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

Anonymous ACL submission

Abstract

Graph-RAG systems achieve strong multi-hop question answering by indexing documents into knowledge graphs, but strong retrieval does not guarantee strong answers. Evaluating KET-RAG, a leading Graph-RAG system, on three multi-hop QA benchmarks (HotpotQA, MuSiQue, 2WikiMultiHopQA), we find that 77% to 91% of questions have the gold answer in the retrieved context, yet accuracy is only 35% to 78%, and 73% to 84% of errors are reasoning failures. We propose two augmentations: (i) SPARQL chain-of-thought prompting, which decomposes questions into triple-pattern queries aligned with the entity-relationship context, and (ii) graph-walk compression, which compresses the context by ∼60% via knowledge-graph traversal with no LLM calls. SPARQL CoT improves accuracy by +2 to +14 pp; graph-walk compression adds +6 pp on average when paired with structured prompting on smaller models. Surprisingly, we show that, with question-type routing, a fully augmented budget open-weight Llama-8B model matches or exceeds the unaugmented Llama-70B baseline on *all three benchmarks* at ∼12× lower cost. A replication on LightRAG confirms that our augmentations transfer across Graph-RAG systems.

1 Introduction

Multi-hop question answering (QA) requires reasoning over multiple pieces of evidence scattered across documents. Graph-based retrieval-augmented generation (Graph-RAG) addresses this by constructing knowledge graphs from document corpora and using graph structure to retrieve relevant context (Edge et al., 2024; Huang et al., 2025).

The retrieval-reasoning gap. Strong retrieval does not guarantee strong answers. Using KET-RAG (Huang et al., 2025), a state-of-the-art Graph-RAG system, with budget open-weight models (§4), we achieve 77% to 91% context coverage across three benchmarks (Table 2). Yet overall accuracy ranges from only 23% to 67% for Llama-3.1-8B and 35% to 78% for Llama-3.3-70B. The culprit is context volume: KET-RAG's retrieved context typically spans ∼10,000 tokens, containing entity descriptions, relationship descriptions, community report summaries, and raw text chunks. The model must locate the relevant facts and chain them together within this large context, a needle-in-a-haystack challenge that intensifies with the number of reasoning hops required. Decomposing errors into retrieval failures (answer absent from context) and reasoning failures (answer present but model incorrect) reveals that reasoning failures account for 73% to 84% of all errors across datasets. This raises a natural question: can inference-time augmentations to the reasoning stage, requiring no retraining or re-indexing, close this gap?

Reasoning augmentations. To address this bottleneck, we propose two synergistic strategies that require no additional model training. Our augmentations apply to any Graph-RAG system whose retrieved context preserves entity-relationship structure; we focus on KET-RAG and verify that the strategies transfer to LightRAG (Guo et al., 2024), another prominent Graph-RAG system.

1. **SPARQL CoT prompting** (§3.1): A structured chain-of-thought prompt that asks the model to decompose the question as a SPARQL query over the retrieved facts, leveraging the inherent entity-relationship structure of the knowledge-graph context. This is a single LLM call with a modified prompt; the only additional cost is slightly longer output.
2. **Graph-walk context compression** (§3.2): A post-retrieval filtering step that performs breadth-first traversal from question entities through the knowledge graph, retaining only structurally connected context. This reduces input tokens by ∼60% with no LLM calls, directly

reducing the needle-in-a-haystack challenge.

Evaluated on three standard multi-hop QA benchmarks (Yang et al., 2018; Trivedi et al., 2022; Ho et al., 2020) across multiple model sizes and configurations (§4), our contributions are:

1. A quantitative error decomposition showing that reasoning, not retrieval, is the dominant bottleneck: 73% to 84% of errors occur when the gold answer *is* present in the retrieved context.
2. SPARQL CoT consistently improves over the no-CoT baseline (+2 to +14 pp), with the largest gains on harder datasets; an ablation (§5.4) confirms that structured decomposition is the primary driver, with SPARQL particularly effective on 70B.
3. A systematic analysis revealing a synergy between graph-walk compression and structured prompting: compression most helps when the model is following a reasoning roadmap, with smaller models gaining +6 pp average.
4. The finding that combining structured prompting, graph-walk compression, and question-type routing enables a budget 8B model to match or exceed the unaugmented 70B baseline on *all three benchmarks* at $\sim$12$\times$ lower cost.
5. A generality check showing that SPARQL CoT transfers to LightRAG (+7.6 pp on 2WikiMHQA), while graph-walk compression's effectiveness depends on retrieval-pipeline structure rather than context size alone.

2 Background: KET-RAG

KET-RAG (Huang et al., 2025) is a multi-granular indexing framework built on Microsoft's GraphRAG (Edge et al., 2024). It reduces the cost of full-corpus entity extraction by combining three retrieval strategies (**K-E-T**): (K) a text-keyword bipartite graph built from all chunks without LLM calls; (E) LLM-based entity/relationship extraction on a PageRank-selected subset of core chunks (β=0.8), producing a knowledge graph queried via GraphRAG's local search; and (T) standard text-chunk retrieval via embedding similarity. A parameter $\theta \in [0, 1]$ balances the entity and keyword channels within a total budget of λ tokens; we use θ=0.5, λ=12,000 (defaults of the published code).

KET-RAG achieves state-of-the-art results on multi-hop QA, outperforming all competing systems in its evaluation (Lewis et al., 2020; Wang et al., 2024; Edge et al., 2024; Sarmah et al., 2024; Gao et al., 2023; Gutiérrez et al., 2024; Guo et al., 2024). More broadly, KET-RAG is an instance of a growing family of Graph-RAG systems, including LightRAG (Guo et al., 2024) and GraphRAG (Edge et al., 2024), whose retrieved context preserves explicit entity-relationship structure (entity descriptions, relationship triples, community summaries). This structure creates an opportunity: rather than treating the context as flat text, reasoning augmentations can exploit its entity-relationship layout. We validate this generality on LightRAG in Appendix A.1. We replace KET-RAG's commercial components (GPT-4o-mini, OpenAI embeddings) with budget open-weight alternatives without loss in coverage (§5.1) and focus on closing the reasoning gap.

