reviewed each pair for factual correctness and question clarity, approving 109 and rejecting 52 (32.3% overall rejection rate). The approved questions break down as follows: 37 single-hop (31.5% rejected), 39 multi-hop abstract (26.4% rejected), and 33 multi-hop specific (38.9% rejected).

The LightRAG knowledge graph extracted from this corpus contains 8,343 entities and 7,440 relationships, with an average degree of 1.78. The most common entity types are Concept (1,736), Method (1,067), Data (792), Artifact (781), and Location (731), as shown in Figure 3.1. High-degree hub entities include VA-FORSK (degree 206), PFAS (132), and LOD (64), reflecting the corpus's emphasis on research programs and emerging pollutants.

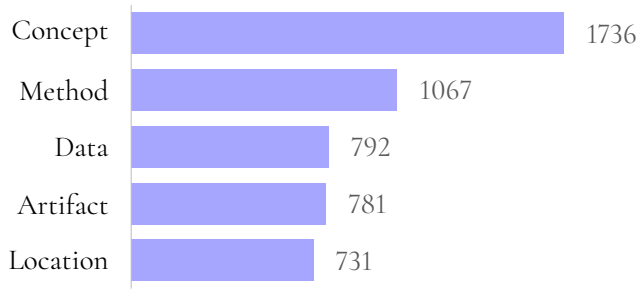


Figure 3.1: Distribution of the five most common entity types in the water treatment knowledge graph (8,343 entities total).

3.2 RAG Implementations

All pipelines, the chunk-based retrieval variants (Dense, Hybrid, QE, HyDE, Rerank, Combined) and the three LightRAG variants (Pure, Mix, Mix+Rerank), share a common architecture: a question passes through a retrieval stage that selects relevant passages from the corpus, then a language model generates an answer conditioned on the retrieved context. They also share a common set of infrastructure choices.

For all LLM calls (query rewriting, answer generation, and entity extraction) we use Claude Haiku 4.5, which provides a good balance between capability and throughput for the high volume of calls required across nine pipelines; other models such as GPT-4o-mini were considered but could not be used due to rate limits. Pipelines that include reranking use Cohere's `rerank-multilingual-v3.0` cross-encoder; the multilingual variant was chosen because the water treatment corpus is in Swedish. Text chunks are embedded using OpenAI's `text-embedding-3-small` model at 768 dimensions. Text is stored as flat chunks of approximately 1200 tokens with 100-token overlap at sentence boundaries, a common size in modern RAG implementations. The same chunk size is used for both chunk-based and LightRAG pipelines, ensuring that retrieval differences reflect strategy rather than chunk granularity. Chunk embeddings are stored alongside full-text search indexes in PostgreSQL (for the chunk-based pipelines) or in Neo4j (a native graph database) vector indexes (for LightRAG).

All pipelines retrieve $k = 5$ chunks for both the GraphRAG-Bench and the water treatment evaluations: chunk-based pipelines retrieve their top 5 chunks, and LightRAG variants use their default 5 source chunks via graph provenance. LightRAG context is dominated by graph traversal output (entity and relationship descriptions plus the 5 source chunks), substantially larger than what chunk-based pipelines produce at the same k. Reranking pipelines use a candidate pool of 50 passages before cutting to the final 5.

Table 3.1 summarises the nine pipelines.

Category	Pipeline	Description
Chunk-Based	Dense	Vector-only search over chunk embeddings
	Hybrid	BM25 + dense fused via RRF
	QE	Multi-query rewrites (original + 3 alternatives), vector search
	HyDE	Hypothetical-passage embedding + vector search
	Rerank	Dense candidate pool (50) + cross-encoder reranking
	Combined	HyDE + QE + hybrid + cross-encoder reranking
Graph-Based	LightRAG	Pure graph traversal, source chunks via provenance
	LightRAG+Mix	Graph traversal + parallel dense search, capped at k
	LightRAG+Mix+Rerank	Graph + dense + cross-encoder reranking

Table 3.1: Summary of the nine retrieval pipelines evaluated in this thesis.

Three shared mechanisms recur across multiple pipelines and are described here to avoid repetition.

Hybrid Search. Several pipelines combine dense vector retrieval with sparse lexical matching using Reciprocal Rank Fusion (Section 2.2.3). Two parallel retrieval paths execute: (1) a vector search over a cosine-similarity index and (2) a lexical search that matches the query against a PostgreSQL keyword index ranked by `ts_rank` (PostgreSQL's built-in text-search ranking function based on term frequency and proximity). The two ranked lists are

fused with RRF ($k = 60$). When the query text is empty, the lexical path returns nothing and the score reduces to pure vector similarity.

Cross-Encoder Reranking. Pipelines that include a reranking stage apply Cohere's cross-encoder (Section 2.2.5) to rescore an initial candidate pool of 50 passages, keeping the top-k.

Generation. All nine pipelines, both chunk-based and LightRAG variants, use the same generation prompt template:

> Answer the question using only the provided context. If the context does not contain enough information, say so.
>
> Context: {context}
>
> Question: {question}

For chunk-based pipelines, {context} is the concatenation of the retrieved text passages. For LightRAG pipelines, {context} is the structured graph context (entities, relationships, and source passages) serialised as text. No additional role instructions, formatting guidance, or chain-of-thought scaffolding is applied, ensuring that performance differences between pipelines reflect retrieval quality rather than prompt engineering.

3.2.1 Chunk-Based Pipelines

Table 3.2 shows which retrieval components each chunk-based pipeline activates.

Pipeline	Dense	BM25+RRF	QE	HyDE	Rerank
Dense	✓				
Hybrid	✓	✓			
QE	✓		✓		
HyDE	✓			✓	
Rerank	✓				✓
Combined	✓	✓	✓	✓	✓

Table 3.2: Retrieval components active in each chunk-based pipeline. Each pipeline builds on the dense baseline by adding components.

Dense Retrieval Baseline. Following the dense passage retrieval paradigm [Karpukhin et al., 2020], the question is embedded and compared against all chunk embeddings. The top-k chunks ranked by cosine similarity are retrieved and concatenated as context. No lexical matching, query augmentation, or reranking is applied. This pipeline establishes the performance floor for embedding-based retrieval.

Hybrid Retrieval. The question is used both as the embedding input for vector search and as the text input for lexical search. The RRF-fused results combine semantic similarity with exact keyword matching, surfacing passages that share terminology with the question even when their embeddings are not the closest vectors. The top-k results are passed as context. Hybrid search reflects a common production-grade retrieval setup and serves as the practical baseline against which the other enhancements are measured.

Query Expansion (QE). Following the query-expansion approach (Section 2.2.4), a language model generates three alternative phrasings of the original question, each approaching the topic from a different angle with varied vocabulary. All four queries (original plus alternatives) are independently embedded and used for vector-only search. Results are deduplicated by chunk identifier and concatenated as context.

