RAGRouter-Bench: A Dataset and Benchmark for Adaptive RAG Routing

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) has become a core paradigm for grounding large language models with external knowledge. Despite extensive efforts exploring diverse retrieval strategies, existing studies predominantly focus on query-side complexity or isolated method improvements, lacking a systematic understanding of how RAG paradigms behave across different query–corpus contexts and effectiveness–efficiency trade-offs. In this work, we introduce RAGRouter-Bench, the first dataset and benchmark designed for adaptive RAG routing. RAGRouter-Bench revisits retrieval from a query–corpus compatibility perspective and standardizes five representative RAG paradigms for systematic evaluation across 7,727 queries and 21,460 documents spanning diverse domains. The benchmark incorporates three canonical query types together with fine-grained semantic and structural corpus metrics, as well as a unified evaluation for both generation quality and resource consumption. Experiments with DeepSeek-V3 and LLaMA-3.1-8B demonstrate that no single RAG paradigm is universally optimal, that paradigm applicability is strongly shaped by query–corpus interactions, and that increased advanced mechanism does not necessarily yield better effectiveness–efficiency trade-offs. These findings underscore the necessity of routing-aware evaluation and establish a foundation for adaptive, interpretable, and generalizable next-generation RAG systems. The code and dataset are available at: https://anonymous.4open.science/r/RAGRouter-Bench/.

1 Introduction

LLMs are prone to hallucinations when confronted with specialized domains, evolving facts, or long-tail information needs (Ji et al., 2023; Mallen et al., 2023). These challenges have motivated the emergence of Retrieval-Augmented Generation (RAG),

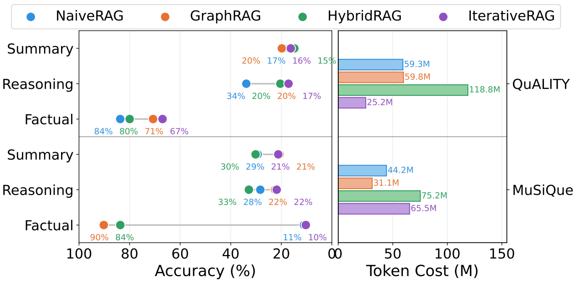

Figure 1: Preliminary Study on Paradigm Conflict. Left: Accuracy of four RAG paradigms across two datasets and three query types. **Right:** Token consumption per paradigm on each dataset.

which has come to underpin modern question answering, creative generation, document summarization, and multi-hop reasoning (Gao et al., 2023). RAG typically follows a two-stage pipeline. It identifies query-relevant evidence from external corpora, and combined with the query to feed into an LLM-based generator, yielding factual and faithful responses (Lewis et al., 2020; Guu et al., 2020). In practice, retrieval constitutes the primary bottleneck in RAG, as it not only defines the information boundary (Cao et al., 2024), but also dominates the system's computational overhead (Jin et al., 2024).

Existing RAG paradigms can be viewed as an evolution of retrieval strategies (Gao et al., 2023). NaiveRAG relies on similarity-based retrieval over unstructured text chunks, favoring efficiency for factoid QA and summarization (Karpukhin et al., 2020). GraphRAG adopts graph retrieval to enable multi-hop reasoning (Edge et al., 2024), while HybridRAG further combines complementary signals such as vector and graph retrieval (Sarmah et al., 2024). IterativeRAG dynamically invokes retrieval modules based on intermediate states, trading efficiency for improved reasoning capability (Asai et al., 2024). Together, these paradigms turn retrieval into a multi-criteria decision problem, highlighting the necessity of *Adaptive RAG Routing* (Jeong et al., 2024; Tang et al., 2025).

