The Reasoning Bottleneck in Graph-RAG: Structured Prompting and Context Compression for Multi-Hop QA

by

Yasaman Zarrinkia
Bachelor of Computer Science, University of Victoria, 2022

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

The Reasoning Bottleneck in Graph-RAG: Structured Prompting and Context Compression for Multi-Hop QA

by

Yasaman Zarrinkia
Bachelor of Computer Science, University of Victoria, 2022

Supervisory Committee

Dr. Alex Thomo, Supervisor
(Department of Computer Science)

Dr. Venkatesh Srinivasan, Co-Supervisor
(Department of Computer Science)

ABSTRACT

Multi-hop question answering requires connecting facts scattered across multiple documents, a task where standard retrieval often falls short. Graph-based retrieval-augmented generation (Graph-RAG) addresses this by building knowledge graphs from document collections and retrieving structured context that preserves entities, relations, and community summaries. Yet strong retrieval does not guarantee strong answers. This thesis studies the reasoning bottleneck in Graph-RAG and asks whether inference-time augmentations, requiring no retraining or re-indexing, can close the gap between what the retriever provides and what the model can actually use.

Evaluating KET-RAG, a leading Graph-RAG system, on three multi-hop benchmarks (HotpotQA, MuSiQue, 2WikiMultiHopQA), the experiments show that 77% to 91% of questions already have the correct answer somewhere in the retrieved context, yet accuracy reaches only 23% to 67% for a budget 8-billion-parameter model and 35% to 78% for a 70-billion-parameter baseline. Decomposing errors reveals that 73% to 84% are reasoning failures: the answer was there, but the model could not use it. Reasoning, not retrieval, is the dominant bottleneck.

To address this, the thesis studies three inference-time mechanisms. First, SPARQL-style chain-of-thought prompting decomposes questions into structured triple-pattern queries that mirror the entity–relationship layout of the retrieved context, improving accuracy by +2 to +14 percentage points. Second, graph-walk compression reduces the retrieved context by approximately 60% through knowledge-graph traversal with no additional model calls, adding +6 percentage points on average when paired with structured prompting on smaller models. Third, question-type routing selects between prompting strategies based on question structure. Surprisingly, combining all three enables a budget 8B model to match or exceed the unaugmented 70B baseline on *all three benchmarks* at approximately 12× lower inference cost. A transfer experiment on LightRAG, a second Graph-RAG system, confirms that structured prompting generalises across systems, while graph-walk compression works best when the retrieval pipeline produces clearly layered context, where some retrieved evidence is more central to the question than other material.

The main contributions are a quantitative decomposition of Graph-RAG errors into retrieval and reasoning failures, evidence that structured prompting is a system-agnostic reasoning augmentation, and the finding that low-cost inference-time interventions can close the gap between a budget model and a much larger baseline without

any change to the retriever or index.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Dr. Alex Thomo, for his guidance, encouragement, and support throughout this research. I am also very grateful to my co-supervisor, Dr. Venkatesh Srinivasan, for his feedback, time, and support during the completion of this thesis.

I would like to thank the faculty, staff, and students in the Department of Computer Science at the University of Victoria for providing a supportive academic environment during my master's studies.

Chapter 1

Introduction

Multi-hop question answering is among the most demanding tasks for language models: answering a question correctly may require identifying and connecting facts scattered across multiple documents, entities, and reasoning hops. Graph-based retrieval-augmented generation (Graph-RAG) has emerged as a promising paradigm for this setting, constructing knowledge graphs from document corpora and using graph structure to retrieve relational context [3, 11, 6]. Yet even the strongest Graph-RAG retrievers leave a gap: retrieved context that contains the gold answer does not guarantee a correct answer, because the model must still reason over that context. This thesis investigates how inference-time augmentations can close this reasoning gap without retraining the retriever or rebuilding the graph index.

1.1 Motivation

Retrieval-augmented generation (RAG) combines external retrieval with language-model generation, allowing answers to be grounded in retrieved evidence rather than relying only on parametric memory [15]. This paradigm is especially useful for question answering tasks whose answers depend on evidence not reliably stored in the model's parameters, and it has become a standard approach for knowledge-intensive NLP tasks where factual accuracy is critical.

Multi-hop question answering was chosen as the evaluation setting because it is a challenging and widely studied setting for current RAG and Graph-RAG research. Unlike single-hop questions that can often be answered from one sentence or one passage, multi-hop questions require the system to connect multiple facts across several

steps [27, 21, 10]. For example, the system may need to first identify an intermediate person, place, event, or entity, and then use that intermediate result to reach the final answer. This makes multi-hop QA difficult for language models and especially relevant for Graph-RAG, since Graph-RAG systems are designed to organize information around entities, relationships, and graph-based summaries.

In this thesis, multi-hop QA is therefore not used only as a difficult benchmark. It is used because it reflects the kind of connected reasoning that Graph-RAG is meant to support: moving from one piece of evidence to another through entities and relationships. In such settings, the required evidence is often relational rather than purely textual, which has motivated graph-based retrieval-augmented generation (Graph-RAG), where retrieval preserves richer structure such as entities, relationships, and community-level summaries [3, 11, 6].

Among recent Graph-RAG systems, KET-RAG is a useful setting for this thesis because it retrieves multi-granular context that combines entity descriptions, relationship descriptions, community summaries, and source text chunks [11]. This thesis uses KET-RAG as its primary Graph-RAG pipeline and studies whether reasoning over retrieved graph-structured context can be improved at inference time. KET-RAG achieves state-of-the-art retrieval coverage, making it an appropriate setting for studying whether the model can use the retrieved evidence effectively after retrieval has already provided substantial relevant context.

The motivation for this work comes from a simple but important observation: retrieving useful evidence is not the same as reasoning correctly over it. Even when relevant facts appear in the retrieved context, the model must still identify the right pieces of evidence, connect them across multiple hops, and avoid being distracted by irrelevant or only loosely related context. Empirically, this gap is substantial: 77–91% of questions have the gold answer present in the retrieved context, yet overall accuracy ranges from only 23–67% for the budget 8B model and 35–78% for the larger 70B model, and reasoning failures account for 73–84% of all errors [30]. In practice, KET-RAG pipelines return approximately 10,000 tokens of heterogeneous context mixing entity descriptions, relationship descriptions, community summaries, and raw text chunks, which increases the chance the needed evidence is present but makes the downstream reasoning problem more demanding.

A secondary but practically important motivation is inference cost. Reasoning over Graph-RAG outputs with large models such as Llama-3.3-70B is expensive. If lightweight inference-time interventions can bring a budget open-weight model close

to or above the performance of a much larger unaugmented baseline, this has direct implications for deployment cost. The interaction between model size, prompting strategy, and context compression is therefore part of this thesis's empirical focus [30].

This thesis therefore investigates how reasoning over Graph-RAG context can be supported more effectively without changing the underlying retrieval pipeline. The focus is not on retrieving more information, but on enabling the model to make better use of information that has already been retrieved—through structured prompting, context compression, and question-type routing applied entirely at inference time.

1.2 Related Work

Retrieval-Augmented Generation. Retrieval-augmented generation was introduced as a framework for improving factual grounding on knowledge-intensive tasks by combining neural generation with external retrieval [15]. A comprehensive survey of the field documents how the paradigm has since expanded along retrieval, augmentation, and generation axes [4]. One challenge that motivates this thesis is the *needle-in-a-haystack* problem: even when retrieved context contains the required evidence, models can fail to locate and use it—a difficulty that worsens with context length and the number of required reasoning hops [16].

Graph-RAG Systems. More recent work has extended RAG from flat-passage retrieval to graph-based retrieval, where entities and their relationships are preserved as part of the retrieval structure [3]. A key distinction among these systems is whether the retrieved output preserves explicit entity–relationship (E-R) structure. GraphRAG [3], KET-RAG [11], and LightRAG [6] all return entity descriptions, relationship triples, and community summaries, making their context amenable to structured prompting and graph-based compression. HybridRAG [18] similarly combines knowledge-graph subgraphs with vector-retrieved passages, though it has so far been evaluated only on financial-domain transcripts. Other Graph-RAG systems use graph structure to guide retrieval but deliver flat passages as output: HippoRAG [7, 8] draws on neurobiological memory models to organise a passage-retrieval index; KGP [23] and Clue-RAG [20] use graph structure for multi-document navigation while returning conventional passages; and GraphRAG-V [28] accelerates retrieval via text-chunk communities, relying on vector-based Louvain-style clustering to con-

struct those communities [29]. Xiang et al. [25] study when graph retrieval helps and identify context inflation—retrieved context that exceeds what the model can reason over effectively—as a key failure mode. This thesis focuses on the first family of systems, where E-R structure in the retrieved context opens the door to structured prompting and topology-aware compression.

Multi-Hop Question Answering. Several benchmark datasets have shaped research in multi-hop question answering. HotpotQA [27] introduced diverse and explainable multi-hop questions over Wikipedia, pairing each question with supporting facts and curated distractors. MuSiQue [21] emphasised harder compositional questions built from single-hop components requiring up to four reasoning hops, with distractors drawn from gold paragraphs of related questions rather than random documents—making them substantially harder to filter. 2WikiMultiHopQA [10] covers the widest range of reasoning types—bridge, comparison, inference, and compositional—across questions constructed from Wikidata. Together, these benchmarks highlight the importance of both retrieving and reasoning over distributed evidence, and they serve as the primary evaluation datasets in this thesis.

Chain-of-Thought and Structured Prompting. Chain-of-thought prompting improves reasoning in large language models by making intermediate steps explicit [24], and zero-shot CoT [14] showed that this benefit can be obtained without few-shot examples. Structured variants push this further by imposing formal decomposition: Decomposed Prompting [13] breaks complex tasks into modular sub-problems handled by specialised sub-prompters, and Program-of-Thought [2] disentangles computation from reasoning by generating executable programs as intermediate steps. First-order-logic reformulation has also been used as a structured reasoning scaffold for fact-checking [1], further illustrating that intermediate representations beyond natural-language CoT can support knowledge-intensive reasoning. SPARQL has been studied as a formalism for knowledge-base QA [5], where its triple-pattern syntax (`?subject predicate ?object`) aligns naturally with entity–relationship structure—but it has not previously been used as a chain-of-thought scaffold for Graph-RAG. Multi-call methods such as IRCoT [22] and Self-Ask [17] interleave reasoning with iterative retrieval; these are orthogonal to this thesis, which operates in a single-retrieval setting where Graph-RAG already provides the full entity–relationship context upfront.

Context Compression. Context compression and filtering reduce the input length presented to language models without losing answer-critical information. LLMLingua [12] compresses prompts via a learned small language model, and RECOMP [26] selectively augments retrieved passages through compression and extraction. Both methods require additional model inference. In contrast, the graph-walk compression studied in this thesis performs post-retrieval filtering via breadth-first traversal of the knowledge graph itself, requiring no additional LLM or embedding calls. This LLM-free design is enabled by the explicit topology of Graph-RAG context—a structure that flat-passage compression methods cannot exploit— and makes it directly compatible with strict cost constraints.

Question-Type Routing. Selecting different reasoning strategies based on question type has been explored in pipeline QA systems, and the benchmark datasets themselves reflect the diversity of reasoning demands: HotpotQA [27] distinguishes bridge from comparison questions, and 2WikiMultiHopQA [10] covers four distinct reasoning types. Prior work has shown that bridge questions and comparison questions benefit from different strategies. In this thesis, question-type routing applies this principle at inference time, directing each question to the prompting strategy best suited to its structure.

The Retrieval–Reasoning Gap. Several studies have observed that retrieval quality and answer quality can diverge in RAG settings. Shi et al. [19] show that language models can be distracted by retrieved content even when the correct answer is already present, and Xiang et al. [25] identify context inflation in Graph-RAG specifically as a failure mode where additional retrieved context degrades accuracy. This thesis differs from both by quantifying the gap as a structured error decomposition and by studying inference-time interventions that target the reasoning side without modifying retrieval.

Research Gap. Existing work has substantially improved retrieval through Graph-RAG systems [3, 11, 6] and has independently shown that structured chain-of-thought prompting improves reasoning [24, 13]. What remains unexplored is how structured prompting, context compression, and routing *interact* specifically when the model is given graph-structured retrieved context at inference time—and whether these lightweight, training-free augmentations can close the performance gap between a

budget small model and a much larger unaugmented baseline. This thesis addresses that intersection directly, studying all three mechanisms over KET-RAG-retrieved context with an additional transfer check on LightRAG.

1.3 Research Gap and Objectives

The central observation motivating this thesis is a striking gap between retrieval quality and answer accuracy in Graph-RAG. Experiments on KET-RAG across HotpotQA, MuSiQue, and 2WikiMultiHopQA show that 77–91% of questions already have the gold answer present in the retrieved context, yet overall accuracy reaches only 23–67% for the budget 8B model and 35–78% for the larger 70B model—and 73–84% of all errors occur on questions where the gold answer *was* in the context [30]. The culprit is not retrieval: Graph-RAG systems such as GraphRAG, KET-RAG, and LightRAG have made retrieval strong by preserving entity–relationship structure in the retrieved context [3, 11, 6]. The culprit is reasoning. Once retrieval quality becomes sufficiently high, the model's ability to locate, connect, and exploit the retrieved evidence becomes the dominant remaining source of error—a bottleneck that retrieval improvements alone cannot address.

This thesis asks whether that reasoning gap can be closed at inference time, without retraining the retriever or rebuilding the graph index. A second, practically important dimension of the gap is cost: if lightweight inference-time interventions can bring a budget open-weight model close to or above the performance of a much larger unaugmented baseline, this has direct implications for deployment. The thesis therefore studies both the accuracy and the cost side of the gap simultaneously.

Concretely, the thesis investigates the following research questions in the KET-RAG Graph-RAG setting on HotpotQA, MuSiQue, and 2WikiMultiHopQA:

RQ1: How large is the retrieval–reasoning gap when KET-RAG is used for multi-hop question answering on these benchmarks? This is studied by decomposing errors into retrieval failures and reasoning failures and by analysing performance on covered questions—those where the gold answer is present in the retrieved context.

RQ2: Does structured prompting that is explicitly aligned with entity–relationship context improve reasoning quality over KET-RAG outputs compared to a generic

prompting baseline? This is evaluated using SPARQL-style chain-of-thought prompting across multiple model sizes and datasets.