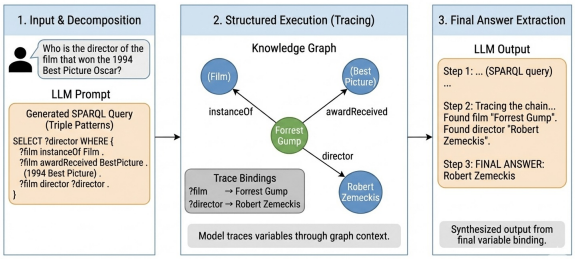

Figure 1: SPARQL CoT prompting. The model generates a SPARQL query whose triple patterns align with the retrieved context, then traces variable bindings to produce the answer in a single LLM call.

3 Methods

3.1 SPARQL Chain-of-Thought Prompting

Multi-hop questions require the model to identify and chain multiple facts scattered across a large retrieved context. Standard prompting leaves the model to perform this chaining implicitly, often causing it to miss intermediate hops or abstain when the reasoning path is unclear.

We observe that KET-RAG's retrieved context is inherently structured: it consists of *entity descriptions*, *relationship descriptions* (subject-predicate-object triples), *community report summaries*, and *text chunks*. This structure closely mirrors a knowledge graph that can be queried with a formal language. We exploit this alignment by introducing a chain-of-thought (CoT) prompting strategy that uses SPARQL (Harris and Seaborne, 2013), the W3C standard query language for RDF knowledge graphs, as a reasoning formalism. SPARQL's triple-pattern syntax (`?subject predicate ?object`) naturally expresses multi-hop entity chains, making it a good fit for decomposing questions over entity-

relationship context. Rather than asking the model to "think step by step" in natural language, we ask it to formulate the question as a structured graph query whose triple patterns map directly onto the entity-relationship pairs in the context (Figure 1).

The prompt template is:

> `Step 1:` Write a simple SPARQL query (max 4 triple patterns, plain English predicates, NO URIs, NO FILTER, NO subqueries). Example:
> `SELECT ?answer WHERE {`
> `?x name "Paradise Creek" .`
> `?x tributaryOf ?y .`
> `?y tributaryOf ?answer .`
> `}`
> `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`

This is a *single* LLM call: the model generates the SPARQL query, traces it through the context, and produces the answer in one response. The key advantage over generic sub-question decomposition is *structural alignment*: the triple patterns (e.g., `?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. The only additional cost is the longer output (the SPARQL query and step-by-step trace, typically 100–200 extra tokens).

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 may respond with "I don't know," which we count as an abstention.

3.2 Graph-Walk Context Compression

While KET-RAG achieves high coverage, much of its ∼10,000-token context is topologically distant from the question's reasoning chain, increasing the needle-in-a-haystack difficulty.

We propose **graph-walk context compression (GW)**, a post-retrieval step that exploits the knowledge graph's own topology to isolate a compact *reasoning subgraph* relevant to the query, with no additional LLM or embedding costs. The algorithm distills the context in four phases:

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 (≥5 characters) appears as part of an entity name; common stop words are excluded.

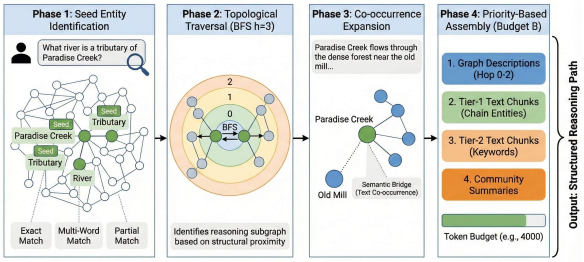

Figure 2: Graph-walk context compression. BFS from question-anchored seed entities identifies structurally connected entities, assembled into compressed context organized by hop level. No LLM calls required.

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.

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.

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 (§3.1): 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.

The entire procedure is pure graph traversal and string matching, requiring no LLM calls or embedding computations. In practice, the BFS reaches 14–21 entities on average, and the priority-based assembly achieves ∼60% compression (∼10,000 → ∼4,000 tokens). Figure 2 illustrates the four phases.

4 Experimental Setup

4.1 Datasets

We use three standard multi-hop QA benchmarks:

- **HotpotQA** (Yang et al., 2018): Bridge and comparison questions over Wikipedia paragraphs. Each question comes with 10 paragraphs (2 gold supporting paragraphs + 8 curated distractors). Bridge questions require following a chain of entities (e.g., "The director of film X was born in what city?"), while comparison questions require attribute comparison (e.g., "Which film was released first, X or Y?").
- **MuSiQue** (Trivedi et al., 2022): Multi-hop questions requiring 2–4 reasoning steps, constructed by composing single-hop questions into chains. The deeper reasoning chains (up to 4 hops) are the primary source of difficulty: each additional hop compounds the chance of a reasoning failure. Each question comes with 20 paragraphs (2–4 gold supporting paragraphs + 16–18 distractors drawn from gold paragraphs of related questions, making them harder to distinguish than random distractors). Generally considered the hardest of the three benchmarks.
- **2WikiMultiHopQA** (Ho et al., 2020): Questions constructed from Wikidata, requiring inference across two Wikipedia articles with the widest range of reasoning types among the three benchmarks (comparison, inference, compositional, and bridge). Each question comes with 10 paragraphs (2 or 4 gold supporting paragraphs + 6–8 distractors), following HotpotQA's format. KET-RAG's own evaluation does not include this dataset; we add it because of its type diversity.

For each dataset, we sample N=500 questions using a fixed seed (42) and pool all per-question paragraphs to construct the retrieval corpus.

4.2 Model Stack

Table 1 shows our model configurations. All LLM calls use the Groq inference platform, which provides fast inference for open-weight models at low per-token cost.

Table 1: Model stack. The index is built once with the 8B model and reused across all QA configurations.

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

The index is model-independent: the knowledge graph and keyword indices do not depend on the QA model, allowing us to build once and vary the QA model without re-indexing. Compared to KET-RAG's default stack (GPT-4o-mini + OpenAI embeddings), ours uses a 3× cheaper LLM and free embeddings with 4× lower dimensionality, for a total indexing cost of ∼\$1 per dataset.

At inference time, the 8B QA model is ∼12× cheaper per token than the 70B (\$0.05 vs. \$0.59/M tokens), motivating the question of whether augmentations can close the accuracy gap.