Hypothetical Document Embeddings (HyDE). Following the HyDE approach (Section 2.2.4), a language model generates a hypothetical passage that would answer the question. Both the hypothetical passage and the original question are embedded and used for vector-only search, and the combined results are deduplicated and concatenated as context.

Reranking. Vector-only search retrieves a broad candidate pool (50 passages). The cross-encoder reranker then jointly scores each candidate–question pair and selects the top-k results.

Combined Pipeline. The combined pipeline integrates all techniques above:

1. **Query augmentation**: A language model generates a hypothetical passage (HyDE) and three alternative phrasings in a single call.
2. **Multi-query search**: Five queries (original, three alternatives, hypothetical passage) are independently embedded. The original and alternatives use hybrid search; the hypothetical passage uses vector-only search, since synthetic text is not expected to share exact vocabulary with the corpus.
3. **Candidate pooling**: Results from all five queries are deduplicated by chunk identifier.
4. **Cross-encoder reranking**: The Cohere Rerank model scores each candidate against the original question and selects the top-k.

This is the most resource-intensive pipeline, requiring one extra LLM call, five embedding operations, five search queries, and one reranking pass.

Figure 3.2 illustrates how the combined pipeline processes a question from augmentation through multi-query retrieval, candidate pooling, and cross-encoder reranking to the final top-k passages.

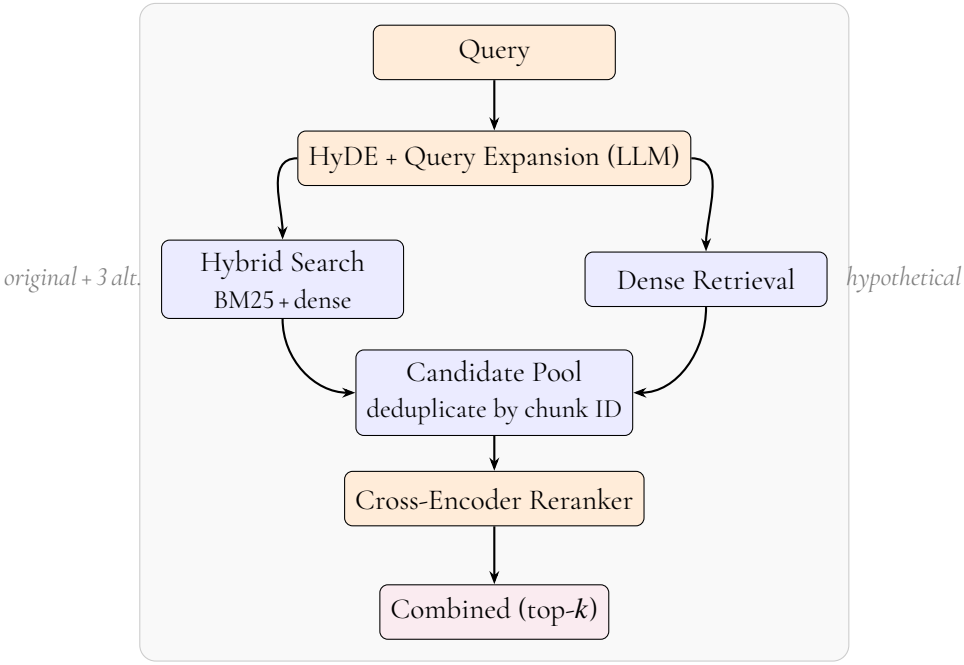


Figure 3.2: Combined pipeline query-time flow. A single LLM call generates a hypothetical passage (HyDE) and three alternative query phrasings. The original query and its alternatives are searched with hybrid retrieval (BM25 + dense); the hypothetical passage uses dense-only search, since synthetic text is not expected to share exact vocabulary with the corpus. All results are pooled, deduplicated, and rescored by a cross-encoder reranker, which selects the top-k passages.

3.2.2 LightRAG Pipelines

LightRAG [Guo et al., 2025] replaces the flat chunk index with a property knowledge graph stored in Neo4j. Three variants are evaluated: pure graph retrieval, graph retrieval combined with dense search (Mix), and the combined approach with cross-encoder reranking (Mix + Rerank).

Knowledge Graph Construction

The indexing pipeline transforms raw text into a searchable knowledge graph in four stages.

1. **Entity and Relationship Extraction.** A language model extracts entities (with name, type, and description) and undirected relationships (with keywords and description) from each chunk. A second refinement pass re-examines each chunk to catch missed items. When the same entity appears across chunks, descriptions accumulate; relationships are similarly merged with keywords collected via set union.

2. **Type Resolution and Keyword Consolidation.** Two deterministic post-processing steps run without LLM involvement: the most frequently extracted type per entity becomes canonical, and relationship keywords are deduplicated.

3. **Description Summarization.** When an entity or relationship accumulates multiple descriptions, a language model synthesizes them into a single coherent summary.

4. **Embedding Generation.** Entity and relationship embeddings are computed from concatenated descriptions and metadata, stored as 768-dimensional vectors with cosine-similarity indexes.

Query Processing

All three variants share a common retrieval procedure.

1. **Keyword Extraction.** A language model extracts *low-level keywords* (specific entities and technical terms) and *high-level keywords* (abstract themes and concepts) from the query.

2. **Dual-Level Search. Local search** uses low-level keywords to retrieve entity nodes and their neighborhoods (connecting relationships). **Global search** uses high-level keywords to retrieve relationship edges representing broad thematic matches.

3. **Neighborhood Expansion.** Entities appearing in global results but not already covered by local search are expanded with their own neighborhood lookups, bridging the local and global retrieval paths.

4. **Result Merging.** Local and global results are merged via round-robin interleaving to prevent either source from dominating, with duplicates removed.

Pipeline Variants

After graph retrieval, the three variants differ in how source text passages are selected. All variants present the generator with a combination of graph elements (entities and relationships extracted from the corpus) together with source text passages.

Pure LightRAG selects source passages solely through graph provenance: each entity and relationship stores references to the chunks from which it was extracted, and the system selects the top-k chunks ranked by the number of graph elements that reference them.

Mix adds a parallel dense search over the chunk store. Source chunks are assembled from three streams (entity provenance, relationship provenance, and dense search), merged via round-robin interleaving and capped at k.

Mix + Rerank proceeds like Mix but retains all candidates from the merge without capping, then applies the cross-encoder reranker to select the top-k passages by relevance score.

Figure 3.3 illustrates the query-time retrieval pipeline.