RQ3: Can low-cost inference-time interventions—specifically graph-walk context compression and question-type routing—enable a budget open-weight model (Llama-3.1-8B) to approach or match a much larger unaugmented baseline (Llama-3.3-70B) in Graph-RAG at substantially lower inference cost?

To answer these questions, the thesis pursues the following objectives:

1. Distinguish retrieval failures from reasoning failures in a Graph-RAG pipeline for multi-hop question answering.
2. Evaluate whether structured prompting aligned with entity–relationship context improves reasoning quality over Graph-RAG outputs.
3. Assess whether low-cost inference-time interventions enable smaller open-weight models to approach or match the performance of much larger baselines at substantially reduced cost.

To address these objectives, the thesis studies three inference-time mechanisms: SPARQL-style chain-of-thought prompting, graph-walk context compression, and question-type routing.

1.4 Contributions

This thesis is based on the paper "The Reasoning Bottleneck in Graph-RAG: Structured Prompting and Context Compression for Multi-Hop QA," submitted to ACL [30]. The thesis expands the paper with additional background, extended experimental analysis, and broader discussion of limitations and future work. All experimental results reported in this thesis come from experiments I conducted as part of this research. The companion ACL submission reports the same project; citations to it are used only to connect thesis claims to the submitted paper, not to indicate that results come from an external source.

In the KET-RAG Graph-RAG setting on HotpotQA, MuSiQue, and 2WikiMultiHopQA, this thesis makes the following contributions:

1. A quantitative decomposition of Graph-RAG errors into retrieval failures and reasoning failures, confirming that reasoning is the dominant bottleneck in this setting—accounting for the large majority of errors across all three benchmarks even when retrieval coverage is high (see Section 1.3).

2. A study of SPARQL-style chain-of-thought prompting as a structured reasoning scaffold for Graph-RAG, showing that it improves answer accuracy by +2 to +14 percentage points over a generic prompting baseline across model sizes and datasets—including, notably, for the 8B model, contrasting with prior findings that chain-of-thought primarily benefits much larger models.

3. An analysis of graph-walk context compression as an LLM-free post-retrieval method that reduces and focuses retrieved context by approximately 60%, and a characterisation of when it helps: it benefits smaller models that have structured reasoning guidance and larger models that gain from noise reduction, but hurts small models operating without a structured prompt.

4. The finding that combining structured prompting, graph-walk compression, and question-type routing enables a budget Llama-3.1-8B model to *match or exceed* the unaugmented Llama-3.3-70B baseline on all three benchmarks simultaneously, at approximately 12× lower inference cost.

5. A transfer-oriented check on LightRAG showing that structured prompting generalises across Graph-RAG systems, with gains of +7.6 pp on 2WikiMHQA, +5.4 pp on MuSiQue, and +2.2 pp on HotpotQA for the 8B model, while graph-walk compression's effectiveness depends on retrieval-pipeline structure rather than context size alone.

Chapter 2

Background

This chapter introduces the technical background needed for the remainder of the thesis. It begins with retrieval-augmented generation and multi-hop question answering, then narrows to graph-based retrieval systems, with particular emphasis on KET-RAG as the primary system studied in this thesis. The chapter then motivates the retrieval–reasoning gap that drives the thesis and concludes with the main reasoning-support mechanisms explored later: SPARQL-style reasoning, graph-walk compression, and question routing.

2.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) combines a language model with an external retrieval mechanism so that generation can be conditioned on retrieved evidence rather than relying only on knowledge stored in model parameters [15]. In a standard RAG pipeline, the system first retrieves documents or passages that appear relevant to an input query and then conditions answer generation on that retrieved context. Formally, given a question q, a retriever R selects a set of passages $\mathcal{D}_q = R(q, \mathcal{C})$ from a corpus $\mathcal{C}$, and a generator G produces an answer $\hat{a} = G(q, \mathcal{D}_q)$ conditioned on both the question and the retrieved evidence.

The appeal of RAG is straightforward. Parametric language models can encode a large amount of factual information during pre-training, but that information is difficult to update, not always verifiable, and not guaranteed to be sufficient for specialized or newly introduced knowledge. Retrieval offers a practical mechanism for grounding generation in external evidence, and can be updated independently

of the language model itself. This is especially important for knowledge-intensive tasks such as open-domain question answering, where correctness depends not only on fluent language generation but also on access to the right supporting facts [15].

At the same time, retrieval does not remove the need for reasoning. Retrieved context may be incomplete, noisy, redundant, or only partially relevant to the question. Even when the required evidence is present, the model must still identify the relevant pieces, connect them correctly, and distinguish useful signals from distracting information [19]. For simple factoid questions, this may be manageable with a small set of passages. For harder reasoning tasks, however, the difficulty increases substantially because evidence is often distributed across multiple documents, entities, or relations.

A useful way to think about RAG is as separating two subproblems. The first is *evidence access*: retrieving information that could support the answer. The second is *evidence use*: correctly interpreting and combining that information to produce a final answer. A system may perform well on the first subproblem while still failing on the second. This distinction is central to the present thesis, which asks what happens when retrieval has already become strong enough that downstream reasoning becomes the main remaining source of error (see Section 1.3).

2.2 Multi-Hop Question Answering

Multi-hop question answering requires a system to combine multiple pieces of evidence in order to answer a question correctly. Unlike single-hop factoid questions, where the answer may be found directly in one passage, multi-hop questions typically require intermediate reasoning steps such as entity linking, bridge-entity identification, comparison, or composition across several facts. For example, a bridge question such as *"The director of film X was born in what city?"* requires first identifying the director from one fact, then finding that person's birthplace from a second fact—two hops that must be chained together [27].

This setting is especially useful for studying reasoning because it exposes two different kinds of failure. A system may fail because it does not retrieve the necessary evidence at all (*retrieval failure*), or because it retrieves that evidence but does not assemble it into a correct reasoning chain (*reasoning failure*). Distinguishing between these two failure modes is important for understanding where improvement is still needed, and it forms the basis of the error decomposition methodology used in this thesis.

This thesis evaluates three widely used multi-hop question answering benchmarks. **HotpotQA** contains bridge and comparison questions over Wikipedia paragraphs; each question is paired with 10 paragraphs (2 gold supporting, 8 curated distractors), emphasizing explainable multi-hop reasoning with supporting facts [27]. **MuSiQue** is constructed by composing single-hop questions into multi-hop chains requiring 2–4 reasoning steps; its distractor paragraphs are drawn from gold paragraphs of related questions, making them harder to filter than random distractors, and it is generally considered the hardest of the three benchmarks [21]. **2WikiMultiHopQA** is built from Wikidata and covers the widest range of reasoning types among the three benchmarks—bridge, comparison, inference, and compositional—with each question supported by 2 or 4 gold paragraphs and 6–8 distractors [10]. Together, these datasets provide a useful and complementary testbed because they vary in style, difficulty, and reasoning demands while all requiring evidence integration beyond a single retrieved passage.

Multi-hop QA is also a stress test for retrieval-augmented systems. Even if all required facts exist somewhere in the corpus, the system must retrieve enough of them to support the reasoning chain. Moreover, the retrieved evidence must be organized in a form that the model can use effectively. As a result, multi-hop QA is not just a benchmark for retrieval quality or for generation quality in isolation, but for the interaction between retrieval and reasoning. Each additional hop compounds the chance of both a retrieval failure and a reasoning failure, which is why accuracy tends to degrade sharply with reasoning depth [21].

2.3 From RAG to Graph-RAG

Traditional RAG systems retrieve flat text units such as passages, documents, or chunks. This is often sufficient for questions whose answer can be supported by a small number of local text spans. However, many knowledge-intensive tasks involve information that is inherently relational: entities are connected through explicit or implicit links, and answering a question may require moving across those links rather than treating every passage as an independent unit.

Graph-based retrieval-augmented generation (Graph-RAG) addresses this by representing information in a more structured way. Instead of retrieving only isolated text passages, Graph-RAG systems construct a knowledge graph from the corpus and may retrieve entities, relations, triples, neighborhoods, paths, or graph-derived sum-

maries [3]. The graph $G = (V, E)$ captures entities V and typed relations E extracted from the corpus, so that retrieval can return not just passages but structured evidence such as entity descriptions, relationship triples `(subject, predicate, object)`, and community summaries derived from graph clustering. This preserves useful relational structure that flat passage retrieval discards.

This shift offers several advantages for multi-hop reasoning. First, graph structure makes entity relations more explicit: the model does not need to infer that two entities are related from prose—the relation is stated directly as a triple. Second, graph-based retrieval can provide multiple granularities of evidence, ranging from local relations to broader community-level summaries. Third, graph structure supports post-retrieval operations such as path extraction, neighborhood pruning, or topology-aware compression—the last of which is a central contribution of this thesis.

However, Graph-RAG also introduces new challenges. Structured retrieval can return richer but more heterogeneous context than a simple ranked list of passages. The language model must interpret entity descriptions, relationship descriptions, community summaries, and textual chunks together, all within a single context window that may span approximately 10,000 tokens. In other words, Graph-RAG can improve the *availability* of relevant evidence while simultaneously making the *use* of that evidence more demanding for the model. This trade-off is one of the main motivations for the present thesis.

2.4 Graph-RAG Systems Relevant to This Thesis

This section describes the three graph-based retrieval systems most relevant to the present thesis: GraphRAG as the foundational framework, KET-RAG as the primary experimental system, and LightRAG as the transfer-evaluation system.

2.4.1 GraphRAG

GraphRAG is a graph-based retrieval framework developed by Microsoft that constructs a knowledge graph from a corpus and uses graph structure to support query-focused retrieval and summarization [3]. Its pipeline extracts entities and relationships from all corpus chunks using an LLM, builds a knowledge graph, applies community detection (Leiden algorithm) to cluster entities into hierarchically nested communities, and generates community summary reports. At query time, retrieval uses either

local search (entity-centric, traversing the knowledge graph neighbourhood) or global search (community summaries for broader, thematic questions). GraphRAG's retrieved context explicitly preserves entity–relationship structure: entity descriptions, relationship triples, and community summaries are all returned alongside source text chunks. This explicit structure is what makes graph-structured prompting strategies applicable.

Its importance in this thesis is both conceptual and historical: GraphRAG represents the broader movement from flat retrieval toward structured retrieval, and KET-RAG is built directly on top of it [11]. The main limitation of the original GraphRAG is cost: full-corpus LLM-based entity extraction is expensive for large document collections, which KET-RAG addresses.

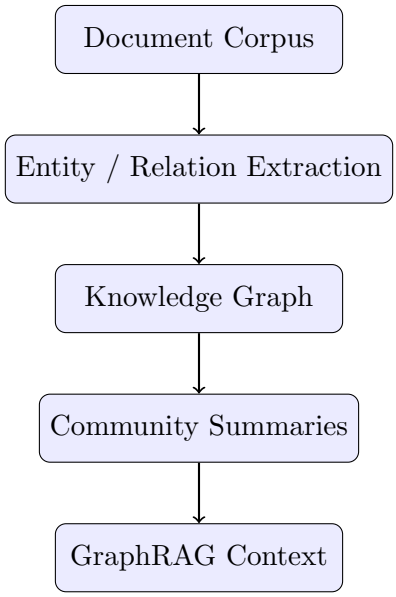

Figure 2.1: GraphRAG retrieval pipeline: entities and relations are extracted from all corpus chunks, a knowledge graph is built, community summaries are generated, and all components are returned as context.

2.4.2 KET-RAG

KET-RAG is the primary retrieval system studied in this thesis [11]. It reduces the cost of GraphRAG's full-corpus entity extraction by combining three complementary retrieval strategies, named **K**–**E**–**T**:

- **K (Keyword):** A text-keyword bipartite graph is built from all corpus chunks

without any LLM calls, providing broad lexical coverage at zero extraction cost.

- **E (Entity):** LLM-based entity and relationship extraction is applied only to a PageRank-selected subset of core chunks ($\beta = 0.8$), producing a knowledge graph that is queried via GraphRAG's local search. This concentrates LLM cost on the most information-dense chunks.
- **T (Text):** Standard text-chunk retrieval via embedding similarity provides a dense retrieval fallback for evidence not well-captured by keyword or entity channels.

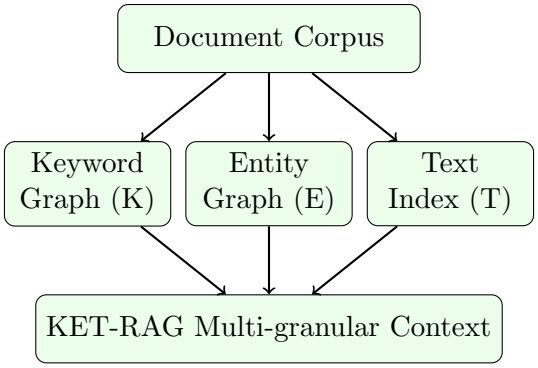

Figure 2.2: KET-RAG retrieval pipeline: three parallel channels — keyword graph (K), entity/relation graph (E), and text chunk index (T) — are combined into a multi-granular context.

A parameter $\theta \in [0, 1]$ balances the entity and keyword channels within a total token budget of λ tokens; the published defaults are $\theta = 0.5$ and $\lambda = 12{,}000$, which are used throughout this thesis [11]. The resulting retrieved context typically spans approximately 10,000 tokens and contains entity descriptions, relationship descriptions, community report summaries, and source text chunks—a rich but heterogeneous mixture that motivates the reasoning-support methods studied here.

The central importance of KET-RAG in this thesis is that it often retrieves *enough* evidence for reasoning to become the more informative bottleneck. As established in Section 1.3, high retrieval coverage coexists with substantially lower answer accuracy across all three benchmarks, leaving a gap of more than 50 percentage points on the hardest dataset. When retrieval is already strong, remaining errors are more revealing: rather than simply indicating that evidence was missed, they show that the model struggles to organise, filter, and reason over evidence that is already present. This

is exactly the condition under which the retrieval–reasoning gap becomes visible and addressable at inference time.

2.4.3 LightRAG

LightRAG is a lightweight graph-based retrieval system that uses dual-level retrieval combining low-level entity–relationship queries with high-level thematic retrieval, and integrates an incremental update mechanism for adding new documents without full re-indexing [6]. Like KET-RAG, its retrieved context preserves explicit entity–relationship structure (entity descriptions, relationship triples, community summaries), making it a suitable target for the same reasoning augmentations studied in this thesis.