4.3 Configurations

We evaluate four QA configurations combining two prompting strategies with optional graph-walk compression:

1. **Baseline:** The QA model receives the full KET-RAG-retrieved context (∼10,000 tokens) and answers directly.
2. **Baseline + GW:** Same as Baseline but with graph-walk compressed context.
3. **SPARQL CoT:** The QA model receives the full context with the SPARQL chain-of-thought prompt (§3.1).
4. **SPARQL CoT + GW:** SPARQL CoT prompting applied to graph-walk compressed context.

Each configuration is run with both the 8B and 70B QA models, yielding 8 runs per dataset and 24 runs total across the three benchmarks, for a total of 12,000 question-answer evaluations. All QA calls use temperature 0.3.

4.4 Evaluation

We report accuracy, token-level F1, and exact match (EM) with SQuAD normalization. Accuracy is determined by a heuristic-first pipeline: normalized string matching (lowercasing, stripping articles/punctuation, substring and last-token comparison), with LLM-judged semantic equivalence (Llama-3.1-8B) as fallback for remaining

Table 2: Context coverage (%) with budget components compared to KET-RAG (Huang et al., 2025) 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)	—	—

mismatches (e.g., "NYC" vs. "New York City"). A 102-judgment audit with Claude Opus 4.6 found 96.1% agreement with our pipeline. We also report abstain rate and context coverage (whether the gold answer appears in the retrieved context, separating retrieval from reasoning failures). All comparisons are *paired* on the same 500 questions and retrieved context for each dataset.

5 Results

5.1 Retrieval Quality: Coverage

With our budget components (free 384-dimensional embeddings, $0.05/M-token indexing LLM), KET-RAG matches or exceeds its published coverage across all three benchmarks (Table 2), despite 3× cheaper indexing LLM and 4× lower embedding dimensionality.

5.2 QA Performance

Table 3 presents accuracy, F1, and EM across all 24 configurations.

The headline result is that a budget 8B model, augmented with SPARQL CoT and graph-walk compression, surpasses the unaugmented 70B baseline on 2WikiMHQA (55.8% vs. 48.8%) and closes the gap on MuSiQue and HotpotQA, at ∼12× lower per-token cost. SPARQL CoT alone improves *every* configuration, with gains up to +14.2 pp on 2WikiMHQA for 8B and +12.2 pp for 70B. When both augmentations are applied, 70B + SPARQL reaches the overall best accuracy (61.0% on 2WikiMHQA, 80.2% on HotpotQA). Difficulty tracks reasoning depth as expected: HotpotQA (2-hop, 67-80%), 2WikiMHQA (mixed, 30-61%), MuSiQue (2-4 hop, 19-44%).

The right half of the table conditions on covered questions (gold answer present in context), separating reasoning failures from retrieval gaps; we discuss this in §5.7. We analyze each factor below.

5.3 Effect of SPARQL CoT Prompting

Table 4 isolates the effect of SPARQL CoT by comparing it to the corresponding baseline (without graph-walk compression).

The accuracy gains correlate strongly with abstain rate reduction: on MuSiQue, 8B abstentions drop from 52.0% to 20.6% (-31.4 pp), suggesting that expressing relationships as triple patterns forces the model to *engage* with multi-hop chains rather than declining, converting open-ended reasoning into structured pattern matching.

5.4 CoT Ablation: SPARQL vs. Generic Decomposition

We ablate whether the benefit stems from the SPARQL formalism or decomposition in general, comparing against a generic CoT prompt with natural-language sub-questions on all three datasets (Table 5).

Three patterns hold across all three datasets.

First, both CoT methods improve substantially over the baseline on every dataset, confirming that structured decomposition is the primary driver of the gains.

Second, question-type routing is the best of the four methods on 70B across all three datasets (HotpotQA 82.4, 2WikiMHQA 66.4, MuSiQue 47.0). A per-native-type breakdown (Appendix A.7) shows why: bridge and structural types, which constitute the majority of questions on each benchmark, favour SPARQL on 70B, while comparison and inference questions favour generic CoT on the datasets where they exist. Routing exploits this complementarity.

Third, the SPARQL-vs-generic ranking depends on model size. 70B reliably benefits from SPARQL on all three datasets (80.2 / 61.0 / 43.8 vs. 79.8 / 56.4 / 38.2), while 8B without GW prefers generic CoT on HotpotQA and MuSiQue and is closer on 2WikiMHQA. A hint at the cause comes from a syntax-overhead mechanism: 8B's SPARQL parse rate averages 87.4% vs. 93.4% for 70B, with the largest per-dataset gap on HotpotQA (10.8 pp), exactly where 8B's SPARQL/generic accuracy gap is also largest (Appendix A.6).

On 8B without GW, routing wins on 2WikiMHQA, ties generic on HotpotQA, and trails on MuSiQue. With GW added (Table 6), routing wins on all datasets, with the largest gain of +16.0 pp on 2WikiMHQA.

Notably, both CoT methods improve accuracy for the 8B model on every dataset, contrasting with findings that CoT primarily helps larger models (Wei et al., 2022).

Table 3: Results (%) on 500 paired questions per dataset. "Covered" = gold answer present in context (count in parentheses). **Bold** = best per column. GW = graph-walk compression. F1/EM use SQuAD normalization; 8B outputs are LLM-normalized to short answers (§4).

		All Questions									Covered Questions								
		Hotpot			MuSiQue			2Wiki			Hotpot (454)			MuSiQue (386)			2Wiki (405)		
Method	**Model**	Acc	F1	EM	Acc	F1	EM	Acc	F1	EM	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	69.6	62.2	51.5	27.5	21.2	15.8	32.1	30.1	24.7
+GW	8B	63.6	55.9	46.0	19.4	14.6	9.0	30.4	28.1	24.8	66.3	58.3	48.0	22.6	17.0	11.4	33.1	30.8	26.9
SPARQL	8B	70.6	60.8	49.4	28.8	22.6	15.4	45.6	36.9	31.0	74.4	64.2	52.6	34.8	**27.4**	19.7	49.9	40.6	33.6
+GW	8B	**76.6**	**62.2**	**50.6**	**30.6**	**22.9**	**15.8**	**55.8**	**41.1**	**34.8**	**79.5**	**64.9**	**52.9**	**35.1**	26.4	**20.0**	**59.5**	**44.5**	**37.5**
Base	70B	78.0	67.9	56.6	35.2	26.9	18.6	48.8	43.0	36.8	81.3	70.9	59.0	40.3	32.5	23.9	54.8	47.9	40.7
+GW	70B	79.2	68.6	56.0	39.8	30.8	20.6	53.6	49.0	44.0	83.3	71.9	58.6	46.2	37.3	26.2	61.0	56.1	50.4
SPARQL	70B	**80.2**	**70.6**	**57.8**	**43.8**	32.8	23.4	**61.0**	**54.9**	**48.6**	**83.9**	**74.3**	**61.2**	**53.2**	40.4	29.6	**66.9**	**60.0**	**52.6**
+GW	70B	79.6	70.5	**57.8**	42.2	**34.0**	**25.8**	59.8	51.5	44.0	83.5	74.0	60.8	51.4	**42.3**	**32.5**	66.7	57.6	49.6