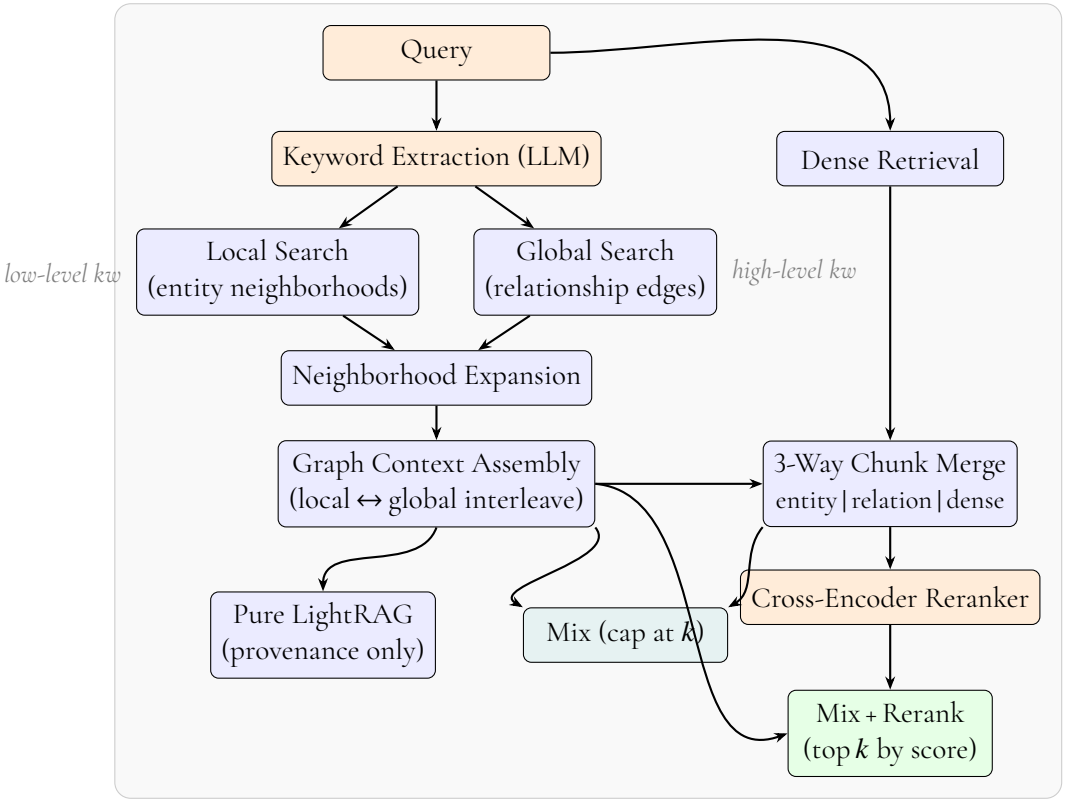


Figure 3.3: LightRAG query-time retrieval. The shared graph-context path (left) drives local and global graph searches whose results are interleaved into a graph context. Mix and Mix+Rerank additionally fuse dense retrieval chunks with entity and relation provenance chunks via a three-way merge; Mix+Rerank replaces the position-based cap with a cross-encoder reranker.

3.3 Evaluation

All nine pipelines are evaluated using LLM-judge scoring, where a language model assesses the quality of each generated answer. The two GraphRAG-Bench domains are evaluated using the benchmark's own evaluation framework, which provides standardised question types and metrics. The water treatment corpus is evaluated using a custom scheme designed to match the domain's characteristics. In both cases Claude Haiku 4.5 serves as the judge. The following subsections describe the specific metrics used for each corpus.

3.3.1 GraphRAG-Bench

The GraphRAG-Bench datasets are evaluated using the framework's own evaluation pipeline [Xiang et al., 2025], which scores generated answers across the four question types using an LLM judge. Each type targets a distinct level of retrieval and reasoning difficulty:

1. **Fact Retrieval** – questions that require locating a single, isolated knowledge point with minimal reasoning (e.g., "Which region of France is Mont St. Michel located in?"). These primarily test keyword matching and basic retrieval accuracy.
2. **Complex Reasoning** – questions that require chaining multiple knowledge points across documents via logical connections, demanding genuine integration rather than simple multi-hop lookup.
3. **Contextual Summarization** – questions that require synthesizing fragmented information from several passages into a coherent, structured answer, testing logical coherence and coverage.
4. **Creative Generation** – questions that require inference beyond the retrieved content, involving hypothetical or novel scenarios where the model must extrapolate from the available evidence.

Following the GraphRAG-Bench evaluation framework [Xiang et al., 2025], each pipeline is assessed using two groups of metrics: generation quality and retrieval quality.

Generation Metrics

Accuracy (ACC) measures semantic correctness of the generated answer relative to the reference answer, scored by an LLM judge on a 0–100 scale. *ROUGE-L* [Lin, 2004] computes the longest common subsequence between the generated and reference answers, capturing lexical overlap independent of word order. *Coverage* (Cov) quantifies how completely the answer addresses all required information points in the reference, penalising partial or omitted details. *Faithfulness Score* (FS) evaluates whether factual claims in the generated answer are grounded in the retrieved context, measuring hallucination risk.

Retrieval Metrics

Evidence Recall measures the fraction of gold-standard (benchmark-annotated reference) evidence passages that appear in the retrieved context, indicating retrieval completeness. *Context Relevance* assesses the semantic alignment between the retrieved passages and the input query, penalising irrelevant content that may dilute the generation prompt.

Not all metrics apply to every question type: Fact Retrieval and Complex Reasoning report ACC and ROUGE-L; Contextual Summarization reports ACC and Coverage; Creative Generation reports ACC, Faithfulness, and Coverage, as creative answers lack a single correct reference but must remain grounded and comprehensive [Xiang et al., 2025].

The GraphRAG-Bench baselines use GPT-4o-mini for both answer generation and LLM-judge scoring. Our pipelines instead use Claude Haiku 4.5 for both roles. Because the generation model and the judge model differ between the two setups, absolute scores are not directly comparable across the two groups. However, comparisons *within* our pipelines are fully controlled (same generation model, same judge, same questions), so relative rankings and trends are internally valid. Results are reported per question type, allowing fine-grained comparison of where each retrieval strategy excels or falls short.

3.3.2 Water Treatment Evaluation

The water treatment corpus is evaluated using a separate set of metrics, since the RAGAS-generated QA pairs do not follow the GraphRAG-Bench question type taxonomy. An LLM judge (Claude Haiku 4.5) scores each generated answer on four dimensions using a 0–10 scale (0 = lowest quality, 10 = highest). For each dimension the judge receives the inputs relevant to that criterion (drawn from the question, generated answer, retrieved context, and reference answer as appropriate) together with a rubric describing what constitutes each score level, and returns a single integer score:

- *Correctness*: factual accuracy relative to the expert-approved reference answer. Additional correct information is acceptable as long as it does not contradict the reference.
- *Relevance*: whether the answer addresses the question asked, independent of correctness or the reference answer.
- *Groundedness*: the degree to which claims in the answer are supported by the retrieved context. Claims that go beyond or contradict the retrieved information lower the score, making this metric a hallucination indicator.
- *Retrieval Relevance*: whether the retrieved context contains information useful for answering the question, evaluated independently of the generated answer.