LightRAG is not the main system under study but serves as a transfer-evaluation setting. The key structural difference from KET-RAG relevant to this thesis is how relevance is distributed across the retrieved context. In simple terms, a relevance gradient means that the retrieved context has different levels of usefulness: some pieces of information are very close to the question, some are only somewhat related, and some are mostly background. This matters for graph-walk compression because the method needs a signal for what to keep and what to remove. If there is a clear difference between highly relevant and weakly relevant context, compression can remove weaker material more safely.

KET-RAG tends to create this kind of layered context because its keyword, entity, and text channels return evidence with different levels of closeness to the question. LightRAG's hybrid retrieval returns more uniformly relevant material, so graph-walk compression has less obvious low-relevance material to remove. As shown in Chapter 5, this structural difference has important implications for graph-walk compression: SPARQL CoT transfers consistently across both systems, but graph-walk compression is effective only when the retrieval pipeline creates a relevance gradient that topology-based pruning can exploit.

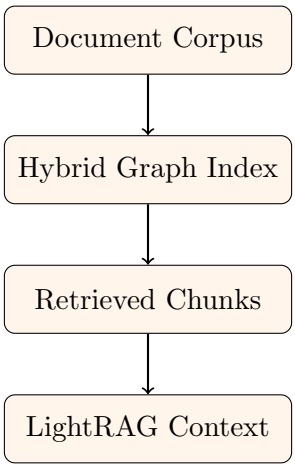

Figure 2.3: LightRAG retrieval pipeline: a hybrid graph index combines low-level entity–relationship queries with high-level thematic retrieval to produce the retrieved context.

2.5 The Retrieval–Reasoning Gap

Graph-RAG research has largely focused on improving retrieval quality: building better graph indexes, retrieving more relevant evidence, or using graph structure to expose connections that flat passage retrieval may miss [3, 11, 6]. These are important advances. However, better retrieval does not automatically imply better end-to-end answer quality.

A useful distinction is between *coverage* and *accuracy*. Coverage asks whether the evidence needed to answer the question is present in the retrieved context—a property that can be checked by testing whether the gold answer string appears in the retrieved text. Accuracy asks whether the system actually produces the correct answer. If coverage is low, poor answer quality simply reflects weak retrieval. But if coverage is high while accuracy remains substantially lower, a different bottleneck is exposed: the model is failing even after the necessary evidence is already available. As established in Section 1.3, this is precisely the situation in the KET-RAG setting studied in this thesis—reasoning, not retrieval, is the dominant source of error.

This gap is especially pronounced in Graph-RAG because retrieved context is long and heterogeneous. The model must identify relevant entities, trace links across several relations, reconcile textual evidence with graph-derived summaries, and ignore distracting context [19, 16]. As retrieval becomes stronger, these downstream

reasoning demands become increasingly visible. KET-RAG's retrieved context spans approximately 10,000 tokens on average—a needle-in-a-haystack challenge whose difficulty grows with the number of reasoning hops required.

The thesis is therefore not centred on designing a new retriever. Instead, it asks whether answer quality can be improved by supporting reasoning *after* retrieval, without retraining the retriever or rebuilding the graph index. This framing distinguishes the present work from retrieval-centred Graph-RAG research and motivates the three inference-time mechanisms introduced in the next section.

2.6 Reasoning Support in Graph-RAG

Once retrieval provides reasonably strong graph-structured context, the next question is how to help the model use that context more effectively. This thesis studies three complementary forms of inference-time reasoning support: SPARQL-style chain-of-thought prompting, graph-walk context compression, and question-type routing. None of these methods requires retraining the language model or modifying the retrieval index; they operate entirely at inference time.

A central form of reasoning support studied in this thesis is prompt engineering. In simple terms, prompt engineering means designing the instruction given to a language model so that it approaches a task in a more organized and useful way. In this thesis, prompt engineering is not treated as cosmetic wording. It is an inference-time method for guiding how the model reads and uses retrieved evidence. The model weights, retriever, and graph index remain unchanged; only the instruction format changes.

This matters in Graph-RAG because the retrieved context is long and heterogeneous. It can contain entity descriptions, relationship descriptions, community summaries, and raw text chunks. A simple prompt asks the model to read all of this and answer directly. A structured prompt instead gives the model a plan for identifying relevant entities, following relationships, and producing the final answer.

SPARQL-style chain-of-thought prompting is the first concrete structured-prompting method studied in this thesis.

2.6.1 SPARQL-Style Chain-of-Thought Prompting

SPARQL is the W3C standard query language for RDF knowledge graphs [9]. Its triple-pattern syntax expresses queries using patterns of the form `(?subject predicate`

`?object)`, where variables are bound through pattern matching against the graph. A multi-hop question can naturally be expressed as a chain of such patterns, decomposing the reasoning task into a sequence of structured lookups rather than an open-ended search.

This structure is directly applicable in Graph-RAG: KET-RAG's retrieved context already exposes entity–relationship information in the form of `(subject, predicate, object)` triples. SPARQL-style chain-of-thought prompting exploits this alignment by asking the model to formulate the question as a structured graph query whose triple patterns map onto the entity–relationship pairs in the context. Rather than searching open-ended prose for each intermediate answer, the model resolves one variable binding at a time, narrowing the search space at each hop [24]. SPARQL has previously been used as a formalism for knowledge-base QA [5], but not as a chain-of-thought scaffold for Graph-RAG retrieved context.

The only additional inference cost is slightly longer output for the query and trace. The full prompt design, including the exact prompt template and its interaction with KET-RAG's context structure, is given in Chapter 3.

2.6.2 Graph-Walk Context Compression

Even when retrieved context contains the needed evidence, that context may be too long, too redundant, or too heterogeneous for the model to navigate effectively. Graph-walk compression addresses this by exploiting the knowledge graph's own topology to isolate a compact reasoning subgraph relevant to the query, with no additional LLM or embedding calls. This distinguishes it from learned compression methods such as LLMLingua [12] and RECOMP [26], which require additional inference-time model calls and cannot exploit the explicit graph topology available in Graph-RAG context.

The core idea is to use the question itself as a starting point for graph traversal. Entities mentioned in the question serve as seeds; a breadth-first search through the knowledge graph retains structurally connected context and discards material that is topologically distant from the reasoning path. The resulting context is reorganised by hop level, creating a natural alignment with SPARQL-style chain-of-thought prompting: each hop-level section corresponds to one triple-pattern step in the structured query.

Importantly, this LLM-free design is enabled by the explicit entity–relationship

structure of Graph-RAG context—a structure that flat-passage compression methods cannot exploit. The full four-phase algorithm and its interaction with model size and prompting strategy are described in Chapter 3.

2.6.3 Question-Type Routing

Not all questions benefit equally from the same prompting strategy. Bridge questions—which require following a chain of entities across facts—are well-served by SPARQL-style triple-pattern decomposition, which naturally formalises entity chains. Comparison and inference questions, which require attribute comparison or implicit reasoning rather than a clean entity chain, benefit more from flexible natural-language decomposition [24, 13].

Question-type routing addresses this by selecting the reasoning strategy based on the classified type of the input question. A lightweight classifier assigns each question to one of a small set of types, and the appropriate prompting strategy is applied. If the chosen strategy abstains—that is, the model determines it cannot find the answer—the question is retried with the alternative strategy. This fallback mechanism keeps abstention rates low without requiring a second full inference call in most cases.

The classifier itself is a single lightweight model call requiring only a few output tokens, keeping total pipeline cost well below that of a single large-model inference call. The routing implementation and its empirical effect on accuracy and abstention rate across question types and datasets are detailed in Section 5.5.

2.7 Chapter Summary

This chapter introduced the technical background needed for the remainder of the thesis. It began with retrieval-augmented generation and the two-subproblem view of evidence access versus evidence use. It then described multi-hop question answering and the three evaluation benchmarks—HotpotQA, MuSiQue, and 2WikiMultiHopQA—along with their key differences in reasoning depth and question type diversity. It explained why graph-structured retrieval is especially relevant in multi-hop settings and introduced the three Graph-RAG systems used in this thesis: GraphRAG as the foundational framework, KET-RAG as the primary experimental setting, and LightRAG as the transfer-evaluation setting.

The chapter then formalised the central problem: the retrieval–reasoning gap, in

which high retrieval coverage coexists with substantially lower answer accuracy, with reasoning failures as the dominant source of error (as quantified in Section 1.3). This gap is especially visible in Graph-RAG because retrieved context is long, heterogeneous, and topologically structured in ways that make downstream reasoning—not retrieval—the limiting factor.

Finally, the chapter introduced the three inference-time mechanisms studied in later chapters—SPARQL-style chain-of-thought prompting, graph-walk context compression, and question-type routing—providing conceptual motivation for each. Their algorithmic detail and full empirical evaluation are presented in Chapters 3 and 5 respectively.

Chapter 3

Proposed Method

This chapter presents the method studied in this thesis. The overall goal is to improve reasoning over retrieved graph-structured context at inference time, without retraining the retriever or rebuilding the index. The chapter first describes the base Graph-RAG setting, then presents the three main components of the method: SPARQL-style chain-of-thought prompting, graph-walk compression, and question routing. It concludes with the full inference pipeline and a brief discussion of transfer to LightRAG.

3.1 Overview of the Proposed Approach

The central premise of this thesis is that strong Graph-RAG retrieval does not by itself guarantee strong downstream reasoning. Even when the retrieved context already contains the evidence needed to answer a question, the language model may still fail because the context is long, heterogeneous, and only partially relevant. As established in Section 1.3, a large majority of errors in this setting are reasoning failures—the gold answer is present in the retrieved context, but the model still fails to produce the correct answer. The proposed approach is therefore not a new retriever. Instead, it is a set of inference-time mechanisms designed to help the model reason more effectively over the context returned by an existing Graph-RAG system.

The approach has three main components. The first is SPARQL-style chain-of-thought prompting, which provides a structured intermediate representation for multi-hop reasoning. The second is graph-walk compression, which reduces the retrieved context to a more focused reasoning subgraph before answer generation. The third is question routing, which selects between prompting strategies based on ques-

tion type. These components can be used separately, but the main interest of the thesis is in how they interact within a single inference pipeline.

A key design choice is that all three components operate at inference time and require no additional model training, no changes to the retrieval index, and no re-indexing of the corpus. This makes it possible to evaluate whether meaningful gains can be obtained by improving reasoning alone, and to isolate the contribution of each component through controlled ablations.

3.2 Base Graph-RAG Setting

The primary experimental setting in this thesis uses KET-RAG as the underlying Graph-RAG retrieval pipeline [11]. As described in Chapter 2, KET-RAG retrieves a structured context that combines three channels—keyword bipartite graph (K), entity–relationship knowledge graph (E), and text-chunk retrieval (T)—within a total token budget of $\lambda = 12{,}000$ tokens and an entity–keyword balance of $\theta = 0.5$. The resulting retrieved context typically spans approximately 10,000 tokens and may include entity descriptions, relationship descriptions, community report summaries, and source text chunks [11].

The method studied in this thesis leaves the underlying retriever and its index unchanged. Instead, it focuses entirely on how the language model reasons over the retrieved context. This design choice is important because it isolates reasoning support from retrieval design: any improvement in answer quality under this setting can be attributed to inference-time reasoning support rather than to changes in the index or retrieval model.

The retrieved KET-RAG context is central to the method for two reasons. First, it frequently contains the information required to answer the question, which makes it possible to measure reasoning failures independently of retrieval failures. Second, because the context contains both graph-oriented evidence (entity and relationship descriptions, community summaries) and text-oriented evidence (raw text chunks), it creates a setting in which the form of reasoning guidance matters. A generic prompt leaves the burden of structure discovery entirely to the model. The proposed method aims to expose that structure more directly, without changing what is retrieved.

3.3 SPARQL-Style Chain-of-Thought Prompting

Building on the prompt-engineering motivation introduced in Section 2.6, the first component of the method is SPARQL-style chain-of-thought (CoT) prompting. The motivation is that many multi-hop questions can be viewed as sequences of relation-following steps, and KET-RAG's retrieved context already exposes entity–relationship structure explicitly. A generic reasoning prompt may not exploit that structure, especially when the context is long and heterogeneous. SPARQL-style prompting addresses this by encouraging the model to represent intermediate reasoning in a form that mirrors the entity–relationship layout of the retrieved context.

In this thesis, SPARQL is used as a reasoning scaffold rather than as a formal database query executed against an RDF store. In other words, the model is not running an actual database query. Instead, the SPARQL-like format is used to make the reasoning path explicit by breaking the question into relationship-following steps. The model is prompted to decompose a question into triple-pattern-like steps (maximum four patterns, plain-English predicates, no URIs, no filters, no subqueries), introduce intermediate variables, and trace those variables through the retrieved context—all in a single LLM call [9]. The prompt template used in the experiments is:

> `Step 1:` Write a simple SPARQL query (max 4 triple patterns, plain English predicates, NO URIs, NO FILTER, NO subqueries).
> `Step 2:` Trace the variable bindings for each triple pattern through the context.
> `Step 3:` Write `FINAL ANSWER: <your answer>`. If the answer is not in the context, write `FINAL ANSWER: I don't know`.

The final answer is extracted via regex matching on `FINAL ANSWER:` in the model's response. If the model determines the answer is not in the context, it responds with "I don't know," which is counted as an abstention. The only additional inference cost over a standard prompt is the longer output—typically 100–200 extra tokens for the query and variable-binding trace.

Figure 3.1 illustrates the overall process. The key structural advantage over generic chain-of-thought prompting is that triple patterns such as `(?x tributaryOf ?y)` correspond directly to the relationship descriptions in the retrieved context, converting an open-ended search over ∼10,000 tokens into a structured template-matching task where each variable binding narrows the search space for the next hop [24].