Table 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

Table 5: CoT ablation across all three datasets (without GW). Routing applies the 3-way classifier (A.2).

Method	**8B Acc**	**8B Abs**	**70B Acc**	**70B Abs**
HotpotQA				
Baseline	67.0	12.2	78.0	7.8
Generic CoT	**75.6**	7.8	79.8	11.2
SPARQL CoT	70.6	5.8	80.2	8.2
Routing	75.0	**1.2**	**82.4**	**6.6**
2WikiMHQA				
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**
MuSiQue				
Baseline	23.6	52.0	35.2	42.4
Generic CoT	**37.0**	29.2	38.2	41.4
SPARQL CoT	28.8	20.6	43.8	34.2
Routing	33.8	**10.2**	**47.0**	**28.6**

Comparison with Self-Ask. Among three established multi-hop CoT methods (IRCoT (Trivedi et al., 2023), Self-Ask (Press et al., 2023), Decomposed Prompting (Khot et al., 2023)), only Self-Ask has a published single-call form; single-call adaptations of the other two collapse to our generic CoT and to Self-Ask respectively, since their distinguishing components (interleaved retrieval; subtask delegation) are multi-call by construction. Under apples-to-apples prompting (Press et al.'s Bamboogle precedent of non-test-tuned prompts on both methods), SPARQL CoT wins 5 of 6 dataset-model cells by +6.7 pp on average for those cells; the single loss is −3.0 pp on MuSiQue 70B (Appendix A.4).

Table 6: 8B with routing + GW vs. unaugmented 70B baseline on KET-RAG. Routing classifies each question, applies the preferred CoT, and falls back on abstain.

	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**

5.5 Question-Type Routing

Because 8B's preferred CoT variant depends on question type (§5.4), we deploy the routing policy across all three datasets with graph-walk compression (Table 6).

With routing and graph-walk compression, the 8B model matches or exceeds the unaugmented 70B baseline on all three benchmarks at ∼12× lower cost. The gains are largest on 2WikiMHQA (+16.0 pp), where question-type diversity (bridge, comparison, inference, compositional) gives routing the most leverage. Without routing, SPARQL+GW alone surpasses 70B only on 2WikiMHQA (+7.0 pp) while trailing on HotpotQA (-1.4) and MuSiQue (-4.6); routing closes these remaining gaps. The routing policy adds only a cheap classifier call (a few tokens) and at most one retry per question, keeping total inference cost well below a single 70B call.

Implementation. The router is a single 8B call (max 5 output tokens) that classifies each question as *bridge*, *comparison*, or *inference* (exact prompt in Appendix A.2). Bridge questions go to SPARQL CoT; comparison and inference questions go to

Table 7: Effect of graph-walk compression: accuracy Δ (pp) vs. non-GW config.

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

Table 8: Reasoning share of total errors (%): incorrect answers where the gold answer *was* present in context.

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

generic CoT. If the chosen method abstains, the question is retried with the other method. The full pipeline thus requires at most three 8B calls per question: classify, answer, and optionally retry.

5.6 Effect of Graph-Walk Compression

Graph-walk compression shows a more complex interaction with model capability and prompting strategy (Table 7).

A consistent pattern emerges: GW helps most for SPARQL + 8B (+6.0 pp avg, +10.2 on 2WikiMHQA) and Base + 70B (+3.5 pp avg), but hurts Base + 8B (-2.9 pp). For SPARQL + 70B, GW incurs only a -1.1 pp average loss while reducing input tokens by ∼60% (∼10,000 → ∼4,000), a favorable accuracy-cost trade-off given the 70B model's higher per-token price. This reveals a complementarity: GW helps when the model has structured guidance (SPARQL + 8B) or sufficient capacity to reason over reduced noise (70B baseline); without either, the small model relies on exhaustive context for pattern matching and compression removes useful signal.

Against truncation and top-k embedding-similarity baselines at the same 4k-token budget on 8B, GW outperforms both on SPARQL CoT across all three datasets (+9.2 / +8.8 / +3.2 pp over top-k), supporting GW's structural alignment with triple-pattern reasoning (Appendix A.9).

5.7 Error Decomposition

To disentangle retrieval and reasoning failures, we decompose errors by whether the gold answer was present in the retrieved context. Table 8 shows the reasoning share of total errors. The "Covered" columns of Table 3 report accuracy, F1, and EM conditioned on coverage across all configurations.

Reasoning failures dominate: on MuSiQue for example, the answer is present in context for 77.2% of questions (Table 2), yet the 8B baseline answers only 23.6% correctly overall, a gap of over 50 points attributable to reasoning failures. SPARQL CoT with 70B closes roughly half this gap (53.2% on covered questions), confirming that the dominant bottleneck in Graph-RAG QA is reasoning, not retrieval.

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 mostly retrieval gaps. For error decomposition specifically, a stricter bridging-entity coverage metric, requiring all entities needed to derive the answer in the retrieved context (Appendix A.5), gives 88.2 / 66.0 / 69.4% coverage and confirms reasoning failures remain the dominant source in 5 of 6 dataset-model cells.