Questions are categorized by reasoning complexity: single-hop (requiring one passage), multi-hop specific (requiring multiple passages about a specific topic), and multi-hop abstract (requiring synthesis across broader themes).

Chapter 4

Results and Discussion

Each pipeline was evaluated on 300 questions per subset, sampled proportionally across the four question types. Generation quality was assessed using ACC, ROUGE-L, Coverage, and Faithfulness Score; retrieval quality was measured using Evidence Recall and Context Relevance. Tables 4.1 and 4.2 present the full results. The original GraphRAG-Bench baselines are reproduced in Appendix A for reference; however, because those baselines use GPT-4o-mini for both generation and scoring while our pipelines use Claude Haiku 4.5, absolute scores are not directly comparable across the two setups.

4.1 Benchmark Evaluation (GraphRAG-Bench)

4.1.1 Generation Performance

Category	Pipeline	Fact Retrieval ACC	Fact Retrieval ROUGE-L	Complex Reasoning ACC	Complex Reasoning ROUGE-L	Contextual Summarize ACC	Contextual Summarize Cov	Creative Generation ACC	Creative Generation FS	Creative Generation Cov
Novel Dataset										
Chunk-Based	Dense	49.80	12.90	46.80	**14.00**	56.60	55.20	50.00	**83.20**	24.60
	Hybrid	47.50	11.70	48.70	13.40	55.10	55.40	43.80	69.40	21.80
	QE	50.70	12.40	48.00	13.20	55.30	58.20	49.00	73.20	27.50
	HyDE	50.00	**13.50**	47.70	**14.00**	55.30	55.10	44.50	79.60	21.30
	Rerank	50.60	12.20	49.40	12.50	56.10	61.40	47.60	70.10	33.30
	Combined	**51.40**	12.30	**49.70**	13.00	58.10	60.80	43.10	69.60	29.60
Graph-Based	LightRAG	38.70	4.00	37.30	7.10	49.70	42.20	56.30	68.10	35.30
	LightRAG+Mix	42.10	3.90	44.00	5.70	60.90	71.20	59.40	67.40	**39.40**
	LightRAG+Mix+Rerank	42.90	3.80	43.20	5.70	**61.70**	**73.60**	**62.10**	72.80	38.10
Medical Dataset										
Chunk-Based	Dense	54.70	18.00	55.00	12.10	63.30	60.70	55.50	71.90	29.20
	Hybrid	55.50	17.40	55.90	**12.40**	68.00	69.20	60.20	53.00	39.80
	QE	**61.30**	**18.20**	59.10	12.10	64.90	65.60	59.50	**72.60**	34.90
	HyDE	60.20	17.80	60.20	11.90	65.70	65.00	53.90	72.30	34.20
	Rerank	60.00	17.20	**60.70**	11.60	66.80	70.70	59.60	61.50	29.90
	Combined	59.40	17.30	59.10	11.80	**68.80**	67.90	57.90	68.10	37.20
Graph-Based	LightRAG	52.10	6.50	58.90	5.60	61.70	70.00	66.00	56.30	44.10
	LightRAG+Mix	52.00	6.50	59.20	5.70	65.50	**78.20**	65.10	62.80	40.40
	LightRAG+Mix+Rerank	52.70	6.60	59.80	5.60	66.20	75.10	**68.20**	62.80	**45.80**

Table 4.1: Generation evaluation results across four question types on both datasets. All pipelines use Claude Haiku 4.5 for generation and scoring. Underlined: best within category. **Bold**: best overall.

Chunk-based pipelines show relatively similar generation scores across variants (differences of less than 5 percentage points on most metrics) despite meaningful differences in retrieval quality (Section 4.1.2). Stacking components beyond reranking yields diminishing returns: the combined pipeline actually scores below QE on medical Fact Retrieval ACC (59.4% vs. 61.3%). A likely explanation is that once the top-k retrieved passages are sufficiently relevant, the generation model saturates; additional retrieval refinement cannot improve answers that are already bottlenecked by the LLM's reasoning capacity or the reference answer's specificity. A second factor is that QE concatenates the deduplicated results from its multi-query search without a top-k cap, while Combined applies the cross-encoder reranker to prune back to the final k chunks. QE therefore feeds the generator roughly twice as much context as Combined (≈9.6k vs. ≈4.6k tokens on medical), which can help when the answer relies on details outside the reranker's top-ranked candidates.

The chunk-based versus graph-based split is most visible on Creative Generation, where LightRAG+Mix+Rerank reaches 68.2% ACC on medical and 62.1% on novel, ahead of every chunk-based pipeline on both subsets (best chunk-based: 60.2% on medical, 50.0% on novel). Graph retrieval surfaces entity descriptions and relationship summaries that capture high-level domain structure, giving the LLM richer material for open-ended synthesis. Conversely, LightRAG underperforms on Fact Retrieval because the graph abstraction discards the verbatim passage detail that factoid questions demand.

ROUGE-L scores for graph-based pipelines are consistently low (3.9–7.1% vs. 11.6–18.2% for chunk-based). This reflects how answers are constructed rather than their quality: graph-based answers are synthesized from entity descriptions, producing different phrasing than the reference answers even when factually equivalent.

Taken together, the generation results suggest that retrieval quality and generation quality are partially decoupled. Within chunk-based pipelines, stacking retrieval enhancements beyond reranking yields little generation gain, as the bottleneck shifts to the LLM's reasoning capacity. For graph-based pipelines, the most important factor is question type: graph retrieval produces competitive or superior answers on creative and synthesis tasks, while chunk-based retrieval is more appropriate for factoid questions that require verbatim passage detail.