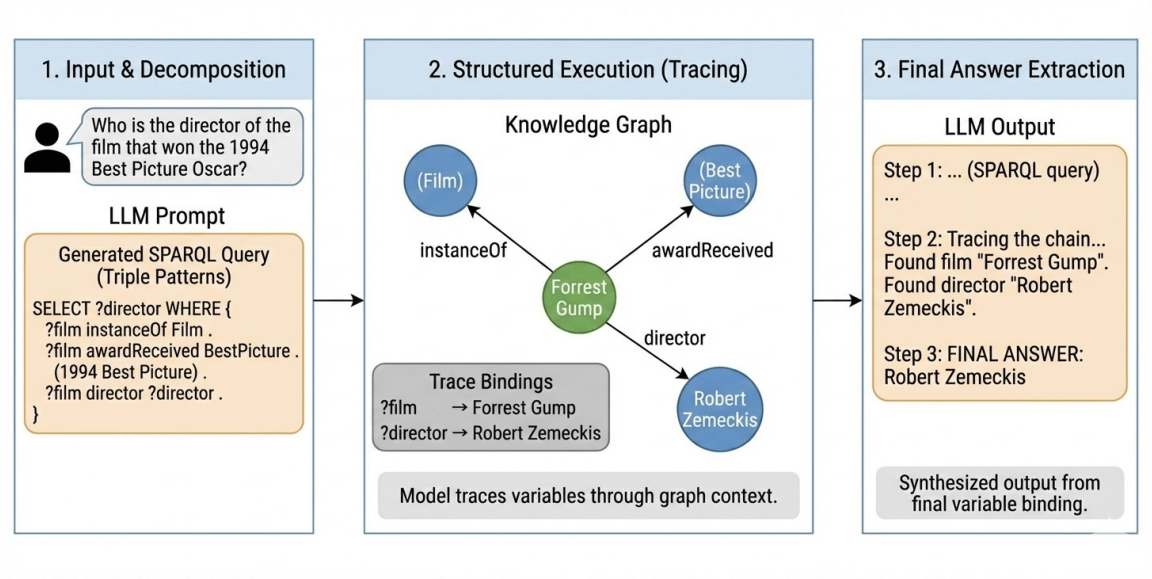

Figure 3.1: SPARQL-style chain-of-thought prompting used as a structured intermediate representation for Graph-RAG question answering. The model generates a SPARQL query whose triple patterns align with the retrieved entity–relationship context, traces variable bindings step by step, and produces the final answer in a single LLM call. Adapted from [30].

3.3.1 Illustrative Example

A simplified example clarifies the role of this prompting style. Consider a question of the form: *"Which university did the spouse of person X attend?"* A structured reasoning trace first identifies the spouse relation, then binds an intermediate variable for the spouse, and finally resolves the university attended by that bound entity. A schematic decomposition can be written as:

$$X \xrightarrow{\texttt{spouse}} ?y$$
$$?y \xrightarrow{\texttt{educated\_at}} ?z$$

The model is guided to find evidence supporting each step in the retrieved Graph-RAG context. As a concrete SPARQL-style query, this would appear as:

```
SELECT ?z WHERE {
  "X" spouse ?y .
  ?y educatedAt ?z .
}
```

The value of this structure is that it separates the reasoning path into interpretable, verifiable components rather than asking the model to jump from the question to the final answer in a single unstructured step. The model is not executing this query against a database; it is using the triple-pattern form as a roadmap for searching the retrieved context.

3.3.2 Advantages and Trade-offs

SPARQL-style prompting has two expected advantages in this setting. First, it encourages explicit multi-hop decomposition, which reduces abstention on harder questions. On MuSiQue, 8B model abstentions drop from 52.0% to 20.6% (-31.4 pp) when SPARQL CoT is applied, suggesting the structure forces the model to engage with multi-hop chains rather than declining. Second, it creates a reasoning format aligned with graph-oriented retrieved evidence, which is especially valuable for the 70B model: on 2WikiMultiHopQA the 70B model gains +12.2 pp with SPARQL CoT versus +7.6 pp with generic CoT.

At the same time, the SPARQL formalism may introduce syntactic overhead for smaller models. On 2WikiMultiHopQA the 8B model gains +14.2 pp with SPARQL CoT but +21.2 pp with generic CoT, indicating that decomposition is the primary driver and that the structured syntax adds friction for the smaller model. This trade-off directly motivates question routing, described in Section 3.5.

3.4 Graph-Walk Context Compression

The second component of the method is graph-walk compression (GW). This component addresses a distinct but complementary problem: even when the correct evidence is present in the retrieved context, reasoning may be degraded if the model must process approximately 10,000 tokens of heterogeneous material to find it—the needle-in-a-haystack problem [19]. Graph-walk compression is a topology-guided post-retrieval procedure that assembles a smaller, more focused reasoning subgraph from the retrieved context before passing it to the language model.

Crucially, this procedure uses no additional LLM or embedding calls—it is pure graph traversal and string matching. This distinguishes it from learned compression approaches such as LLMLingua [12] and RECOMP [26], which require additional inference-time models and incur their own computational cost. Graph-walk compres-

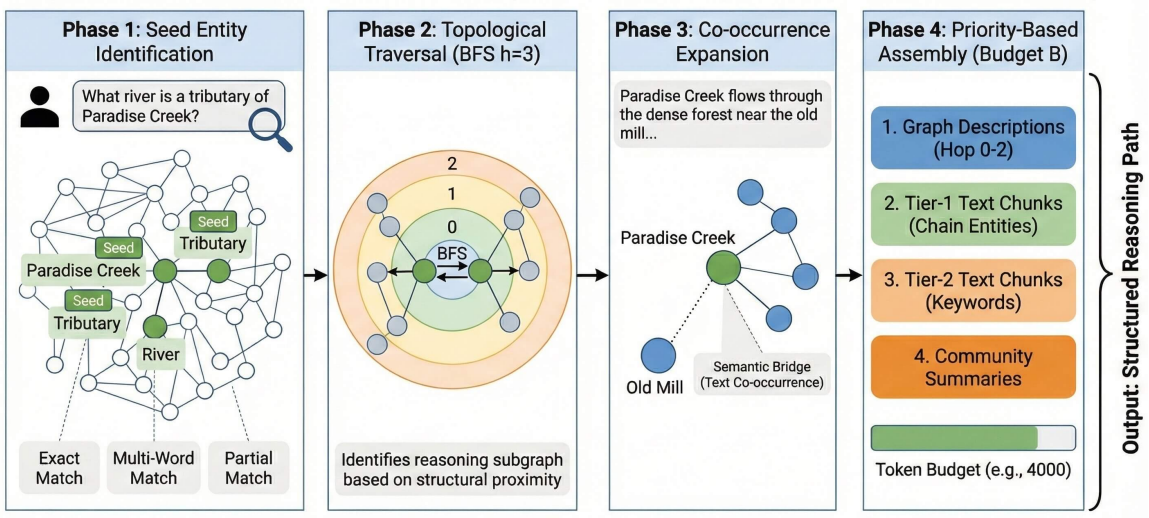

Figure 3.2: Graph-walk context compression as a topology-guided post-retrieval procedure. Starting from question-linked seed entities, the method performs bounded BFS traversal up to 3 hops, expands with semantically relevant co-occurrence bridges, and assembles a compressed context under a token budget of $B = 4{,}000$ tokens, prioritizing structurally closer evidence. No LLM calls are required. Adapted from [30].

sion instead exploits the topology already present in the KET-RAG knowledge graph, making it compatible with strict cost constraints.

3.4.1 Algorithm

The algorithm distils the retrieved context in four phases, illustrated in Figure 3.2.

Phase 1: Seed entity identification. Question entities are anchored in the retrieved context via a multi-heuristic string-matching pipeline: (a) exact substring matching (case-insensitive), (b) multi-word matching where all content words of a multi-word entity appear in the question, and (c) partial matching where a significant question word ($\geq$5 characters) appears as part of an entity name, with common stop words excluded. The matched entities become the seeds for the subsequent traversal.

Phase 2: Topological traversal. A breadth-first search (BFS) is performed from the seed entities through the knowledge graph up to a depth of 3 hops. Each reached entity is tagged with its hop distance from the nearest seed, establishing a structural proximity ranking. In practice, the BFS reaches 14–21 entities on average.

Phase 3: Co-occurrence expansion. The BFS-reached entity set is expanded via a semantic bridge: if a text chunk mentions both a BFS-reached entity and an entity not reached by BFS, the latter is added to the context. This recovers entities that are semantically related but not directly connected in the extracted graph, helping to avoid pruning bridge entities that are referenced indirectly.

Phase 4: Priority-based assembly. The compressed context is assembled within a token budget B (default $B = 4{,}000$ tokens), prioritized by structural proximity to the question:

1. Entity and relationship descriptions, ordered by hop distance from seeds (closer = higher priority).
2. Tier-1 text chunks: chunks mentioning BFS-reached entities, ranked by mention count.
3. Tier-2 text chunks: chunks sharing question keywords but no BFS-reached entities.
4. Community report summaries, if present (lowest priority).

Crucially, the output is reorganized *by hop level*, creating a natural interface with SPARQL CoT: the hop-level sections align with the triple-pattern chain in the generated query, letting the model resolve each variable binding within a localized context region rather than scanning the entire input. If no seed entities are matched, the algorithm falls back to the original uncompressed context. In practice, priority-based assembly achieves approximately 60% token reduction ($\sim$10,000 $\rightarrow$ 4,000 tokens).

3.4.2 Interaction with Model Size and Prompting Strategy

Graph-walk compression does not benefit all configurations equally. A consistent pattern emerges from the experimental results: GW helps most when the model has structured reasoning guidance (SPARQL CoT with the 8B model, +6.0 pp on average) or sufficient capacity to reason over reduced noise (70B baseline, +3.5 pp on average). Without either condition, the small model relies on exhaustive context for pattern matching, and compression removes signal rather than noise, resulting in an average loss of -2.9 pp for the 8B baseline.

This interaction is a key finding of the thesis. Graph-walk compression should not be treated as a universally beneficial step; its value depends on whether the model is equipped to exploit a focused context. The complementarity between GW and SPARQL CoT is that the hop-level reorganization of the compressed context aligns directly with the triple-pattern chain of the structured prompt, allowing the model to resolve each hop within a localized context region.

3.5 Question-Type Routing

The third component of the method is question-type routing. The motivation arises directly from the CoT ablation: different question types benefit from different prompting strategies, and no single prompt is uniformly optimal across all question types and model sizes. Routing treats prompt selection as part of the inference pipeline itself, directing each question to the strategy best suited to its structure.

3.5.1 Motivation

The ablation on 2WikiMultiHopQA shows that the 8B model prefers generic CoT over SPARQL CoT (+21.2 pp vs. +14.2 pp), while the 70B model shows the opposite preference (+12.2 pp SPARQL vs. +7.6 pp generic). Analysing by question type reveals why: bridge questions—which require following an explicit entity chain—align naturally with SPARQL triple-pattern decomposition and see the largest SPARQL CoT gains (+18.2 pp for the 8B model on 2WikiMultiHopQA bridge/comparison questions). Comparison and inference questions, which require attribute comparison or implicit reasoning rather than explicit relation following, benefit more from the flexibility of generic CoT [13]. A fixed prompting strategy therefore leaves accuracy on the table for at least one question type.

3.5.2 Implementation

The router is a single 8B LLM call (maximum 5 output tokens) that classifies the input question as *bridge*, *comparison*, or *inference*. The exact prompt used in the experiments is:

```
Classify this question as exactly one of: bridge, comparison,
inference.
```

```
Definitions:
- bridge: answer requires following an entity chain across facts
- comparison: answer compares two entities/values
- inference: answer requires implicit reasoning, not a clean
entity chain
Reply with exactly one word: bridge, comparison, or inference
Question: {question}
```

The routing policy applied after classification is:

- **Bridge questions** → SPARQL-style chain-of-thought prompting.
- **Comparison and inference questions** → generic chain-of-thought prompting.

If the selected method abstains (i.e., the model responds with "I don't know"), the question is retried with the alternative prompting strategy. The full pipeline therefore requires at most three 8B LLM calls per question: one for classification, one for the primary answer attempt, and one optional retry. This keeps total inference cost well below a single 70B call, since the 8B model costs approximately 12× less per token than the 70B model ($0.05 vs. $0.59 per million tokens).

3.5.3 Effect on Abstention and Accuracy

Routing reduces abstention rates to the lowest of any configuration: 1.0% on HotpotQA, 6.2% on MuSiQue, and 7.2% on 2WikiMultiHopQA. The gains are largest on datasets with high question-type diversity. On 2WikiMultiHopQA, routing raises 8B accuracy by +16.0 pp over the unaugmented 70B baseline—the largest single gain reported in this thesis. Without routing, SPARQL+GW surpasses the 70B baseline only on 2WikiMultiHopQA (+7.0 pp) while trailing on HotpotQA (−1.4 pp) and MuSiQue (−4.6 pp); routing closes these remaining gaps.

3.6 Integrated Inference Pipeline

The three components above combine into a single inference pipeline. For a given input question, the pipeline proceeds as follows:

1. **Retrieval.** The base Graph-RAG system (KET-RAG) retrieves graph-structured context of approximately 10,000 tokens.

2. **Compression (optional).** If graph-walk compression is enabled, the retrieved context is filtered and reorganized into a hop-level subgraph of approximately 4,000 tokens. If no seed entities are matched, the full context is retained.

3. **Routing.** A lightweight classifier (single 8B LLM call, max 5 output tokens) assigns the question to a type—bridge, comparison, or inference—and selects the corresponding prompting strategy (SPARQL CoT or generic CoT). In non-routing configurations, a fixed prompting strategy is applied.

4. **Answer generation.** The QA model generates a response using the selected prompt and the (optionally compressed) retrieved context.

5. **Fallback (routing only).** If the primary answer attempt abstains, the question is retried with the alternative prompting strategy.

6. **Answer extraction.** The final answer is extracted via regex matching on `FINAL ANSWER:`, with LLM-judged semantic equivalence as a fallback for normalization.

This pipeline design is important for interpreting the results. The thesis studies not only each component in isolation but also their interactions: structured prompting changes the form of reasoning, compression changes the form of context presented to the model, and routing changes which reasoning strategy is applied to a given question. Together, the fully augmented 8B pipeline (routing + GW + SPARQL/generic CoT) matches or exceeds the unaugmented 70B baseline on all three benchmarks at approximately 12× lower inference cost.

Table 3.1 summarises the four main configurations evaluated in the experimental study.

Table 3.1: Experimental configurations evaluated in this thesis. GW = graph-walk compression. Each configuration is run with both the 8B and 70B QA models, except Routing + GW which is evaluated with the 8B model only.