5.8 Analysis by Question Type

A per-type breakdown (Appendix A.8) confirms that SPARQL CoT's gains concentrate on chain-following types (bridge_comparison sees +18.2 / +28.1 pp on 2WikiMultiHopQA for 8B/70B) and are smallest on inference (+3.7 pp), consistent with the SPARQL formalism's structural match. GW compression similarly benefits the deepest hops (+10 to +12 pp on MuSiQue 3-hop and 4-hop) and inference questions (+16.7 pp on 2WikiMultiHopQA), where focused context most helps integration across entities.

6 Discussion

Reasoning, not retrieval, is the bottleneck. Interestingly, KET-RAG achieves 77% to 91% coverage even with budget components (§5.1), yet accuracy lags far behind: the coverage-to-accuracy gap exceeds 50 percentage points on MuSiQue (§5.7). As the graph index is model-independent, we can build once with inexpensive models and upgrade the QA model without re-indexing.

Structured decomposition is the primary lever. Our key findings show that SPARQL CoT improves every configuration we tested (§3.1), including 8B models, contrasting with findings that chain-of-thought primarily benefits models above ∼100B parameters (Wei et al., 2022). The ablation (§5.4) shows the benefit comes from decomposition itself; the SPARQL formalism adds further value for 70B.

How far can a small model go? 8B + SPARQL CoT + GW surpasses the unaugmented 70B baseline on 2WikiMHQA (55.8% vs. 48.8%) at $\sim$12$\times$ lower per-token cost. On MuSiQue, however, 70B retains a +4.6 pp advantage, suggesting that deeper reasoning chains (3–4 hops) require larger models, while 2-hop questions can be scaffolded effectively even with small models. Question-type routing (§5.5) pushes this further: by steering each question to the appropriate CoT variant and retrying on abstention, the fully augmented 8B surpasses unaugmented 70B on all three benchmarks.

Generality beyond KET-RAG. To test whether these results are system-specific, we replicate the experiment on LightRAG (Guo et al., 2024) across all three benchmarks (Appendix A.1). SPARQL CoT consistently improves LightRAG: +7.6 pp on 2WikiMHQA, +5.4 pp on MuSiQue, and +2.2 pp on HotpotQA, confirming that structured reasoning transfers whenever the context contains entity-relationship structure. Graph-walk compression, by contrast, hurts on LightRAG, consistent with its hybrid retrieval lacking the β-stratified relevance gradient that GW traversal exploits in KET-RAG. This suggests SPARQL CoT is a Graph-RAG-family-agnostic prompting strategy whenever the retrieved context exposes triple-structured entity-relationship content, while GW compression requires a retrieval-pipeline relevance gradient to be effective. Retrievers that return flat passages or summarized community reports (e.g., HippoRAG, KGP) fall outside this structural condition and would require a different formalism.

Deployment recommendations. The results suggest a simple decision rule. For budget deployments (8B), always enable routing with graph-walk compression: it matches or exceeds 70B at $\sim$12$\times$ lower cost, with negligible overhead (one classifier call plus at most one fallback). For quality-maximizing deployments (70B), use SPARQL CoT without compression; GW's $\sim$1 pp accuracy cost rarely justifies the token savings at this scale. GW should be avoided only for Base + 8B, where the small model lacks reasoning guidance to navigate compressed context.

7 Related Work

Graph-RAG systems. A key distinction across Graph-RAG systems is whether the retrieved context preserves entity-relationship (E-R) structure. GraphRAG (Edge et al., 2024), KET-RAG (Huang et al., 2025), and LightRAG (Guo et al., 2024) return entity descriptions, relationship triples, and text chunks, making their context amenable to structured prompting and graph-based compression. HybridRAG (Sarmah et al., 2024) also combines KG subgraphs with vector-retrieved passages, though it has only been evaluated on financial transcripts. Other systems use graph structure for retrieval but return flat passages: HippoRAG (Gutiérrez et al., 2024, 2025), KGP (Wang et al., 2024), Clue-RAG (Su et al., 2025), and GraphRAG-V (Yu et al., 2025). Xiang et al. (2025) identify context inflation as a key failure mode. All the above target retrieval quality; we study reasoning over the retrieved E-R context with budget models.

Chain-of-thought prompting. IRCoT (Trivedi et al., 2023) and Decomposed Prompting (Khot et al., 2023) are multi-call by construction; their single-call adaptations collapse to our generic CoT and to Self-Ask (Press et al., 2023) respectively, which we compare against directly (§5.4, Appendix A.4). SPARQL has been used for knowledge base QA (Gu et al., 2023) but never as a CoT scaffold for Graph-RAG.

Context compression. LLMLingua (Jiang et al., 2023) and RECOMP (Xu et al., 2023) compress passages via learned models, incurring inference cost. Graph-RAG context has exploitable topology that enables LLM-free compression via BFS. The interaction between compression and structured prompting has not been previously studied.

8 Conclusion

Once retrieval achieves high coverage, the dominant bottleneck in Graph-RAG QA is reasoning. We propose three inference-time augmentations that target this gap: SPARQL CoT (+2 to +14 pp), graph-walk compression, and question-type routing; none requires retraining or re-indexing. Together, they enable a fully augmented 8B model to match or exceed an unaugmented 70B baseline on all three benchmarks at $\sim$12$\times$ lower cost. Because the graph index is model-independent, stronger future models can be swapped in without re-indexing. For budget deployments, routing with graph-walk compression matches or exceeds 70B; for quality-maximizing deployments on 70B, SPARQL CoT alone suffices. Code (MIT): `https://anonymous.4open.science/r/graph-rag-qa-pub-CB00`

Limitations

We evaluate with two Llama-3 sizes and a single embedding model; cross-architecture experiments (e.g., Mistral, Qwen) would better characterize generality.

In SPARQL CoT, the model writes the full reasoning chain in one LLM call. If an early step fails (e.g., the first variable cannot be bound to any entity in the context), the rest of the chain inherits that failure; we detect this via the abstain mechanism, but do not retry. Multi-call recovery falls outside the paper's single-call cost-efficient scope and is future work.

Our SPARQL parse-rate analysis (Appendix A.6) measures structural-syntax validity only; we have no direct predicate-grounding metric.

We use the default $\theta = 0.5$ for KET-RAG's entity-keyword balance, where per-dataset tuning might improve results. A β=0 ablation of retrieval stratification and a soft NL-infused structured-prompt variant are natural next steps.

We used an AI coding assistant (Claude Code) to help with experiment scripting and paper editing; all scientific contributions were made by the authors.