4.1.2 Retrieval Performance

Category	Pipeline	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation	
		Recall	Relevance	Recall	Relevance	Recall	Relevance	Recall	Relevance
Novel Dataset									
Chunk-Based	Dense	63.20	72.70	71.20	73.60	72.40	76.40	27.80	**55.00**
	Hybrid	57.70	62.10	75.40	72.50	73.80	77.80	28.20	45.00
	QE	71.80	74.10	79.80	74.50	77.00	**79.20**	27.00	**55.00**
	HyDE	68.70	72.60	78.50	76.40	71.00	70.80	25.80	47.50
	Rerank	**82.10**	**81.30**	**88.30**	81.30	79.10	**79.20**	34.90	50.00
	Combined	81.60	80.60	**88.30**	**81.60**	**79.60**	76.40	34.90	47.50
Graph-Based	LightRAG	23.80	13.50	19.80	17.90	27.00	18.50	78.90	10.00
	LightRAG+Mix	22.80	15.20	20.80	19.50	29.10	19.00	**89.40**	10.00
	LightRAG+Mix+Rerank	75.90	76.00	86.00	79.40	77.20	73.10	83.40	15.00
Medical Dataset									
Chunk-Based	Dense	66.10	78.10	53.90	64.50	68.70	**75.00**	51.10	57.30
	Hybrid	65.60	70.50	53.30	57.10	71.70	69.00	55.40	62.50
	QE	80.00	81.30	62.20	64.50	77.40	**75.00**	51.10	**66.70**
	HyDE	76.20	84.40	61.30	**69.60**	71.60	70.20	54.70	**66.70**
	Rerank	81.00	86.70	61.80	67.90	78.60	73.80	55.20	59.40
	Combined	**81.20**	**87.80**	**64.30**	67.90	**81.40**	73.80	58.10	58.30
Graph-Based	LightRAG	50.70	53.70	24.30	31.10	41.40	37.50	**99.40**	22.90
	LightRAG+Mix	48.10	56.90	29.20	31.10	41.60	44.60	98.90	18.80
	LightRAG+Mix+Rerank	79.10	85.20	61.80	68.20	76.90	73.80	**99.40**	22.90

Table 4.2: Retrieval evaluation results across four question types on both datasets. All pipelines use Claude Haiku 4.5 for scoring. Underlined: best within category. **Bold**: best overall.

Unlike generation, where scores plateau across chunk-based pipelines, retrieval metrics clearly separate each component's contribution. Reranking is the single most impactful addition: it lifts medical Fact Retrieval relevance from 78.1% (dense) to 86.7%, a gain larger than QE or HyDE provide individually. The cross-encoder's token-level attention between query and passage produces finer-grained relevance judgments than cosine similarity alone, which explains why it consistently outperforms first-stage retrieval enhancements that only diversify the candidate pool.

Pure LightRAG and LightRAG+Mix both score far below chunk-based pipelines on retrieval: novel Fact Retrieval relevance drops to 13.5% and 15.2% respectively. This is partly struc-

tural: the GraphRAG-Bench retrieval metrics evaluate against gold-standard (benchmark-annotated reference) *text passages*, but graph retrieval returns entity descriptions and relationship triples that may cover the same information without matching the expected passages. LightRAG+Mix+Rerank recovers dramatically (novel: 76.0%, medical: 85.2%), demonstrating that the graph does surface relevant candidates, but they are simply buried among noise when combined with dense results via naive interleaving. The cross-encoder acts as a quality filter, promoting the best candidates regardless of their retrieval source.

On Fact Retrieval, all pipelines score lower on the novel subset than on medical. The medical corpus contains precise, domain-specific entities (e.g., "left ventricular ejection fraction") that produce clean graph neighborhoods. The novel corpus contains multiple literary works sharing generic entities ("Princess", "King", "Forest"), causing neighborhood expansion to retrieve context from unrelated narratives. This cross-corpus contamination disproportionately affects LightRAG: pure LightRAG Fact Retrieval context relevance drops from 53.7% (medical) to 13.5% (novel), while chunk-based pipelines are less affected (combined drops from 87.8% to 80.6%).

4.2 Domain Evaluation (Water Treatment)

All pipelines are run at k=5 (chunk-based pipelines retrieve 5 chunks; LightRAG variants use their default 5 source chunks). Each configuration is evaluated over 5 independent runs, and we report mean ± standard deviation across runs to give a sense of variability in both the LLM-judge scores and the token-usage measurements.

Category	Pipeline	Correct.	Relev.	Ground.	Retr. Rel.	Ctx Tokens
Chunk-Based	Dense	5.65 ± 0.15	8.05 ± 0.12	9.36 ± 0.05	7.66 ± 0.02	4 436 ± 1
	Hybrid	5.72 ± 0.15	8.27 ± 0.09	9.33 ± 0.07	7.68 ± 0.03	4 431 ± 4
	QE	6.34 ± 0.07	8.81 ± 0.06	9.39 ± 0.04	7.84 ± 0.14	9 561 ± 117
	HyDE	5.80 ± 0.12	8.48 ± 0.05	9.40 ± 0.05	7.70 ± 0.07	6 173 ± 52
	Rerank	6.86 ± 0.14	8.52 ± 0.04	9.42 ± 0.04	8.60 ± 0.09	4 628
	Combined	7.01 ± 0.25	8.77 ± 0.12	**9.46 ± 0.03**	**8.68 ± 0.07**	4 647 ± 5
Graph-Based	LightRAG	5.20 ± 0.08	7.67 ± 0.11	9.06 ± 0.03	6.80 ± 0.16	21 469 ± 49
	LightRAG+Mix	5.97 ± 0.08	8.30 ± 0.02	9.23 ± 0.02	7.64 ± 0.06	21 724 ± 81
	LightRAG+Mix+Rerank	**7.21 ± 0.12**	**8.91 ± 0.05**	9.30 ± 0.04	8.24 ± 0.11	22 370 ± 189

Table 4.3: Water treatment evaluation results with k=5 (0–10 scale, mean ± std over 5 runs). Rerank candidates=50, LightRAG source chunks=5. Underlined: best within category. **Bold**: best overall.

Category	Pipeline	Single-hop	Multi-hop Abstract	Multi-hop Specific
Chunk-Based	Dense	6.34 ± 0.07	5.18 ± 0.35	5.43 ± 0.14
	Hybrid	6.44 ± 0.40	5.32 ± 0.19	5.37 ± 0.17
	QE	6.62 ± 0.14	6.17 ± 0.20	6.23 ± 0.32
	HyDE	6.50 ± 0.13	5.42 ± 0.15	5.46 ± 0.38
	Rerank	7.88 ± 0.07	6.34 ± 0.23	6.33 ± 0.18
	Combined	7.78 ± 0.28	6.72 ± 0.18	6.49 ± 0.37
Graph-Based	LightRAG	5.41 ± 0.38	5.02 ± 0.47	5.18 ± 0.08
	LightRAG+Mix	6.57 ± 0.16	5.69 ± 0.07	5.63 ± 0.26
	LightRAG+Mix+Rerank	**7.98 ± 0.04**	**6.87 ± 0.13**	**6.76 ± 0.30**

Table 4.4: Correctness by question category on the water treatment corpus at k=5 (0–10 scale, mean ± std over 5 runs). Underlined: best within category. **Bold**: best overall.