Configuration	**Description**	**GW**	**Routing**
Baseline	Full context, direct answer	×	×
Baseline + GW	Compressed context, direct answer	✓	×
SPARQL CoT	Full context, SPARQL CoT prompt	×	×
SPARQL CoT + GW	Compressed context, SPARQL CoT	✓	×
Routing + GW	Compressed context, type-adaptive prompt	✓	✓

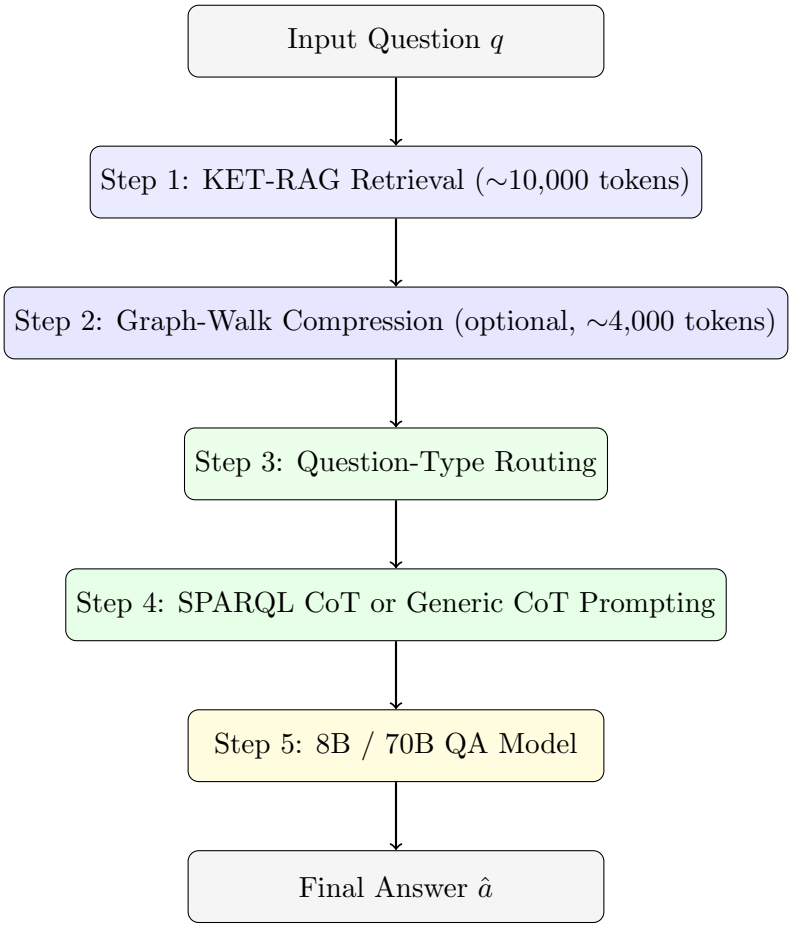

Figure 3.3: Integrated inference pipeline. Retrieval is followed by optional graph-walk compression, question-type routing, and structured prompting before a single LLM call produces the final answer. If the model abstains, the alternative prompting strategy is retried automatically.

3.7 Transfer to LightRAG

Although KET-RAG is the primary Graph-RAG system studied in this thesis, the method is also examined on LightRAG as a transfer setting [6]. The purpose is not to conduct a full parallel study on LightRAG, but to determine whether the reasoning-oriented interventions generalize beyond the KET-RAG pipeline and whether the same components transfer with similar effect sizes.

The transfer check reveals an important structural distinction. SPARQL CoT consistently improves LightRAG results (+7.6 pp on 2WikiMultiHopQA, +5.4 pp on MuSiQue, +2.2 pp on HotpotQA), confirming that structured reasoning over entity–relationship context is a system-agnostic strategy applicable whenever the retrieved context preserves explicit entity–relationship structure. Graph-walk compression, however, *hurts* on LightRAG across all three benchmarks (−8.6, −6.6, −11.0 pp), despite LightRAG returning a similar context size (median 13,000 tokens vs. 10,600 tokens for KET-RAG).

This divergence is explained by a structural difference in retrieval: KET-RAG's stratified K–E–T pipeline creates a relevance gradient across retrieved content, which graph-walk BFS traversal can exploit to identify the most answer-relevant subgraph. LightRAG's hybrid retrieval, by contrast, returns uniformly relevant material, so topology-based pruning degrades the context indiscriminately rather than filtering noise. A second, more subtle finding concerns the magnitude of SPARQL CoT gains on LightRAG. Although SPARQL CoT transfers positively to LightRAG (+7.6 pp on 2WikiMHQA, +5.4 pp on MuSiQue, +2.2 pp on HotpotQA for the 8B model), the gains are noticeably smaller than those observed on KET-RAG. This is likely attributable to a difference in how the two systems structure their retrieved context: LightRAG aggregates entities into high-level descriptions during extraction, producing a context that is less precisely organised as discrete subject–predicate–object triples. SPARQL CoT's triple-pattern decomposition is therefore a less precise match for LightRAG's context layout than for KET-RAG's, which returns explicit relationship triples as first-class retrieved items. Taken together, these two structural differences establish that SPARQL CoT is a system-agnostic reasoning strategy, but its full benefit is realised when the retrieved context is structured as explicit entity–relationship triples rather than aggregated entity descriptions. This finding establishes that graph-walk compression's effectiveness depends on retrieval-pipeline structure, not simply context size, and it scopes the applicability of the method ap-

propriately.

3.8 Chapter Summary

This chapter presented the three inference-time components of the proposed method: SPARQL-style chain-of-thought prompting, graph-walk context compression, and question-type routing. SPARQL CoT provides a structured reasoning scaffold aligned with the entity–relationship layout of KET-RAG's retrieved context, converting multi-hop reasoning into a template-matching task at a cost of only 100–200 extra output tokens. Graph-walk compression reduces the retrieved context by approximately 60% using topology-guided BFS traversal with no LLM calls, and is most effective when paired with structured prompting for smaller models. Question-type routing adaptively selects between SPARQL CoT and generic CoT based on question classification, enabling the fully augmented 8B pipeline to match or exceed the unaugmented 70B baseline at $\sim$12$\times$ lower cost. The transfer check on LightRAG shows that SPARQL CoT generalises across Graph-RAG systems, while graph-walk compression requires a retrieval-pipeline relevance gradient to be effective.

The next chapter describes the experimental setup used to evaluate these components, including the datasets, model stack, configurations, and evaluation metrics.

Chapter 4

Experimental Setup

This chapter describes the experimental setup used to evaluate the method presented in Chapter 3. It specifies the datasets, retrieval systems, model stack, evaluated configurations, metrics, and evaluation protocol used throughout the thesis. The goal of this chapter is to make clear what was tested and how performance was measured, while deferring interpretation of the results to Chapter 5.

4.1 Datasets

The thesis evaluates Graph-RAG reasoning on three multi-hop question answering benchmarks: HotpotQA, MuSiQue, and 2WikiMultiHopQA [27, 21, 10]. These datasets were chosen because they require connected reasoning across multiple facts, making them appropriate for evaluating whether Graph-RAG context helps a model answer questions that cannot usually be solved from a single sentence or passage. Together they cover a range of question types (bridge, comparison, inference, compositional), reasoning depths (2–4 hops), and distractor difficulties, as summarised in Table 4.1.

HotpotQA contains bridge and comparison questions over Wikipedia paragraphs [27]. Each question is paired with 10 paragraphs consisting of 2 gold supporting paragraphs and 8 curated distractors. It is the most commonly studied of the three benchmarks and, being limited to 2-hop reasoning, it represents the easiest setting in this thesis.

MuSiQue is constructed by composing single-hop questions into multi-hop chains requiring 2–4 reasoning steps [21]. Its distractor paragraphs are drawn from gold para-

graphs of related questions, making them semantically harder to filter than random distractors. Each question is paired with 20 paragraphs (2–4 gold, 16–18 distractors). MuSiQue is generally considered the hardest of the three benchmarks because deeper reasoning chains compound the chance of a reasoning failure at each hop.

2WikiMultiHopQA is constructed from Wikidata and covers the widest range of reasoning types among the three benchmarks—bridge, comparison, inference, and compositional [10]. Each question is paired with 10 paragraphs (2 or 4 gold, 6–8 distractors), following HotpotQA's format. KET-RAG's own evaluation does not include this dataset; it is added here specifically because of its question-type diversity, which makes it the most informative benchmark for evaluating question routing.

For each dataset, I randomly selected 500 questions. To make this random selection reproducible, I used a fixed random seed of 42. In simple terms, this means the selection was random, but once selected, the same 500 questions were kept fixed for every experiment. Every model, prompting strategy, and compression setting was evaluated on exactly the same questions, making the comparisons fair. All per-question paragraphs are pooled to construct the retrieval corpus. All comparisons are paired on the same sampled questions and retrieved contexts within each dataset.

Table 4.1: Statistics of the three benchmark datasets used in the thesis. Each benchmark is sampled to 500 questions using a fixed random seed of 42.

Dataset	**Question types**	**Hops**	**Paragraphs/Q**	**Gold paragraphs/Q**
HotpotQA	Bridge, comparison	2	10	2
MuSiQue	Compositional	2–4	20	2–4
2WikiMultiHopQA	Bridge, comparison, inference, compositional	2–4	10	2–4

4.2 Retrieval Systems

4.2.1 KET-RAG (Primary System)

The primary retrieval system is KET-RAG [11]. KET-RAG is used as the main Graph-RAG pipeline because it achieves strong context coverage while providing a demanding reasoning setting, and because its stratified K–E–T retrieval creates the relevance gradient that graph-walk compression requires. The default published parameters are used throughout: entity–keyword balance $\theta = 0.5$ and total token budget $\lambda = 12{,}000$ [11].

An important baseline comparison concerns retrieval quality. The thesis replaces KET-RAG's default commercial components (GPT-4o-mini, OpenAI embeddings) with budget open-weight alternatives: MiniLM-L6-v2 embeddings (384 dimensions, free) and Llama-3.1-8B as the indexing LLM ($0.05/M tokens). Despite this 3× cheaper indexing LLM and 4× lower embedding dimensionality, the budget stack matches or exceeds KET-RAG's published context coverage across all three benchmarks (Table 5.1), confirming that retrieval quality is not compromised.

4.2.2 LightRAG (Transfer Setting)

LightRAG is evaluated as an additional transfer setting [6]. It is not the primary system under study, but it is used to test whether the reasoning-oriented interventions transfer beyond KET-RAG. The same three benchmarks and the same 500-question samples are used. The LightRAG experiments evaluate a corresponding subset of configurations (Baseline, SPARQL CoT, Baseline + GW, SPARQL CoT + GW) to determine whether observed patterns are specific to KET-RAG or reflect a broader Graph-RAG phenomenon.

4.3 Model Stack

Table 4.2 summarises the full model stack used in the experiments. All LLM inference is performed on the Groq platform, which provides low-latency, low-cost inference for open-weight models.

Table 4.2: Model stack used in the experiments. The knowledge graph and keyword indices are built once with the budget stack and reused across all QA configurations without re-indexing. Cost estimates are based on public Groq.com token pricing.

Component	**Model**	**Dimensions**	**Cost**
Embeddings	MiniLM-L6-v2	384d	Free
Index LLM	Llama-3.1-8B (Groq)	—	$0.05/M tokens
QA (budget)	Llama-3.1-8B (Groq)	—	$0.05/M tokens
QA (standard)	Llama-3.3-70B (Groq)	—	$0.59/M tokens

The 8B model costs approximately 12× less per token than the 70B model ($0.05 vs. $0.59 per million tokens), which directly motivates the research question of whether inference-time augmentations can close the accuracy gap between the two model sizes.

These cost comparisons are based on public Groq.com token pricing at the time the experiments were conducted. Therefore, the reported cost ratio should be interpreted as an approximate provider-specific inference-cost comparison rather than a universal hardware-independent cost measurement. The total indexing cost is approximately $1 per dataset.

A key design choice is that the knowledge graph and keyword indices are *model-independent*: they are built once with the budget stack and reused across all QA configurations. This means the retrieval artifacts do not depend on whether the downstream QA model is 8B or 70B, which allows changes in answer quality to be attributed cleanly to the QA-time reasoning interventions rather than to retrieval differences.

4.4 Evaluated Configurations

Table 4.3 summarises all evaluated configurations. The main KET-RAG study evaluates four core QA configurations:

1. **Baseline:** the QA model receives the full KET-RAG-retrieved context (∼10,000 tokens) and answers directly with no structured prompting.

2. **Baseline + GW:** the same setup as Baseline, but the retrieved context is first processed by graph-walk compression (∼4,000 tokens output).

3. **SPARQL CoT:** the QA model receives the full context together with the SPARQL-style chain-of-thought prompt described in Section 3.3.

4. **SPARQL CoT + GW:** SPARQL-style chain-of-thought prompting applied to graph-walk compressed context.

Each configuration is evaluated with both QA models—Llama-3.1-8B and Llama-3.3-70B—yielding 8 runs per dataset and 24 runs in total across the three benchmarks, for 12,000 question-answer evaluations overall. All QA calls use temperature 0.3.

In addition to these four core configurations, question-type routing is evaluated as a fifth configuration combining graph-walk compression with adaptive prompt selection. The router classifies each question as bridge, comparison, or inference (single 8B call, max 5 output tokens) and selects between SPARQL CoT and generic CoT accordingly, with fallback on abstention.

Table 4.3: All evaluated configurations in the main KET-RAG study. GW = graph-walk compression. The routing configuration uses adaptive prompt selection between SPARQL CoT and generic CoT based on question type. Each configuration is run with both the 8B and 70B QA models except routing, which is 8B only.

Config.	**Prompt**	**GW**	**Routing**	**Models**
Baseline	Direct answer	×	×	8B, 70B
Baseline + GW	Direct answer	✓	×	8B, 70B
SPARQL CoT	SPARQL CoT	×	×	8B, 70B
SPARQL CoT + GW	SPARQL CoT	✓	×	8B, 70B
Routing + GW	SPARQL / Generic	✓	✓	8B

4.5 Evaluation Metrics

4.5.1 Primary Metrics

The primary metrics are **accuracy**, token-level **F1**, and **exact match (EM)**. F1 and EM are computed using SQuAD-style normalization: lowercasing, removal of articles ("a", "an", "the"), stripping punctuation, and whitespace normalization. For 8B model outputs, answers are additionally normalized to short-form responses before scoring, to improve comparability with 70B outputs that naturally tend toward concise answers.

Accuracy is determined using a two-stage heuristic-first pipeline:

1. **Heuristic matching:** normalized string matching with lowercasing, article and punctuation stripping, substring matching, and last-token comparison.