References

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*.

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

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Yu Gu, Xiang Deng, and Yu Su. 2023. 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.

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

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.

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

Steve Harris and Andy Seaborne. 2013. SPARQL 1.1 query language. W3C Recommendation.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. 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.

Yiqian Huang, Shiqi Zhang, and Xiaokui Xiao. 2025. Ket-rag: A cost-efficient multi-granular indexing framework for graph-rag. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 1003–1012.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. 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.

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

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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

Bhaskarjit Sarmah, Dhagash Mehta, Benika Hall, Rohan Rao, Sunil Patel, and Stefano Pasquali. 2024. 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.

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

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. 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.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. 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.

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

A Appendix

This appendix provides supplementary analysis and experimental details referenced in the main paper. §A.1 presents the full LightRAG generality experiment summarized in §6. §A.2 gives the routing classifier prompt used in §5.5. §A.3 gives the generic CoT prompt used as a baseline throughout the paper. §A.4 reports the full Self-Ask comparison summarized in §5.4, with adaptation notes on IRCoT and Decomposed Prompting. §A.5 presents the bridging-entity coverage setup and per-cell results referenced in §5.7. §A.6 reports per-cell SPARQL parse rates supporting the syntax-overhead discussion in §5.4. §A.7 gives the per-native-type SPARQL CoT vs generic CoT breakdown referenced in §5.4. §A.8 presents the per-question-type SPARQL CoT and GW gain tables referenced in §5.8. §A.9 compares graph-walk compression against simpler alternatives (truncation, top-k embedding similarity) at the same budget.

A.1 Generality: LightRAG Experiment

To test whether SPARQL CoT generalises beyond KET-RAG, we replicate the full experiment on LightRAG (Guo et al., 2024), a prominent open-source Graph-RAG system, across all three benchmarks. We use the same models as our KET-RAG setup (Llama-3.1-8B for indexing, both 8B and 70B for QA) and retrieve hybrid (keyword + vector) context for each of the 500 questions, then run all eight QA configurations (4 methods $\times$ 2 models). LightRAG context coverage is 450/500 on HotpotQA, 374/500 on MuSiQue, and 417/500 on 2WikiMHQA.

Table 9 mirrors Table 3 for direct comparison. SPARQL CoT consistently improves all configurations, with larger gains for 8B, confirming that structured decomposition disproportionately benefits smaller models. The overall gains are smaller than on KET-RAG, likely because LightRAG aggregates entities into high-level descriptions during extraction, making triple-pattern decomposition a less precise match for the retrieved context.

Graph-walk compression, by contrast, *hurts* on LightRAG across all configurations, the opposite of KET-RAG. Because the two systems retrieve similar-sized contexts (median $\sim$13,000 vs. $\sim$10,600 tokens), the difference is not one of context length but of context *structure*: KET-RAG's retrieval separates core chunks (β=0.8) from pe-

Table 9: LightRAG generality experiment (%) on 500 paired questions per dataset. "Covered" = gold answer present in LightRAG context (count in parentheses). **Bold** = best per model group. GW = graph-walk compression. F1/EM use SQuAD normalization; 8B outputs are LLM-normalized to short answers (§4).

		All Questions									Covered Questions								
		Hotpot			MuSiQue			2Wiki			Hotpot (450)			MuSiQue (374)			2Wiki (417)		
Method	**Model**	Acc	F1	EM	Acc	F1	EM	Acc	F1	EM	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	65.8	57.8	47.3	34.2	23.7	16.6	34.5	31.2	26.6
+GW	8B	54.4	47.8	38.4	25.4	18.1	11.8	35.2	31.9	27.6	56.0	49.1	39.1	30.7	22.0	15.2	35.7	32.2	27.6
SPARQL	8B	**66.8**	**57.5**	**46.6**	**37.4**	**26.9**	**18.6**	**57.4**	44.3	36.2	**69.6**	**59.3**	**47.8**	**46.3**	**33.0**	**24.1**	**61.9**	**47.1**	38.8
+GW	8B	66.2	55.2	44.4	33.6	23.5	15.8	56.4	**44.6**	**37.8**	69.3	57.0	45.6	40.1	28.1	20.3	60.2	46.6	**39.8**
Base	70B	79.8	68.1	55.6	41.6	32.0	24.2	59.8	49.1	41.8	83.3	70.7	57.8	52.1	40.2	31.8	64.5	52.1	44.8
+GW	70B	68.8	59.9	49.4	35.0	25.7	17.2	51.2	45.8	39.2	71.8	62.2	51.1	44.4	32.5	23.0	54.9	49.0	42.0
SPARQL	70B	**82.0**	**71.6**	**57.8**	**47.0**	**36.5**	**28.0**	**67.4**	**58.1**	**49.0**	**85.8**	**74.2**	**59.8**	**58.8**	**46.6**	**36.9**	**72.4**	**62.7**	**52.8**
+GW	70B	64.8	56.5	45.4	36.6	29.7	23.6	50.2	45.0	37.6	67.3	58.2	46.4	47.1	38.4	31.3	53.7	47.8	40.0

ripheral ones, creating a relevance gradient that graph-walk traversal can exploit. LightRAG's hybrid retrieval, by contrast, returns material without such stratification, so pruning removes relevant content as readily as noise.

These results support our central claim that SPARQL CoT is agnostic across Graph-RAG systems that expose triple-structured entity-relationship content: it improves multi-hop QA on both KET-RAG and LightRAG, regardless of which produced the context. The GW finding adds a practical caveat: compression is most beneficial when the retrieval pipeline introduces a clear relevance gradient that graph-walk traversal can exploit, rather than being determined by context size alone.

A.2 Routing Classifier Prompt

The question-type classifier (§5.5) uses the following prompt with the same 8B model (max 5 output tokens):

```
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 or
comparison or inference

Question: {question}
```

If the model returns a label outside {bridge, comparison, inference}, the system defaults to bridge.