Category	Pipeline	Total In	Total Out	Gen. In	Gen. Out	Context
Chunk-Based	Dense	6 176 ± 2	311 ± 3	6 176 ± 2	311 ± 3	4 436 ± 1
	Hybrid	6 168 ± 6	311 ± 2	6 168 ± 6	311 ± 2	4 431 ± 4
	QE	13 440 ± 209	511 ± 4	13 347 ± 209	378 ± 5	9 561 ± 117
	HyDE	8 695 ± 81	599 ± 2	8 553 ± 81	332 ± 1	6 173 ± 52
	Rerank	6 443 ± 0	325 ± 1	6 443 ± 0	325 ± 1	4 628
	Combined	6 555 ± 9	604 ± 1	6 413 ± 9	334 ± 1	4 647 ± 5
Graph-Based	LightRAG	23 548 ± 86	490 ± 7	23 228 ± 86	398 ± 6	21 469 ± 49
	LightRAG+Mix	23 633 ± 115	504 ± 5	23 313 ± 115	410 ± 6	21 724 ± 81
	LightRAG+Mix+Rerank	24 447 ± 208	528 ± 5	24 127 ± 208	434 ± 6	22 370 ± 189

Table 4.5: Average token usage per question on the water treatment corpus at k=5 (mean ± std over 5 runs). **Total In/Out**: tokens summed across every LLM call (e.g. query expansion plus generation). **Gen. In/Out**: tokens for only the final answer-generation call. **Context**: retrieved passages included in the generator's prompt (subset of Gen. In).

LightRAG+Mix+Rerank achieves the highest overall correctness (7.21 ± 0.12) and relevance (8.91 ± 0.05) on the water treatment corpus, narrowly ahead of the fully stacked Combined chunk-based pipeline on both metrics (7.01 ± 0.25 and 8.77 ± 0.12). The 0.20-point correctness gap is small relative to the standard deviations involved: the confidence intervals overlap, so the ranking should be read as suggestive rather than decisive. On groundedness and retrieval relevance the picture flips: Combined leads on both (9.46 and 8.68 respectively), with LightRAG+Mix+Rerank trailing slightly (9.30 and 8.24). Graph-derived context appears to occasionally introduce material that the generator cannot fully anchor its claims to, which the chunk-based pipelines avoid. A complementary explanation is that the LLM judge may underrate groundedness when claims are restructured rather than quoted: chunk-based pipelines feed raw passages whose phrasing tends to survive into the answer, while LightRAG feeds entity and relationship summaries that the generator paraphrases, weakening the surface overlap between context and answer that the judge uses as a proxy for grounding.

The correctness-by-category breakdown (Table 4.4) shows LightRAG+Mix+Rerank leading on all three question types: single-hop (7.98 ± 0.04), multi-hop abstract (6.87 ± 0.13), and multi-hop specific (6.76 ± 0.30). The margins over the best chunk-based pipeline are modest (0.10, 0.15, and 0.27 points respectively) and again partially within the variance of the LLM-judge scores, but the consistency across categories is itself informative. Chunk-based pipelines split the runner-up positions: Rerank wins single-hop within its category (7.88 ± 0.07), while Combined wins both multi-hop categories. The water treatment domain's inherently relational structure, where treatment stages are linked by physical flows, chemical dependencies, and regulatory requirements, plausibly explains why graph traversal helps most on the multi-hop questions where reasoning across connected concepts matters.

Decomposing where LightRAG+Mix+Rerank's score actually comes from is informative. Pure LightRAG, retrieving only via graph traversal and provenance, scores 5.20 out of 10 on the correctness metric, lower than dense retrieval alone (5.65). Adding a parallel dense search (LightRAG+Mix) lifts correctness to 5.97, still below all chunk-based pipelines except dense and hybrid. The reranker is what closes the gap: LightRAG+Mix+Rerank reaches 7.21, a +1.24 jump from LightRAG+Mix and the same component that drives most of the chunk-based gains. The graph structure itself contributes at most 0.20 correctness points over the best chunk-based pipeline with the same reranker (Combined: 7.01), within the standard deviations involved. In other words, most of LightRAG+Mix+Rerank's score comes from the reranker, not from graph traversal; the graph component is at best a marginal contributor on top of a reranked retrieval pipeline.

From a deployment perspective, hybrid search, a common production-grade retrieval setup, scores 5.72 correctness, placing it among the lowest of the chunk-based pipelines. Simply adding a cross-encoder reranker on top of the same retrieval (rerank) lifts correctness to 6.86, a 1.14-point gain with minimal additional infrastructure. Stacking the remaining components on top of reranking (Combined) adds another 0.15 points, reaching 7.01. Most of the chunk-based gain therefore comes from reranking alone, mirroring the GraphRAG-Bench results.

The token usage table (Table 4.5) quantifies the cost of these improvements. Since both groups retrieve 5 source chunks at this setting, the comparison directly isolates what graph traversal adds. Dense and Hybrid consume roughly 6,170 total input tokens per question, while LightRAG+Mix+Rerank requires 24,447, a fourfold increase driven almost entirely by graph-derived context (22,370 tokens vs. 4,436 for dense). Combined achieves 7.01 correctness at 6,555 total input tokens; LightRAG+Mix+Rerank reaches 7.21 at 24,447 tokens. For practitioners operating under token-budget constraints, this is the central trade-off: a 0.20-point correctness gain (within the LLM-judge noise floor) at roughly 4× the input token cost. Reranking on top of dense retrieval offers the most favourable cost–quality position among chunk-based methods: rerank adds only 267 input tokens over dense while lifting correctness by 1.21 points (5.65 to 6.86).

Chapter 5

Conclusion

This thesis set out to answer when graph-based retrieval justifies its added complexity over simpler chunk-based alternatives. The answer is domain- and task-dependent. Across two GraphRAG-Bench domains and a Swedish water treatment corpus, cross-encoder reranking consistently emerges as the single most impactful retrieval enhancement, outweighing query expansion, HyDE, and hybrid search both individually and in combination. Stacking all chunk-based components yields diminishing returns.

LightRAG's advantage is concentrated on tasks that require synthesising relational knowledge: creative generation and contextual summarisation. On factoid questions pure graph retrieval underperforms chunk-based pipelines, because the graph abstraction trades verbatim textual detail for structural coverage. The reranker is doing most of the heavy lifting in the winning pipeline (LightRAG+Mix+Rerank): pure LightRAG scores below dense retrieval, and the cross-encoder reranker accounts for most of the gain on top of that. The graph structure itself adds at most a marginal contribution over the best chunk-based pipeline with the same reranker (around 0.20 correctness points on the water treatment corpus, within the noise floor), at roughly 4× the input token cost. Graph retrieval pays off, but only when paired with a strong reranker, and even then the marginal benefit over a fully stacked chunk-based pipeline is modest. The two findings are reconcilable: on creative generation and contextual summarization tasks, where reasoning across many connected facts matters, graph traversal does provide a real edge over reranked chunks; on factoid lookups and most everyday retrieval tasks, the reranker is doing the work and graph structure adds little.