2. **LLM-judged fallback:** for cases not resolved by heuristic matching, semantic equivalence is judged by Llama-3.1-8B (e.g., resolving "NYC" vs. "New York City").

A 102-judgment audit conducted with Claude Opus 4.6 found 96.1% agreement with this evaluation pipeline, supporting its reliability.

4.5.2 Supplementary Metrics

Two additional metrics are reported throughout the results chapter.

Abstain rate measures the fraction of questions for which the model responded with "I don't know" rather than a candidate answer. Tracking abstain rates is important because changes in accuracy under SPARQL CoT are closely linked to abstention:

the structured prompt forces the model to engage with multi-hop chains rather than declining, so abstain rate reduction is a key signal of whether the method is working as intended.

Coverage measures the fraction of questions for which the gold answer string appears in the retrieved context. Coverage is used to separate retrieval failures (answer absent from context) from reasoning failures (answer present but model still incorrect). This distinction operationalises the retrieval–reasoning gap that is central to the thesis and allows performance to be reported both for all questions and for *covered questions only*—the subset where retrieval is confirmed successful.

4.5.3 Error Decomposition

To quantify the retrieval–reasoning gap, the thesis uses a two-way error decomposition. For each configuration and dataset, errors are classified as:

- **Retrieval failure:** the gold answer is absent from the retrieved context (i.e., the question is *not covered*).
- **Reasoning failure:** the gold answer is present in the retrieved context (i.e., the question *is covered*) but the model still produces an incorrect answer.

The *reasoning share* of total errors is defined as the fraction of all incorrect answers that are reasoning failures. A high reasoning share indicates that retrieval is not the primary bottleneck and that inference-time reasoning support has room to help. As established in Section 1.3, reasoning failures account for the large majority of errors across all three benchmarks—confirming that the proposed method targets the right bottleneck.

4.6 Evaluation Protocol

All configurations are evaluated on the same fixed sets of 500 questions per dataset and the same pre-retrieved contexts. Comparisons are *paired at the question level*: every configuration within a dataset sees exactly the same sampled questions and retrieved graph-structured context, so that observed differences in accuracy, F1, and EM can be attributed to the reasoning-time interventions rather than to variation in retrieval or dataset sampling.

Results are reported at three levels of granularity:

- **All questions:** overall accuracy, F1, and EM across the full 500 question sample.
- **Covered questions only:** the same metrics conditioned on coverage, separating reasoning failures from retrieval failures.
- **By question type and hop depth:** accuracy gains broken down by question type (bridge, comparison, inference, compositional) and, for MuSiQue, by hop depth (2-hop, 3-hop, 4-hop). This granularity is used to characterise when each intervention is most effective.

When question routing is enabled, the routing policy is applied per question, and a fallback is triggered if the primary strategy abstains. The full routing pipeline requires at most three LLM calls per question: one for classification, one for the primary answer, and one optional retry.

4.7 Implementation Notes

The experiments use the released KET-RAG and LightRAG codebases together with the associated indexing pipelines [11, 6]. The SPARQL-style prompt and generic CoT prompt templates, the routing classifier prompt, the graph-walk compression implementation, and all configuration files are released alongside the thesis as part of the companion experiment package [30].

Graph-walk compression is implemented as a pure post-retrieval graph procedure with no additional LLM or embedding calls. It performs seed-entity identification via multi-heuristic string matching, BFS traversal to depth 3, co-occurrence expansion, and priority-based assembly within a token budget of $B = 4{,}000$ tokens, as described in Section 3.4. The method falls back to the full uncompressed context if no seed entities are matched. An AI coding assistant (Claude Code) was used to help with experiment scripting; all scientific contributions were made by the author.

All experiments use temperature 0.3. As described in Section 4.1, the benchmark samples are selected using a fixed random seed of 42 so that the same randomly selected 500 questions are used across all configurations. The cost values used for model comparison are based on public Groq.com token pricing accessed in May 2026, as reported in Table 4.2. The released package includes the sampled benchmark subsets, the pre-retrieved contexts for both KET-RAG and LightRAG, all prompts,

configuration files, and result artefacts, allowing full reproducibility of the reported results.

4.8 Chapter Summary

This chapter described the full experimental setup used to evaluate the proposed method. Three multi-hop QA benchmarks are used—HotpotQA, MuSiQue, and 2WikiMultiHopQA—each sampled to 500 questions. KET-RAG serves as the primary retrieval system, with LightRAG as a transfer-evaluation setting. The model stack consists of MiniLM-L6-v2 embeddings, Llama-3.1-8B as both the indexing and budget QA model, and Llama-3.3-70B as the standard baseline. Five configurations are evaluated across both model sizes, with evaluation by accuracy, F1, EM, abstain rate, coverage, and a two-way error decomposition into retrieval and reasoning failures. The next chapter presents the results and analysis obtained under this setup.

Chapter 5

Results and Analysis

This chapter presents the results obtained under the experimental setup described in Chapter 4. The analysis proceeds as follows: retrieval coverage (§5.1), overall QA performance (§5.2), the effect of SPARQL CoT prompting (§5.3), the SPARQL vs. generic CoT ablation (§5.4), question-type routing (§5.5), the effect of graph-walk compression (§5.6), error decomposition (§5.7), analysis by question type (§5.8), and transfer to LightRAG (§5.9).

The main metric discussed in this chapter is answer accuracy. Accuracy means the percentage of questions for which the system produced the correct final answer. For example, on a 500-question test set, 60% accuracy means 300 questions were answered correctly. Other metrics such as F1, exact match, abstention rate, and coverage are also reported, but unless otherwise stated, improvements discussed in this chapter refer to answer accuracy.

Unless explicitly stated otherwise, all results in this chapter are from my own experimental runs using the setup described in Chapter 4.

5.1 Retrieval Quality: Coverage

Before analyzing answer quality, it is important to verify that the budget KET-RAG setup provides sufficiently strong retrieval. Table 5.1 reports context coverage for the budget setup used in this thesis and compares it with the published KET-RAG coverage values where available.

With free 384-dimensional MiniLM-L6-v2 embeddings and a $0.05/M-token indexing LLM, the budget stack matches or exceeds the published KET-RAG coverage

Table 5.1: Context coverage (%) with budget components compared to KET-RAG [11] with GPT-4o-mini + OpenAI embeddings.

Dataset	Ours	KET Low	KET High
HotpotQA	90.8 (454/500)	60.2	82.6
MuSiQue	77.2 (386/500)	77.0	79.6
2WikiMHQA	81.0 (405/500)	—	—

on all three benchmarks, despite using a 3× cheaper indexing LLM and 4× lower embedding dimensionality. On HotpotQA it exceeds even the upper reported KET-RAG value (90.8% vs. 82.6%); on MuSiQue it falls within the published range. No published comparison is available for 2WikiMHQA, so that result should be read on its own merit. Overall, the reproduced setup provides retrieval strong enough to make the central question of this thesis tractable: whether remaining errors are caused primarily by retrieval failure or by downstream reasoning failure.

5.2 QA Performance

The primary comparison across configurations in this section is based on answer accuracy; F1 and exact match (EM) are reported as supporting metrics. Tables 5.2 and 5.3 report the full KET-RAG results. Table 5.2 shows accuracy, F1, and EM on all 500 evaluated questions per benchmark; Table 5.3 conditions on covered questions only—those for which the gold answer is already present in the retrieved context. This separation distinguishes overall end-to-end performance from reasoning performance *after* retrieval has already succeeded.

The headline result is that the budget 8B model augmented with SPARQL CoT and graph-walk compression (SPARQL+GW) surpasses the unaugmented 70B baseline on 2WikiMHQA (55.8% vs. 48.8%) at approximately 12× lower per-token cost, and closes the gap substantially on HotpotQA and MuSiQue. Among all configurations, 70B + SPARQL achieves the overall best accuracy on two of the three benchmarks (80.2% on HotpotQA, 61.0% on 2WikiMHQA), while 70B + SPARQL+GW leads on MuSiQue F1 and EM (34.0% / 25.8%). Difficulty scales with reasoning depth as expected: HotpotQA (2-hop, 67–80%), 2WikiMHQA (mixed types, 30–61%), MuSiQue (2–4 hop, 19–44%).

The covered-question results (Table 5.3) are equally important. Even conditioning on successful retrieval—i.e., questions where the gold answer is already in the

Table 5.2: Results (%) on all 500 evaluated questions. GW = graph-walk compression. F1 and EM use SQuAD normalization; 8B outputs are normalized to short answers. **Bold** = best per column.

		HotpotQA			MuSiQue			2WikiMHQA		
Method	**Model**	**Acc**	**F1**	**EM**	**Acc**	**F1**	**EM**	**Acc**	**F1**	**EM**
Base	8B	67.0	59.5	49.4	23.6	18.0	12.4	31.4	29.1	24.2
+GW	8B	63.6	55.9	46.0	19.4	14.6	9.0	30.4	28.1	24.8
SPARQL	8B	70.6	60.8	49.4	28.8	22.6	15.4	45.6	36.9	31.0
+GW	8B	**76.6**	**62.2**	**50.6**	**30.6**	**22.9**	**15.8**	**55.8**	**41.1**	**34.8**
Base	70B	78.0	67.9	56.6	35.2	26.9	18.6	48.8	43.0	36.8
+GW	70B	79.2	68.6	56.0	39.8	30.8	20.6	53.6	49.0	44.0
SPARQL	70B	**80.2**	**70.6**	**57.8**	**43.8**	32.8	23.4	**61.0**	**54.9**	**48.6**
+GW	70B	79.6	70.5	**57.8**	42.2	**34.0**	**25.8**	59.8	51.5	44.0

Table 5.3: Results (%) on covered questions only (gold answer present in retrieved context). Covered counts: HotpotQA 454, MuSiQue 386, 2WikiMHQA 405. GW = graph-walk compression. **Bold** = best per column.

		HotpotQA (454)			MuSiQue (386)			2WikiMHQA (405)		
Method	**Model**	**Acc**	**F1**	**EM**	**Acc**	**F1**	**EM**	**Acc**	**F1**	**EM**
Base	8B	69.6	62.2	51.5	27.5	21.2	15.8	32.1	30.1	24.7
+GW	8B	66.3	58.3	48.0	22.6	17.0	11.4	33.1	30.8	26.9
SPARQL	8B	74.4	64.2	52.6	34.8	**27.4**	19.7	49.9	40.6	33.6
+GW	8B	**79.5**	**64.9**	**52.9**	**35.1**	26.4	**20.0**	**59.5**	**44.5**	**37.5**
Base	70B	81.3	70.9	59.0	40.3	32.5	23.9	54.8	47.9	40.7
+GW	70B	83.3	71.9	58.6	46.2	37.3	26.2	61.0	56.1	50.4
SPARQL	70B	**83.9**	**74.3**	**61.2**	**53.2**	40.4	29.6	**66.9**	**60.0**	**52.6**
+GW	70B	83.5	74.0	60.8	51.4	**42.3**	**32.5**	66.7	57.6	49.6

context—performance remains well below the coverage ceiling in most configurations. On MuSiQue, for instance, coverage is 77.2% yet the 8B baseline answers only 27.5% of covered questions correctly, a gap of roughly 50 percentage points. This pattern confirms that retrieval alone is not the main limiting factor; a substantial share of errors arises from reasoning failures over already-retrieved evidence.

5.3 Effect of SPARQL CoT Prompting

Table 5.4 isolates the effect of SPARQL CoT by comparing it to the corresponding non-GW baseline. Accuracy gains are consistent across both model sizes and all three

benchmarks.

Table 5.4: Effect of SPARQL CoT: accuracy Δ (pp) and abstain Δ (pp) vs. baseline, without GW.

Model	**Hotpot**	**MuSiQue**	**2Wiki**
Accuracy Δ (pp)			
8B	+3.6	+5.2	+14.2
70B	+2.2	+8.6	+12.2
Abstain Δ (pp)			
8B	−6.4	−31.4	−24.4
70B	+0.4	−8.2	−4.2

The gains are largest on the harder benchmarks—+14.2 pp (8B) and +12.2 pp (70B) on 2WikiMHQA, and +5.2 pp (8B) and +8.6 pp (70B) on MuSiQue—where multi-hop reasoning chains are longer and harder to execute without explicit decomposition. Notably, gains are consistent across both model sizes—including the 8B model— contrasting with prior work suggesting that chain-of-thought prompting primarily benefits models above approximately 100B parameters [24]. This suggests that when retrieved context already exposes explicit entity–relationship structure, the decomposition scaffold itself drives the improvement rather than raw model capacity. The abstain rate reductions are equally revealing. On MuSiQue, 8B abstentions drop from 52.0% to 20.6% (−31.4 pp), suggesting that SPARQL-style triple-pattern decomposition forces the model to engage with multi-hop chains rather than declining when the reasoning path is unclear. In contrast, the 70B model on HotpotQA shows a marginal abstain *increase* (+0.4 pp), consistent with it already having low abstain rates and strong baseline accuracy on the easiest benchmark.

5.4 CoT Ablation: SPARQL vs. Generic Decomposition

A key question is whether the benefit of SPARQL CoT comes from the SPARQL formalism specifically or from decomposition in general. Table 5.5 compares SPARQL CoT with a generic chain-of-thought (CoT) prompt on 2WikiMHQA, the benchmark with the highest question-type diversity.

Table 5.5: CoT ablation on 2WikiMHQA (without GW).

Method	**8B Acc**	**8B Abs**	**70B Acc**	**70B Abs**
Baseline	31.4	46.0	48.8	34.8
Generic CoT	52.6	25.2	56.4	36.8
SPARQL CoT	45.6	21.6	61.0	30.6
Routing	**58.4**	**14.2**	**66.4**	**26.8**

Both CoT methods improve substantially over the baseline, confirming that *decomposition itself* is the primary driver. The relative ranking, however, depends on model size. The 8B model favours generic CoT (+21.2 pp) over SPARQL CoT (+14.2 pp): the SPARQL syntax likely imposes additional overhead on the smaller model that the looser natural-language format avoids. The 70B model shows the opposite preference—+12.2 pp with SPARQL vs. +7.6 pp with generic CoT—because it can leverage the structural alignment between triple patterns and KET-RAG's entity–relationship context. SPARQL CoT also reduces abstain rates more consistently across both model sizes (21.6% vs. 25.2% for 8B; 30.6% vs. 36.8% for 70B).