A.3 Generic CoT Prompt

The generic CoT baseline (§5.4) uses the following single-call decomposition prompt:

```
You are answering a multi-hop question
using ONLY the provided context.
Step 1: Break down the question into simpler
sub-questions that need to be answered one at a
time.
Step 2: For each sub-question, find the relevant
information in the context and write the interme-
diate answer.
Step 3: Combine the intermediate answers to
reach the final answer.
Step 4: Write FINAL ANSWER: <your answer
in a few words>
If the answer is not in the context, write FINAL
ANSWER: I don't know
CONTEXT: {context}
QUESTION: {question}
```

A.4 Self-Ask Comparison: Setup and Full Results

We compare SPARQL CoT against Self-Ask (Press et al., 2023) using the verbatim few-shot prompts from Press et al.'s Tables 13 (4-shot) and 10 (6-shot).

Adaptation for retrieval. Press et al. evaluate Self-Ask with parametric knowledge or an external search engine. Our setting is retrieval-grounded, so we insert the retrieved text as a labelled `Context:` block between the few-shot demonstrations and the test question. This is the only adaptation. Following Press et al.'s footnote 4, we append `Are follow up questions needed here: Yes.` to the prompt for smaller-than-Davinci models (both Llama-3.1-8B and Llama-3.3-70B qualify).

Decoding. Temperature 0.3, max_tokens 2048. Both methods receive the same retrieved context, without GW compression.

Symmetric prompts (no test-set tuning). For an apples-to-apples comparison, neither method's prompt is tuned to the test dataset. Self-Ask uses Press et al.'s 4-shot prompt across all three test

Table 10: Self-Ask vs. SPARQL CoT, full accuracies (%) on the same 500 questions per dataset, both methods using non-test-tuned prompts.

Dataset	Model	S-Ask	SPARQL	Gap (pp)
HotpotQA	8B	57.0	**70.6**	+13.6
HotpotQA	70B	70.0	**80.2**	+10.2
2WikiMHQA	8B	42.2	**45.6**	+3.4
2WikiMHQA	70B	58.6	**61.0**	+2.4
MuSiQue	8B	25.0	**28.8**	+3.8
MuSiQue	70B	**46.8**	43.8	−3.0

Table 11: Reasoning share of total errors (%) under the bridging-entity coverage metric, per dataset-model cell. Values >50% indicate reasoning is the dominant error source.

Dataset	Model	Reasoning share (%)
HotpotQA	8B	78.8
HotpotQA	70B	70.9
2WikiMultiHopQA	8B	58.0
2WikiMultiHopQA	70B	46.5
MuSiQue	8B	63.6
MuSiQue	70B	60.2

datasets, following Press et al.'s own Bamboogle precedent of prompt transfer. SPARQL CoT uses its generic single-example prompt uniformly.

Adaptation notes (IRCoT, Decomposed Prompting). Of three established multi-hop methods (IRCoT (Trivedi et al., 2023), Self-Ask, Decomposed Prompting (Khot et al., 2023)), only Self-Ask has a published single-call form. The other two are defined by their multi-call interactions: IRCoT by interleaved retrieval, Decomposed Prompting by sub-task handler delegation. A single-call adaptation of either strips out the contribution that distinguishes it from chain-of-thought: single-call IRCoT collapses to chain-of-thought over the upfront context (which we test via generic CoT), and single-call Decomposed Prompting collapses to Self-Ask's decompose-and-answer pattern (which we test directly). We therefore selected Self-Ask as the established single-call baseline.

A.5 Bridging-Entity Coverage: Setup and Results

The substring coverage metric used in Tables 2 and 8 counts a question as covered if the gold answer string appears in the retrieved context. It measures retrieval quality directly and matches the metric used by KET-RAG and prior RAG work. It can, however, both overcount (the gold string appearing in a distractor) and undercount (the answer paraphrased).

For error decomposition specifically, we additionally compute a stricter *bridging-entity* coverage metric that requires all entities needed to derive the answer to appear in the retrieved context (case-insensitive substring). Each benchmark publishes its supporting-fact annotations in a different format, so the bridging-entity set is constructed using each benchmark's native annotation:

- **HotpotQA:** the article titles of the two supporting paragraphs (the bridging entities in HotpotQA's annotation convention), plus the final answer.
- **2WikiMultiHopQA:** the subjects and objects of all evidence triples (i.e., the entities along the explicit reasoning chain).
- **MuSiQue:** all intermediate decomposition-step answers (the bridging values produced en route to the final answer).

The average bridging-entity set size is 3.0 on HotpotQA, 4.0 on 2WikiMultiHopQA, and 2.7 on MuSiQue. Coverage under the bridging-entity metric is 88.2% on HotpotQA, 66.0% on 2WikiMultiHopQA, and 69.4% on MuSiQue. The steeper drop on 2WikiMultiHopQA (from 81.0% under substring matching) reflects its larger average set size, since covering 4 entities per question is strictly harder than covering 3 or 2.7.

Reasoning failures remain the dominant source of errors in 5 of 6 dataset-model cells. The only flip is 2WikiMultiHopQA 70B, where retrieval narrowly dominates (53.5% retrieval vs. 46.5% reasoning). The direction of the central claim is robust to the choice of metric.

A.6 SPARQL Syntax-Validity Rates

To quantify the syntax-overhead mechanism noted in §5.4, we measure the parse rate of the SPARQL block produced by each SPARQL CoT response (Table 12). From each model output we extract the first `SELECT ... { ... }` block (brace-balanced) and check whether it parses under rdflib's SPARQL grammar. Because the model uses bare predicate names like `tributaryOf` (we deliberately give it no schema), we first wrap bare names in IRI brackets `<...>` before parsing. The metric therefore measures *structural* grammar validity (well-formed SELECT, WHERE, and triple-pattern syntax), not predicate vocabulary correctness.

Both models produce mostly valid SPARQL (averages 87.4% on 8B, 93.4% on 70B), but 8B is consistently more malformed, and the gap is largest on HotpotQA (10.8 pp), where 8B's SPARQL CoT vs.

Table 12: SPARQL parse rate (%) per dataset and model. Both models attempt a SPARQL query on essentially every question (extraction rate 96–99%); the rate is computed over all 500 questions.

Dataset	8B	70B
HotpotQA	81.8	92.6
2WikiMultiHopQA	91.4	95.4
MuSiQue	89.0	92.2
Average	87.4	93.4

generic CoT accuracy deficit is also largest (§5.4). This supports the hypothesised syntax-overhead mechanism but does not establish it as the sole cause; alternative explanations (longer reasoning traces in generic CoT contributing via test-time compute, better alignment of natural-language CoT with smaller models' post-training distributions) remain plausible.