The water treatment evaluation reinforces that these findings transfer to real-world industrial domains. The domain's interconnected treatment processes create naturally relational knowledge that graph retrieval is well positioned to exploit, and LightRAG+Mix+Rerank leads on all three question categories (single-hop, multi-hop abstract, and multi-hop specific), though by small margins.

Several limitations bound the confidence of these findings. The evaluation spans three corpora, which is a narrow empirical base; results should be treated as indicative rather than definitive. The GraphRAG-Bench results are based on a single evaluation run, which reduces the statistical reliability of individual pipeline rankings compared to the water treatment results, which were averaged over five runs. The 109-question water treatment evaluation set is also small, and observed correctness gaps of around 0.20 points are within the range where noise could influence rankings. In terms of generalisability, the reranking finding is the most consistent, holding across all three corpora and both paradigms, while the task-type specialisation of graph retrieval is more domain-sensitive and may not transfer to corpora with fundamentally different knowledge structures.

The central answer is: cross-encoder reranking delivers the most reliable and consistent gains across all corpora and question types. Graph-based retrieval adds value specifically on relational and creative tasks, but only when a reranker is already in place, and even then the additional gain over a fully stacked chunk-based pipeline is modest, making the reranker, not the graph, the critical component.

5.1 Future Work

This study evaluates three corpora across nine pipelines; scaling to more domains would strengthen the generalisability of the findings. Domains such as legal documentation, software codebases, or conversational logs present knowledge structures that differ fundamentally from both clinical guidelines and technical manuals, and it is unclear whether the observed advantage of graph-based retrieval on relational queries would persist or collapse in those settings.

All pipelines use a single generation and scoring model. This is a moderate limitation: the reranking finding, consistent across three corpora and both pipeline families, is likely robust to model choice. The graph versus chunk-based comparison is more sensitive, since a different LLM may handle structured entity descriptions versus raw text passages differently, and a different judge may apply different scoring criteria. Critically, the 0.20-point correctness margin between LightRAG+Mix+Rerank and the best chunk-based pipeline is close enough that a different judge could plausibly reverse the ranking. Rerunning the evaluation with a different LLM would test how much of the observed pattern is model-agnostic.

Finally, the token usage analysis suggests a natural extension: a formal cost–quality Pareto analysis across pipelines. The current results show that LightRAG+Mix+Rerank consumes roughly four times more input tokens than dense at k=5 while outperforming it by 1.56 correctness points on the water treatment corpus. Formalising this tradeoff, and identifying the operating point where marginal quality gains no longer justify the token cost, would give practitioners clearer budget-aware deployment guidance.

References

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A GraphRAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 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*, pages 150–158. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.eacl-demo.16.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, 2025.

Guangtong Han, Yeping Wu, Hongyu He, Zhaozhuo Jin, and Ang Li. RAG vs. GraphRAG: A systematic evaluation and key insights. *arXiv preprint arXiv:2502.11371*, 2025.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, Jose Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37, 2021. doi: 10.1145/3447772.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question an-

swering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

Siran Li et al. Enhancing retrieval-augmented generation: A study of best practices. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, 2004.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019.

Adrian Raudaschl. RAG-fusion: A new take on retrieval-augmented generation. In *arXiv preprint arXiv:2402.03367*, 2025.

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 (EMNLP-IJCNLP)*, pages 3982–3992. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1410.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, number 500-225 in NIST Special Publication, pages 109–126. National Institute of Standards and Technology (NIST), 1994.

Sentence-Transformers. Cross-encoder applications (sentence-transformers documentation). `https://sbert.net/examples/cross_encoder/applications/README.html`, 2024. Accessed: 2026-04-24.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=wCu6T5xFjeJ`.

Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019. doi: 10.1038/s41598-019-41695-z.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

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.

Chuxiong Xiang et al. GraphRAG-Bench: A comprehensive benchmark for evaluating GraphRAG pipelines. *arXiv preprint*, 2025.

Guangzhi Xiong et al. Benchmarking retrieval-augmented generation for medicine. *Findings of the Association for Computational Linguistics: ACL*, 2024.

Yue Yu et al. RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. *Advances in Neural Information Processing Systems*, 2024.

Yuxin Zeng et al. How significant are the real performance gains? an unbiased evaluation framework for GraphRAG. *arXiv preprint arXiv:2506.06331*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Yifan Zhou et al. In-depth analysis of graph-based RAG in a unified framework. *arXiv preprint arXiv:2503.04338*, 2025.

Appendices

Appendix A

GraphRAG-Bench Baseline Results

Tables A.1 and A.2 reproduce the original GraphRAG-Bench results [Xiang et al., 2025] for reference. These baselines use GPT-4o-mini for both generation and LLM-judge scoring, whereas our pipelines use Claude Haiku 4.5; absolute scores are therefore not directly comparable across the two setups.

Category	Model	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation		
		ACC	ROUGE-L	ACC	ROUGE-L	ACC	Cov	ACC	FS	Cov
Novel Dataset										
Basic RAG	RAG (w/o rerank)	58.76	37.35	41.35	15.12	50.08	82.53	41.52	47.46	37.84
	RAG (w rerank)	60.92	36.08	42.93	15.39	51.30	83.64	38.26	49.21	40.04
Graph RAG	MS-GraphRAG	49.29	26.11	50.93	24.09	64.40	75.58	39.10	55.44	35.65
	HippoRAG	52.93	26.65	38.52	11.16	48.70	85.55	38.85	71.53	38.97
	HippoRAG2	60.14	31.35	53.38	33.42	64.10	70.84	48.28	49.84	30.95
	LightRAG	58.62	35.72	49.07	24.16	48.85	63.05	23.80	57.28	25.01
	Fast-GraphRAG	56.95	35.90	48.55	21.12	56.41	80.82	46.18	57.19	36.99
	RAPTOR	49.25	23.74	38.59	11.66	47.10	82.33	38.01	70.85	35.88
	Lazy-GraphRAG	51.65	36.97	49.22	23.48	58.29	76.94	43.23	50.69	39.74
Medical Dataset										
Basic RAG	RAG (w/o rerank)	63.72	29.21	57.61	13.98	63.72	77.34	58.94	35.88	57.87
	RAG (w/ rerank)	64.73	30.75	58.64	15.57	65.75	78.54	60.61	36.74	58.72
GraphRAG	MS-GraphRAG	38.63	26.80	47.04	21.99	41.87	22.98	53.11	32.65	39.42
	HippoRAG	56.14	20.95	55.87	13.57	59.86	62.73	64.43	69.21	65.56
	HippoRAG2	66.28	36.69	61.98	36.97	63.08	46.13	68.05	58.78	51.54
	LightRAG	63.32	37.19	61.32	24.98	63.14	51.16	67.91	78.76	51.58
	Fast-GraphRAG	60.93	31.04	61.73	21.37	67.88	52.07	65.93	56.07	44.73
	RAPTOR	54.07	17.93	53.20	11.73	58.73	78.28	62.38	59.98	63.63
	Lazy-GraphRAG	60.25	31.66	47.82	22.68	57.28	55.92	62.22	30.95	43.79

Table A.1: Generation evaluation results from GraphRAG-Bench [Xiang et al., 2025], using GPT-4o-mini for generation and scoring.