The routing result in Table 5.5 is the most notable: by directing bridge questions to SPARQL CoT and comparison/inference questions to generic CoT, the 8B routing policy reaches 58.4% accuracy—surpassing the 70B baseline (48.8%) without any increase in model size. This confirms that there is no single universally optimal prompting strategy, and that adaptive selection based on question type is more effective than committing to either format alone.

5.5 Question-Type Routing

Because the preferred CoT strategy depends on question type and model size, the thesis deploys question-type routing combined with graph-walk compression across all three benchmarks. Table 5.6 compares the resulting 8B pipeline against the unaugmented 70B baseline.

The routing + GW pipeline is the strongest overall finding of the thesis. The 8B model matches or exceeds the unaugmented 70B baseline on all three benchmarks at approximately 12× lower per-token cost. The effect is largest on 2WikiMHQA (+16.0 pp accuracy, −27.6 pp abstain), where question-type diversity gives the routing policy the most leverage. Without routing, the SPARQL+GW 8B configuration already surpasses the 70B baseline on 2WikiMHQA (+7.0 pp) but trails on HotpotQA

Table 5.6: 8B with routing + GW vs. unaugmented 70B baseline on KET-RAG. Routing classifies each question, applies the preferred CoT, and retries with the alternative strategy if the first attempt abstains.

	Accuracy (%)			Abstain (%)		
Dataset	**8B**	**70B**	Δ	**8B**	**70B**	Δ
HotpotQA	**79.8**	78.0	+1.8	**1.0**	5.6	−4.6
MuSiQue	**35.6**	35.2	+0.4	**6.2**	16.2	−10.0
2WikiMHQA	**64.8**	48.8	**+16.0**	**7.2**	34.8	**−27.6**

(−1.4 pp) and MuSiQue (−4.6 pp); routing closes these remaining gaps. Abstain rates drop to the lowest of any evaluated configuration: 1.0% on HotpotQA, 6.2% on MuSiQue, and 7.2% on 2WikiMHQA, compared to 5.6%, 16.2%, and 34.8% for the 70B baseline.

Implementation details. The router is a single 8B call (max 5 output tokens) that classifies each question as *bridge*, *comparison*, or *inference*. Bridge questions are directed to SPARQL CoT; comparison and inference questions are directed to generic CoT. If the selected strategy abstains, the question is retried with the alternative. The complete pipeline requires at most three 8B calls per question (classify, answer, optional retry), keeping total inference cost well below a single 70B call.

5.6 Effect of Graph-Walk Compression

Graph-walk compression has a more nuanced effect than SPARQL CoT. Rather than helping uniformly, its benefit depends on both model size and prompting strategy, as shown in Table 5.7.

Table 5.7: Effect of graph-walk compression: accuracy Δ (pp) vs. the corresponding non-GW configuration.

Method	**Model**	**Hot.**	**Mus.**	**2Wi.**	**Avg**
Base	8B	−3.4	−4.2	−1.0	−2.9
Base	70B	+1.2	+4.6	+4.8	+3.5
SPARQL	8B	+6.0	+1.8	+10.2	+6.0
SPARQL	70B	−0.6	−1.6	−1.2	−1.1

A consistent interaction pattern emerges. GW helps most for SPARQL + 8B (+6.0 pp average, +10.2 pp on 2WikiMHQA) and Base + 70B (+3.5 pp average).

It hurts Base + 8B (−2.9 pp average), because the small model without structured guidance relies on exhaustive context for pattern matching and compression removes useful signal. For SPARQL + 70B, GW incurs only a −1.1 pp average accuracy loss while reducing input tokens by approximately 60% (∼10,000 → ∼4,000 tokens), which is a favourable accuracy–cost trade-off given the 70B model's higher per-token price.

The underlying mechanism is a complementarity between compression structure and reasoning guidance. When GW organizes the compressed context by hop level, it aligns directly with the triple-pattern chain in the SPARQL CoT prompt, letting the model resolve each variable binding within a localized context region. Without structured reasoning, this alignment is unexploited and the smaller model benefits more from having the full context available for pattern matching.

5.7 Error Decomposition

A central argument of the thesis is that Graph-RAG failures are predominantly reasoning failures rather than retrieval failures. Table 5.8 quantifies this by reporting the reasoning share of total errors—the fraction of all incorrect answers where the gold answer was already present in the retrieved context.

Table 5.8: Reasoning share of total errors (%): incorrect answers where the gold answer *was* present in the retrieved context.

Config	**Hotpot**	**MuSiQue**	**2Wiki**
8B Baseline	84	73	80
70B + SPARQL CoT	74	64	69

As established in Section 1.3, reasoning failures account for 73–84% of 8B baseline errors across the three benchmarks. Even in the stronger 70B + SPARQL CoT configuration, the reasoning share remains at 64–74%, indicating that reasoning failures persist even after structured prompting substantially reduces them. This is direct evidence for the retrieval–reasoning gap: when the model fails despite the gold answer being available, the problem is not that retrieval was insufficient but that the model could not use the available evidence effectively.

The pattern in Table 5.3 reinforces this conclusion. On HotpotQA, 70B + SPARQL CoT reaches 83.9% on covered questions—approaching the 90.8% coverage ceiling—

suggesting that for 2-hop questions the remaining errors are largely retrieval gaps. On MuSiQue, however, 70B + SPARQL CoT reaches only 53.2% on covered questions despite 77.2% coverage, a gap of roughly 24 points that cannot be attributed to retrieval failure. This confirms that deeper reasoning chains are harder to reason over even when the evidence is present.

5.8 Analysis by Question Type

Tables 5.9 and 5.10 break down gains by question type and hop depth. This analysis reveals when each intervention is most effective and directly motivates the routing design.

Table 5.9: SPARQL CoT accuracy improvement (pp) over baseline, by question type. HotpotQA: bridge (398), comparison (102). 2WikiMHQA: compositional (205), bridge_comparison (121), comparison (120), inference (54). MuSiQue: by hop depth (264/146/90).

	HotpotQA		**2WikiMHQA**				**MuSiQue**		
Model	Bri.	Comp.	Comp.	BrCmp.	Cmp.	Inf.	2h	3h	4h
8B	+3.0	+5.9	+16.6	+18.2	+10.9	+3.7	+8.7	−0.6	+4.5
70B	+2.3	+2.0	+4.9	+**28.1**	+12.5	+3.7	+9.4	+5.5	+11.1

The largest SPARQL CoT gains appear on chain-following question types. On 2WikiMHQA, bridge_comparison questions—which require following an entity chain and comparing attributes—see +18.2 pp (8B) and +28.1 pp (70B). Inference questions, which require more implicit reasoning, show the smallest gains (+3.7 pp for both models), confirming the design intuition of SPARQL CoT: triple-pattern decomposition naturally formalizes entity chains and is most effective when the question structure matches the formalism. On MuSiQue, the 70B model shows consistent gains across all hop depths, with the largest at 4-hop (+11.1 pp), where structured decomposition helps most with the longest reasoning chains. The 8B model shows a marginal loss at 3-hop (−0.6 pp), suggesting that the SPARQL formalism overhead can occasionally outweigh the decomposition benefit for harder intermediate-depth chains. Overall, these results establish how far a budget model can go: with SPARQL CoT, the 8B model recovers meaningful accuracy even at 4-hop depth (+4.5 pp), suggesting that structured decomposition partially compensates for the capacity limitations that typically cause small models to fail on long reasoning chains. The

3-hop dip reflects the sweet spot where the SPARQL formalism overhead temporarily exceeds its decomposition benefit, not a general inability to handle deep chains.

Table 5.10: GW accuracy Δ (pp) by question type / hop depth. Counts in parentheses.

Dataset	Category	SPARQL+8B	Base+70B
Hotpot	Bridge (398)	+4.3	+1.2
	Comparison (102)	+12.8	+1.0
2Wiki	Composit. (205)	+5.8	−1.0
	Br-Comp. (121)	+11.6	+17.4
	Comparison (120)	+13.3	+2.5
	Inference (54)	+16.7	+3.7
MuSiQue	2-hop (264)	−1.9	−1.2
	3-hop (146)	+6.8	+11.7
	4-hop (90)	+4.4	+10.0

Graph-walk compression similarly benefits deeper and more structured reasoning types. On MuSiQue, GW is slightly negative at 2-hop (−1.9 pp for SPARQL+8B, −1.2 pp for Base+70B) but strongly positive at 3-hop (+6.8, +11.7 pp) and 4-hop (+4.4, +10.0 pp). This hop-depth pattern reveals the primary mechanism of graph-walk compression: it is not a generic noise filter but a reasoning-depth amplifier. For shallow 2-hop questions, the full context is already tractable and compression only removes useful signal. For deeper chains, the compressed hop-level structure reduces the needle-in-a-haystack difficulty enough that the model can complete each reasoning step before its attention is diluted by irrelevant context. On 2WikiMHQA, bridge_comparison questions benefit most from GW for the 70B baseline (+17.4 pp), while inference questions gain most from GW + SPARQL CoT (+16.7 pp). Together, these breakdowns confirm that routing over question types is more effective than any single fixed strategy—and that the hop depth of a question is a reliable signal for whether graph-walk compression will help.

5.9 Transfer to LightRAG

To test whether the results are specific to KET-RAG, the full experiment is replicated on LightRAG across all three benchmarks. Table 5.11 reports the results.

Two findings stand out. First, SPARQL CoT consistently improves LightRAG accuracy across both model sizes and all three benchmarks: +2.2 pp on HotpotQA,

Table 5.11: LightRAG generality experiment (%) on 500 paired questions per dataset. "Covered" = gold answer present in LightRAG context (HotpotQA 450, MuSiQue 374, 2WikiMHQA 417). GW = graph-walk compression. **Bold** = best per model group. F1/EM use SQuAD normalization.

		HotpotQA			**MuSiQue**			**2WikiMHQA**		
Method	**Model**	**Acc**	**F1**	**EM**	**Acc**	**F1**	**EM**	**Acc**	**F1**	**EM**
Base	8B	62.2	55.1	45.2	26.6	19.3	12.8	32.2	29.0	24.6
+GW	8B	54.4	47.8	38.4	25.4	18.1	11.8	35.2	31.9	27.6
SPARQL	8B	**66.8**	**57.5**	**46.6**	**37.4**	**26.9**	**18.6**	**57.4**	**44.3**	**36.2**
+GW	8B	66.2	55.2	44.4	33.6	23.5	15.8	56.4	44.6	37.8
Base	70B	79.8	68.1	55.6	41.6	32.0	24.2	59.8	49.1	41.8
+GW	70B	68.8	59.9	49.4	35.0	25.7	17.2	51.2	45.8	39.2
SPARQL	70B	**82.0**	**71.6**	**57.8**	**47.0**	**36.5**	**28.0**	**67.4**	**58.1**	**49.0**
+GW	70B	64.8	56.5	45.4	36.6	29.7	23.6	50.2	45.0	37.6

+5.4 pp on MuSiQue, and +7.6 pp on 2WikiMHQA for the 8B model, confirming that structured decomposition transfers whenever the retrieved context contains entity–relationship structure. Second, graph-walk compression hurts accuracy on LightRAG across all configurations, in direct contrast to its mostly positive effect on KET-RAG. The accuracy losses range from −6.6 pp (MuSiQue, Base+70B) to −11.0 pp (HotpotQA, Base+70B).

Crucially, this divergence is not explained by context size: LightRAG's median context is ∼13,000 tokens vs. ∼10,600 tokens for KET-RAG, so LightRAG actually retrieves *more* tokens. The difference is structural. As defined in Section 2.4.3, a relevance gradient means that the retrieved context contains clearer and weaker layers of relevance. KET-RAG's β-stratified retrieval creates this kind of relevance gradient by separating core chunks from more peripheral chunks, and graph-walk BFS traversal can exploit that structure to identify the answer-relevant subgraph. LightRAG's hybrid keyword + vector retrieval returns more uniformly relevant material, so topology-based pruning removes useful content as readily as noise. This finding scopes the applicability of GW compression precisely: it is effective when the retrieval pipeline creates a structural relevance gradient, not simply when the context is long.

5.10 Discussion of Findings

Taken together, the results support four main conclusions.

Retrieval quality is not the bottleneck. The budget KET-RAG setup achieves 77–91% context coverage across all three benchmarks, confirming that the experimental setting is not one where poor end-to-end performance can be explained by weak retrieval. The coverage–accuracy gap of over 50 percentage points on MuSiQue is almost entirely attributable to reasoning failures.

Decomposition is the primary lever. SPARQL CoT consistently improves performance, including for 8B models, contrasting with prior findings that CoT primarily benefits models above ∼100B parameters [24]. The ablation confirms that decomposition itself drives most of the benefit; the SPARQL formalism adds further value for 70B models, which can leverage structural alignment with the entity–relationship context.

Compression and routing provide conditional gains. Graph-walk compression and question-type routing are not universally beneficial but are highly effective under specific conditions: GW helps when the model has structured reasoning guidance (SPARQL + 8B) or sufficient capacity to reason over reduced noise (70B baseline); routing helps when question-type diversity makes a single prompt sub-optimal. Together they enable the fully augmented 8B pipeline to match or exceed the unaugmented 70B baseline on all three benchmarks.

Generality requires structural compatibility. SPARQL CoT is system-agnostic and transfers to LightRAG without modification. Graph-walk compression depends on a retrieval-pipeline relevance gradient and does not transfer to LightRAG. The distinction provides a practical deployment guideline: SPARQL CoT can be applied to any Graph-RAG system whose context preserves entity–relationship structure; GW compression should only be used when the retrieval pipeline creates a clear topological gradient.

Practical deployment guidance. These findings translate into a concrete decision procedure for deploying Graph-RAG with budget models. If the Graph-RAG system returns explicit entity–relationship triples (e.g. KET-RAG, GraphRAG), apply

SPARQL CoT together with graph-walk compression on an 8B model: this matches a 70B unaugmented baseline on all benchmarks at approximately 12× lower per-token inference cost, making it the recommended configuration when cost is a primary constraint. If the system returns aggregated entity descriptions without discrete triples (e.g. LightRAG), apply SPARQL CoT without graph-walk compression, as topology-based pruning will degrade uniformly relevant context. If the evaluation dataset contains mixed question types (bridge, comparison, inference), add question-type routing: it adds only one lightweight classification call per question but closes the remaining accuracy gaps on HotpotQA and MuSiQue and produces the largest single gain on 2WikiMultiHopQA (+16.0 pp over the unaugmented 70B baseline). If budget is maximally constrained and a single intervention must be chosen, SPARQL CoT alone on an 8B model captures the majority of the gain at minimal additional cost.