A.7 Per-Native-Type SPARQL CoT vs Generic CoT

This appendix gives the per-native-type breakdown of SPARQL CoT vs generic CoT (without GW) that underpins the routing rule (§5.5), shown in Table 13. The two methods are complementary rather than competing: SPARQL is strongest on structural types (where triple-pattern decomposition aligns with the entity-relationship context), generic CoT is strongest on comparison and inference (where the question logic is not a clean entity chain). Routing dispatches by question type to exploit this complementarity.

The complementarity manifests in two ways. On 70B, SPARQL CoT wins on structural types (HotpotQA bridge, 2WikiMultiHopQA compositional and bridge_comparison with its +19.8 pp gain, and every MuSiQue hop-count); generic CoT wins on explicit comparison and inference. On 8B without GW, the SPARQL syntax overhead dominates and generic CoT wins most cells (a deficit GW reverses on bridge-like questions, Table 6). The routing rule (bridge → SPARQL CoT, comparison and inference → generic CoT) maps these per-type winners directly, which is why routing is the best of the four methods on every dataset at 70B (Table 5) and on 2WikiMultiHopQA at 8B.

A.8 Per-Question-Type Gains

This appendix gives the full per-question-type breakdown of SPARQL CoT gains over baseline (no CoT, no GW) and of GW compression effects, summarised in §5.8.

Table 13: Per-native-type accuracy (%) of SPARQL CoT (S) and generic CoT (G), no GW, on all three datasets. $\Delta = \mathrm{S} - \mathrm{G}$. `br_comp` = 2WikiMultiHopQA's `bridge_comparison` type.

		8B			70B		
Type	**n**	**S**	**G**	**Δ**	**S**	**G**	**Δ**
HotpotQA							
bridge	398	71.1	74.9	−3.8	80.2	78.4	+1.8
comparison	102	68.6	78.4	−9.8	80.4	85.3	−4.9
2WikiMultiHopQA							
compositional	205	41.0	35.6	+5.4	48.3	46.3	+2.0
br_comp	121	37.2	46.3	−9.1	53.7	33.9	+19.8
comparison	120	66.7	84.2	−17.5	91.7	94.2	−2.5
inference	54	35.2	61.1	−25.9	57.4	61.1	−3.7
MuSiQue (hop-count labels follow native naming)							
2hop	264	37.5	40.2	−2.7	49.2	41.3	+8.0
3hop1	110	19.1	40.0	−20.9	40.9	40.0	+0.9
4hop1	60	20.0	40.0	−20.0	41.7	40.0	+1.7
3hop2	36	22.2	25.0	−2.8	38.9	27.8	+11.1
4hop3	17	11.8	5.9	+5.9	11.8	11.8	0.0
4hop2	13	15.4	7.7	+7.7	23.1	15.4	+7.7

The largest gains appear on chain-following types: bridge_comparison questions see +18.2 pp (8B) and +28.1 pp (70B) on 2WikiMHQA, while inference questions that require implicit reasoning show the smallest gains (+3.7 pp). This aligns with the design of SPARQL CoT: triple-pattern decomposition naturally formalizes entity chains, making it most effective when the question structure matches the formalism. On MuSiQue, the 70B model shows substantial SPARQL CoT benefit at all hop depths (+9.4 at 2-hop, +5.5 at 3-hop, +11.1 at 4-hop), with the largest gain at 4-hop.

GW compression similarly benefits deeper hops: on MuSiQue, GW is slightly negative at 2-hop but strongly positive at 3-hop (+11.7 for Base+70B) and 4-hop (+10.0). On 2WikiMHQA, GW helps most on inference (+16.7 pp for SPARQL+8B) and bridge_comparison questions (+17.4 pp for Base+70B), where integrating information across multiple entities benefits most from focused context.

A.9 Compression Baselines

To assess GW's specific contribution against simpler compression strategies at the same 4k-token budget, we compare it against two baselines on Llama-3.1-8B (where GW's headline effect is most pronounced, Table 7). Results are reported in Table 16.

Baselines.

Table 14: SPARQL CoT accuracy improvement (pp) over baseline, by question type. HotpotQA: bridge (398) and 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

Table 15: 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

Table 16: Compression-baseline accuracy (%) on 8B at the same 4k-token budget. `no-comp` = full retrieved context ($\sim$10k tokens).

Dataset	no-comp	+GW	+trunc	+top-k
Baseline (no CoT)				
HotpotQA	67.0	63.6	34.6	65.0
2WikiMHQA	31.4	30.4	18.6	30.8
MuSiQue	23.6	19.4	10.2	23.6
SPARQL CoT				
HotpotQA	70.6	**76.6**	42.4	67.4
2WikiMHQA	45.6	**55.8**	32.2	47.0
MuSiQue	28.8	**30.6**	17.2	27.4

- **Truncation**: take the first $\sim$16k characters of the raw context, cut at the nearest paragraph break in the last 20% of the budget.
- **Top-k embedding similarity**: split the context at KET-RAG's natural markers (per-entity, per-relationship, per-text-chunk), embed each chunk with `all-MiniLM-L6-v2`, embed the question, greedy-fill to budget by cosine similarity, restore original order in the assembled output.

Three findings emerge. First, truncation collapses uniformly (-14 to -27 pp vs no-compression on every cell), confirming that compression strategy matters: simply chopping the context is not a valid alternative. Second, top-k embedding similarity matches or slightly beats GW on the no-CoT baseline (HotpotQA $+1.4$, 2WikiMHQA $+0.4$, MuSiQue $+4.2$), but loses to GW on SPARQL CoT across all three datasets (-9.2, -8.8, -3.2). Third, GW's specific contribution is its interaction with SPARQL CoT: the +GW minus top-k gap is largest where structured reasoning has the most leverage (HotpotQA $+9.2$ pp, 2WikiMHQA $+8.8$ pp), consistent with the paper's structural-alignment thesis. GW preserves the entity-relationship chains that triple-pattern reasoning navigates; embedding similarity preserves text similar to the question, not necessarily the bridging structure. We report 8B only since GW's headline effect is on 8B; the 70B comparison is a natural extension.