Category	Model	Fact Retrieval		Complex Reasoning		Contextual Summarize		Creative Generation	
		Recall	Relevance	Recall	Relevance	Recall	Relevance	Recall	Relevance
Novel Dataset									
Basic RAG	RAG (w/o rerank)	61.37	74.66	59.80	80.82	69.08	80.05	32.48	82.84
	RAG (w/ rerank)	83.21	77.77	64.47	82.08	73.38	83.10	39.59	78.73
Graph RAG	MS-GraphRAG	61.04	27.30	73.03	39.09	82.02	43.13	53.55	35.07
	HippoRAG	80.44	56.34	87.91	58.75	90.95	59.46	65.51	46.64
	HippoRAG2	70.29	79.25	69.77	85.75	82.50	87.82	42.18	79.10
	LightRAG	73.69	33.08	85.52	37.46	87.59	38.02	71.72	38.06
	Fast-GraphRAG	64.48	47.86	73.51	55.21	78.58	49.74	56.31	46.27
	RAPTOR	62.14	54.08	67.80	61.26	75.79	63.00	58.66	58.46
	Lazy-GraphRAG	59.25	30.76	57.73	42.98	77.38	43.62	55.24	31.94
Medical Dataset									
Basic RAG	RAG (w/o rerank)	86.24	63.71	84.97	84.11	84.14	89.94	44.88	58.73
	RAG (w rerank)	87.83	64.73	86.49	85.56	85.87	91.35	45.23	60.50
Graph RAG	MS-GraphRAG	38.06	05.67	61.32	04.25	59.66	05.24	66.59	02.76
	HippoRAG	87.25	52.44	83.80	42.19	83.46	49.13	81.66	45.03
	HippoRAG2	78.70	87.96	77.00	80.94	77.40	86.85	61.12	78.64
	LightRAG	80.32	41.27	82.91	42.79	85.71	43.11	81.34	45.17
	Fast-GraphRAG	66.82	45.86	74.93	38.80	77.27	47.58	62.99	25.15
	RAPTOR	85.40	69.38	89.70	53.20	88.86	58.73	72.70	52.71
	Lazy-GraphRAG	74.29	19.90	78.65	17.50	78.72	21.35	83.41	15.09

Table A.2: Retrieval evaluation results from GraphRAG-Bench [Xiang et al., 2025], using GPT-4o-mini for scoring.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA

EXAMENSARBETE When Does Graph RAG Pay Off?
A Systematic Comparison of LightRAG and Chunk-Based RAG Pipelines
STUDENT Erik Lundberg
HANDLEDARE Pierre Nugues (LTH)
EXAMINATOR Elin A. Topp (LTH)

Kunskapsgrafer eller textsökning: vad lönar sig för AI?

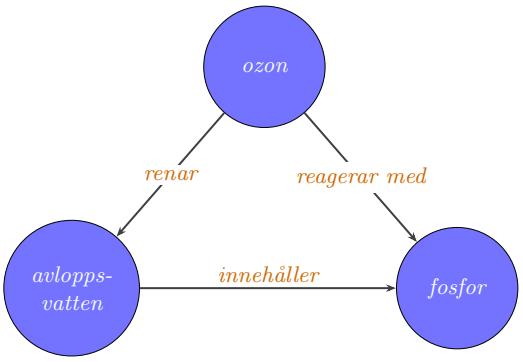
POPULÄRVETENSKAPLIG SAMMANFATTNING **Erik Lundberg**

AI-assistenter som hjälper oss hitta svar i stora dokumentsamlingar blir alltmer avancerade, men ger komplexare lösningar alltid bättre resultat? En jämförelse av nio olika sökstrategier på medicinska riktlinjer, romaner och svenska dokument om vattenrening visar att de enklaste metoderna ofta räcker långt.

När en språkmodell ska svara på en fråga om till exempel ett internt regelverk, en bipacksedel eller en teknisk manual, räcker det sällan med det modellen ”vet” själv. Risken är att den hittar på. Lösningen är att först låta en sökmotor leta fram relevanta textstycken ur dokumentsamlingen, och sedan låta språkmodellen svara utifrån just dem. Tekniken kallas *retrieval-augmented generation* (RAG) och är idag standard i alla seriösa AI-assistenter, från kundtjänstbottar till medicinska beslutsstöd.

Frågan är vilken sökmotor man ska välja. Den enklaste varianten letar efter ord och liknande meningar i texten. En mer komplex variant bygger en *kunskapsgraf*: ett nätverk där ord som ”ozon”, ”fosfor” och ”avloppsvatten” kopplas ihop med relationer som ”renar” eller ”reagerar med”. I teorin kan en sådan graf hjälpa AI:n att resonera över flera dokument samtidigt, men det kostar både tid och beräkningskraft att bygga och använda en. När är det värt det?

I mitt examensarbete jämförde jag nio olika sökstrategier (sex textbaserade och tre grafbaserade) på tre olika dokumentsamlingar: kliniska riktlinjer, äldre romaner, och tjugo svenska tekniska dokument om vattenrening från Svenskt Vatten. För vattenrening skapades 109 frågor som granskades av en expert, för att kunna mäta vilken sökstrategi som faktiskt hittade rätt svar.



Tre saker visade sig. För det första: den mest effektiva komponenten var inte den mest avancerade, utan en enkel teknik som kallas *omsortering*, där en mindre modell rangordnar träffarna efter hur väl de matchar frågan. Ytterligare tekniker ovanpå gav bara små förbättringar. För det andra: kunskapsgrafer var bättre på frågor som krävde att resonera över flera delar av texten, men sämre på enklare faktafrågor. För det tredje: det fullt utvecklade textbaserade systemet nådde nästan samma kvalitet som det grafbaserade, men med bara en fjärdedel så mycket indata till språkmodellen. Avancerat är inte alltid bäst.