5.11 Chapter Summary

This chapter presented the empirical results of the thesis. The budget KET-RAG setup achieves strong retrieval coverage (77–91%), confirming that reasoning rather than retrieval is the dominant bottleneck. SPARQL CoT consistently improves accuracy by +2 to +14 pp, with the largest gains on harder benchmarks and chain-following question types. Graph-walk compression adds +6 pp on average when paired with SPARQL CoT for the 8B model, but hurts the 8B baseline (−2.9 pp average) and the 70B SPARQL CoT configuration (−1.1 pp average). The routing + GW pipeline enables the 8B model to match or exceed the unaugmented 70B baseline on all three benchmarks at ∼12× lower cost, with the largest gain on 2WikiMHQA (+16.0 pp accuracy, −27.6 pp abstain). Error decomposition confirms that 73–84% of 8B baseline errors are reasoning failures, and the LightRAG transfer experiment shows that SPARQL CoT is system-agnostic while GW compression requires a retrieval-pipeline relevance gradient to be effective.

The next chapter concludes the thesis by summarizing the main findings, discussing limitations, and outlining directions for future work.

Chapter 6

Conclusion

This thesis studied the retrieval–reasoning gap in Graph-RAG for multi-hop question answering. The central motivation was a simple but important observation: strong retrieval does not necessarily lead to strong answer accuracy. In the KET-RAG setting evaluated here, 77–91% of questions already have the gold answer in the retrieved context, yet, as established in Section 1.3, reasoning failures account for 73–84% of all errors across the three evaluated benchmarks. This indicates that, once retrieval quality becomes sufficiently strong, reasoning over the retrieved context becomes a separate and practically important bottleneck.

To study this bottleneck, the thesis evaluated three inference-time mechanisms applied on top of an existing KET-RAG pipeline: SPARQL-style chain-of-thought prompting, graph-walk context compression, and question-type routing. These methods operate without retraining the retriever or rebuilding the graph index, making it possible to evaluate whether meaningful gains can be achieved by improving reasoning alone and to isolate the contribution of each component through controlled ablations.

6.1 Summary of Contributions

The empirical results support five main contributions.

1. A quantitative error decomposition of Graph-RAG failures. The thesis provides a dataset-level decomposition of errors into retrieval failures (gold answer absent from context) and reasoning failures (gold answer present but model still incorrect). As reported in Section 5.7, in the 8B baseline 84% of HotpotQA errors, 73%

of MuSiQue errors, and 80% of 2WikiMHQA errors are reasoning failures. Even in the strongest KET-RAG configuration (70B + SPARQL CoT), the reasoning share remains at 64–74%, confirming that reasoning is the dominant bottleneck once retrieval coverage is high. This decomposition provides a principled foundation for studying inference-time augmentations and an empirical answer to the question of where Graph-RAG effort is best spent.

2. SPARQL CoT as a system-agnostic reasoning augmentation. SPARQL-style chain-of-thought prompting consistently improves accuracy over the direct baseline across both model sizes and all three benchmarks, with gains of +2 to +14 pp. The largest improvements appear on harder benchmarks and chain-following question types—bridge_comparison questions on 2WikiMHQA gain +18.2 pp (8B) and +28.1 pp (70B)—confirming that triple-pattern decomposition helps most when the question structure aligns with explicit relational transitions. An ablation confirms that decomposition is the primary driver of the benefit, with the SPARQL formalism adding further value for the 70B model through its structural alignment with KET-RAG's entity–relationship context. Crucially, SPARQL CoT also transfers to LightRAG (+2.2 to +7.6 pp), establishing it as a system-agnostic augmentation applicable to any Graph-RAG system whose retrieved context preserves entity–relationship structure.

3. Graph-walk compression as a conditional, low-cost context filter. Graph-walk compression reduces the retrieved context by approximately 60% ($\sim$10,000 $\rightarrow$ $\sim$4,000 tokens) using pure graph traversal with no additional LLM calls. Its value, however, depends on the interaction between model capacity, prompting strategy, and retrieval-pipeline structure. For the 8B model with structured SPARQL CoT guidance, GW adds +6.0 pp on average; for the 8B baseline without structured guidance, it hurts ($-$2.9 pp average) because the model relies on exhaustive context for pattern matching. GW also fails to transfer to LightRAG ($-$6.6 to $-$11.0 pp), not because LightRAG's context is smaller—it is actually larger—but because LightRAG's uniform retrieval creates no relevance gradient for graph-walk BFS to exploit. This finding establishes that compression effectiveness depends on retrieval-pipeline structure rather than context length alone.

4. A budget 8B pipeline that matches the unaugmented 70B baseline. Combining SPARQL CoT, graph-walk compression, and question-type routing enables a budget Llama-3.1-8B model to match or exceed the unaugmented Llama-3.3-70B baseline on all three benchmarks at approximately 12× lower per-token cost. The largest gain is +16.0 pp on 2WikiMHQA, where question-type diversity gives the routing policy the most leverage. Routing reduces abstain rates to the lowest of any configuration (1.0%, 6.2%, 7.2% on the three benchmarks vs. 5.6%, 16.2%, 34.8% for the unaugmented 70B baseline). This result demonstrates that the cost–quality trade-off in Graph-RAG can be meaningfully shifted through inference-time reasoning support, without any change to the retriever or index.

5. A transfer check confirming system-agnostic and system-dependent components. A replication on LightRAG shows that SPARQL CoT transfers across Graph-RAG systems, with gains of +7.6 pp on 2WikiMHQA, +5.4 pp on MuSiQue, and +2.2 pp on HotpotQA for the 8B model. Graph-walk compression, by contrast, hurts on LightRAG across all benchmarks (−6.6 to −11.0 pp), because LightRAG's uniform retrieval creates no relevance gradient for topology-based pruning to exploit. This establishes that structured prompting is system-agnostic while compression effectiveness depends on retrieval-pipeline structure rather than context size alone.

More broadly, the thesis contributes a methodological shift for Graph-RAG evaluation. Rather than measuring only whether better retrieval improves downstream performance, it introduces the coverage–accuracy decomposition as a diagnostic that exposes reasoning quality independently of retrieval quality. This decomposition reveals that Graph-RAG benchmarks can simultaneously show high retrieval coverage and low answer accuracy—and that progress on one does not imply progress on the other. The decomposition is applicable to any Graph-RAG system for which retrieved context can be inspected, and it provides a concrete criterion for deciding when retrieval improvement is still needed versus when reasoning support is the more productive investment.

6.2 Limitations

Several limitations should be noted when interpreting the results.

Benchmark scope. The thesis is centered on three benchmark multi-hop QA datasets. Although HotpotQA, MuSiQue, and 2WikiMHQA are widely studied and cover a range of question types and hop depths, they do not cover all Graph-RAG use cases. The findings are strongest for multi-hop QA settings and should be generalized more cautiously to other applications such as grounded summarization, knowledge-intensive dialogue, or open-ended analytical generation over structured corpora.

Depth of transfer evaluation. The LightRAG analysis serves as a transfer check rather than an equally deep parallel study. Consequently, the thesis provides stronger characterization of the retrieval–reasoning gap within KET-RAG than across Graph-RAG systems in general. The finding that GW compression is ineffective on LightRAG is important, but the thesis does not explore whether a LightRAG-specific compression strategy—one designed for uniform rather than stratified retrieval—could recover the benefit.

Approximate coverage metric. Coverage is defined as the presence of the gold answer string in the retrieved context. This is useful for separating retrieval failure from reasoning failure, but it is necessarily approximate: a question may count as covered even when the relevant evidence is noisy, indirect, or hard to integrate, and conversely may count as uncovered when the answer is paraphrased. The 102-judgment audit found 96.1% agreement between the heuristic–LLM evaluation pipeline and Claude Opus 4.6, but residual evaluation noise remains.

Prompting sensitivity. Prompting is known to be sensitive to formulation, model family, and decoding choices. The performance differences reported here may vary under alternative prompt designs or with model architectures not tested. The thesis identifies a strong and consistent pattern across two model sizes and three benchmarks, but not necessarily the final or optimal prompting formulation for Graph-RAG reasoning.

Zero-shot routing classifier. The question-type router is a zero-shot single-call classifier that assigns each question to one of three types—bridge, comparison, or inference—based only on the question text. Its classification accuracy is not independently evaluated, and systematic misclassification (e.g., bridge questions labelled as inference) would silently degrade routing performance. On datasets with ques-

tion types outside this taxonomy, or with different naming conventions, the policy may generalise poorly without adaptation. A learned classifier trained on labelled question-type annotations would provide stronger guarantees and is a natural direction for future work (Section 6.3).

Inference-time scope. The thesis intentionally studies reasoning support as an inference-time layer above an existing retriever. This is a strength with respect to practical cost-effectiveness and modularity, but it also means the thesis does not explore whether jointly adapting retrieval and reasoning could yield even greater gains—for example, through iterative retrieval conditioned on intermediate reasoning steps.

6.3 Future Work

The results suggest several concrete directions for future work.

Richer reasoning scaffolds. SPARQL-style prompting is effective as an interpretable one-call scaffold, but it is unlikely to be the only useful structure for Graph-RAG reasoning. Future work could explore program-of-thought decomposition [2], multi-step planning over retrieved subgraphs, or graph-neural prompting strategies that make the traversal structure explicit to the model without requiring a formal query language.

Learned and finer-grained routing. The routing policy studied here is a lightweight zero-shot classifier. Future work could investigate learned routing—for example, training a small classifier on question-type labels—finer-grained question categories, or multi-stage decision mechanisms that condition prompt selection on intermediate reasoning states or confidence signals rather than question type alone.

Retrieval–reasoning co-adaptation. This thesis treats reasoning as an inference-time layer that leaves retrieval unchanged. A natural extension is to use reasoning signals to refine retrieval: intermediate variable bindings in a SPARQL trace could identify missing entities and trigger targeted re-retrieval, bridging the gap between single-pass Graph-RAG and iterative interleaved approaches such as IRCoT [22].

LightRAG-compatible compression. The failure of graph-walk BFS compression on LightRAG stems from a structural property of that retrieval pipeline rather than from compression in principle. Future work could design a compression strategy adapted to uniformly relevant retrieval, for example by using semantic similarity ranking or community-structure clustering rather than topological proximity to seed entities.

Model and embedding diversity. This thesis evaluates two model sizes from a single model family (Llama-3) and a single embedding model (MiniLM-L6-v2). A finer-grained sweep across model families (e.g. Mistral, Qwen), model scales, and embedding dimensions would better characterise the degradation curve and determine whether the observed patterns, such as the 8B model's preference for generic CoT over SPARQL CoT, are specific to Llama-3 or reflect a more general relationship between model capacity and structured prompting.

Broader Graph-RAG applications. The retrieval–reasoning gap identified here may also affect Graph-RAG applications beyond benchmark QA, including grounded summarization, knowledge-intensive dialogue, and analytical tasks over structured corpora. Studying whether the same inference-time mechanisms help in these settings would further characterize the generality of the bottleneck and the scope of the proposed solution.

6.4 Concluding Remarks

The main conclusion of this thesis is that strong retrieval alone is not sufficient for strong Graph-RAG question answering. Even when the correct evidence is already present in the retrieved context (as is the case for 77–91% of evaluated questions, established in Section 1.3), the model may still fail if it cannot reason effectively over that evidence. For this reason, improving Graph-RAG requires more than retrieving more or better information. It also requires better support for using the information that has already been retrieved.

This thesis showed that structured prompting, graph-walk compression, and adaptive routing are practical ways to address this problem at inference time, at low additional cost, and without any change to the retriever or index. Together they enable a budget 8B model to match a 12× more expensive 70B baseline, suggesting that the

cost–quality frontier in Graph-RAG can be advanced significantly by investing in the reasoning layer. More broadly, the thesis argues that future progress in Graph-RAG will depend not only on retrieval quality, but also—and perhaps primarily—on the design of reasoning support over retrieved graph-structured evidence.

Bibliography

[1] Sara Asghari, Laks V. S. Lakshmanan, Venkatesh Srinivasan, and Alex Thomo. Fact-checking with large language models via cost-effective first-order logic reformulation. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 66–78, 2025.

[2] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

[3] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[4] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2024.

[5] Yu Gu, Xiang Deng, and Yu Su. Don't generate, discriminate: A proposal for grounding language models to real-world environments. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4928–4949, 2023.

[6] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024.

[7] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[8] Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *International Conference on Machine Learning*, pages 21497–21515, 2025.

[9] Steve Harris and Andy Seaborne. Sparql 1.1 query language. W3C Recommendation, 2013.

[10] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[11] Yiqian Huang, Shiqi Zhang, and Xiaokui Xiao. Ket-rag: A cost-efficient multi-granular indexing framework for graph-rag. *arXiv preprint arXiv:2502.09304*, 2025.

[12] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, 2023.

[13] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*, 2023.

[14] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213, 2022.

[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[16] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[17] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

[18] Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616, 2024.

[19] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR, 2023.

[20] Yaodong Su, Yixiang Fang, Yingli Zhou, Quanqing Xu, and Chuanhui Yang. Clue-RAG: Towards accurate and cost-efficient graph-based RAG via multipartite graph and query-driven iterative retrieval. *arXiv preprint arXiv:2507.08445*, 2025.

[21] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[22] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, 2023.

[23] Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214, 2024.

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting

elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[25] Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690*, 2025.

[26] Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

[27] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[28] Tengkai Yu, Venkatesh Srinivasan, and Alex Thomo. GraphRAG-V: Fast multi-hop retrieval via text-chunk communities. In *International Conference on Advances in Social Networks Analysis and Mining*, pages 3–14, 2025.

[29] Tengkai Yu, Venkatesh Srinivasan, and Alex Thomo. Efficient vector-based Louvain algorithm for massive low-rank graphs. In *Proceedings of the 29th International Conference on Extending Database Technology (EDBT)*, pages 537–543, 2026.

[30] Yasaman Zarrinkia, Venkatesh Srinivasan, and Alex Thomo. The reasoning bottleneck in graph-rag: Structured prompting and context compression for multi-hop qa. *arXiv preprint arXiv:2603.14045*, 2026.