Dataset	Design			Domain			Query			Corpus		Evaluation	
	Query	Corpus	Routing	Wiki.	Lr.	Ps.	Fac.	Rea.	Sum.	Sem.	Str.	Effect.	Effi.
HotpotQA (Yang et al., 2018)	✓	✗	Fixed	✓	✗	✗	✗	✓	✗	✓	✗	✓	✓
MuSiQue (Trivedi et al., 2022)	✓	✗	Fixed	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓
MultiHop-RAG (Tang and Yang, 2024)	✓	✗	Fixed	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗
WebQSP (Yih et al., 2016)	✓	✗	Fixed	✓	✗	✗	✓	✓	✗	✗	✗	✓	✓
QuALITY (Pang et al., 2022)	✓	✗	Fixed	✗	✓	✗	✗	✓	✓	✗	✗	✓	✗
GraphRAG-Bench (Xiang et al., 2025)	✓	✗	Fixed	✗	✗	✓	✓	✓	✓	✗	✓	✓	✗
GraphRAG-Bench (Xiao et al., 2025)	✓	✗	Fixed	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓
MemoRAG (Qian et al., 2024)	✗	✓	Fixed	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓
RAGRouter-Bench (Ours)	✓	✓	**Adaptive**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison with existing RAG benchmarks. Domain: Wiki (Wikipedia), Lr (Literature), PS (Professional Specialized). **Query:** Fac. (Factual), Rea. (Reasoning), Sum. (Summary). Corpus: Sem. (Semantic), Str. (Structural). Evaluation: Effect. (Effectiveness), Effi. (Efficiency).

To ground this discussion, we conduct preliminary experiments by evaluating representative RAG paradigms across different corpora (Figure 1), yielding three key insights. (i) no single paradigm consistently dominates across all settings, indicating the absence of a one-size-fits-all solution; (ii) the optimal RAG choice depends not only on query characteristics but also critically on the underlying corpus; (iii) more sophisticated methods do not guarantee better performance, as simpler alternatives can achieve comparable results with substantially lower overhead. Together, these findings underscore that RAG routing hinges on query–corpus compatibility and effectiveness–efficiency trade-offs, calling for systematic benchmarking across queries, corpora, and retrieval strategies.

Nevertheless, existing research exhibits several limitations, as summarized in Table 1. **(1) Query-centric Assumption.** Prior studies largely assume that the optimal RAG paradigm is attributed solely to the semantic complexity or reasoning difficulty of the query (Jeong et al., 2024; Tang et al., 2025). This query-centric view systematically overlooks the semantic and structural properties of the corpus, and more fundamentally, ignores the query-corpus compatibility that essentially shapes RAG effectiveness. **(2) Missing Fine-grained Signals for Routing.** Existing studies (Gao et al., 2023; Peng et al., 2024) primarily examine query-side factors in isolation, without providing quantifiable metrics to support causal and interpretable analysis of how routing decisions are jointly shaped by fine-grained characteristics of both queries and corpora. This limitation obscures our understanding of the applicability boundaries of different RAG paradigms and hinders principled routing design. **(3) Lack of Routing-Oriented Benchmarks.** Current RAG datasets and benchmarks suffer from limited domain coverage, incomparable method designs, and insufficient consideration of effectiveness–efficiency trade-offs (Chen et al., 2024; Lyu et al., 2024; Friel et al., 2024; Jin et al., 2024), which together preclude comprehensive and systematic identification of suitable and sustainable RAG approaches across query–corpus combinations and constraining further development of adaptive and interpretable RAG systems.

To this end, we introduce RAGRouter-Bench, a multi-dimensional dataset and benchmark tailored for adaptive RAG routing. Motivated by query–corpus compatibility, RAGRouter-Bench models each instance as a **(query, corpus, method, performance)** tuple, enabling systematic and interpretable analysis of routing behaviors across diverse settings. The benchmark comprises 7,727 queries over 21,460 documents, supporting large-scale analysis of performance across diverse query–corpus combinations. Specifically, we first standardize a set of representative RAG paradigms under a unified retriever abstraction, allowing fair and comparable evaluation despite diverse variants in real-world applications. RAGRouter-Bench then captures variability from both the query and corpus perspectives. On the query side, we curate and augment three canonical query types, namely factual, reasoning, and summarization. On the corpus side, the benchmark spans multiple domains and characterizes corpora using both semantic and structural properties, enabling fine-grained analysis of how corpus characteristics, individually and interactively, influence routing decisions. Finally, RAGRouter-Bench adopts a unified evaluation protocol to examine effectiveness–efficiency trade-offs, where we not only measures response quality through quantitative metrics and LLM-as-a-judge evaluation, but also report construction and inference as efficiency indicators.

In summary, our contributions are four-fold: (i) We introduce the first dataset and benchmark for adaptive RAG routing, providing a comprehensive

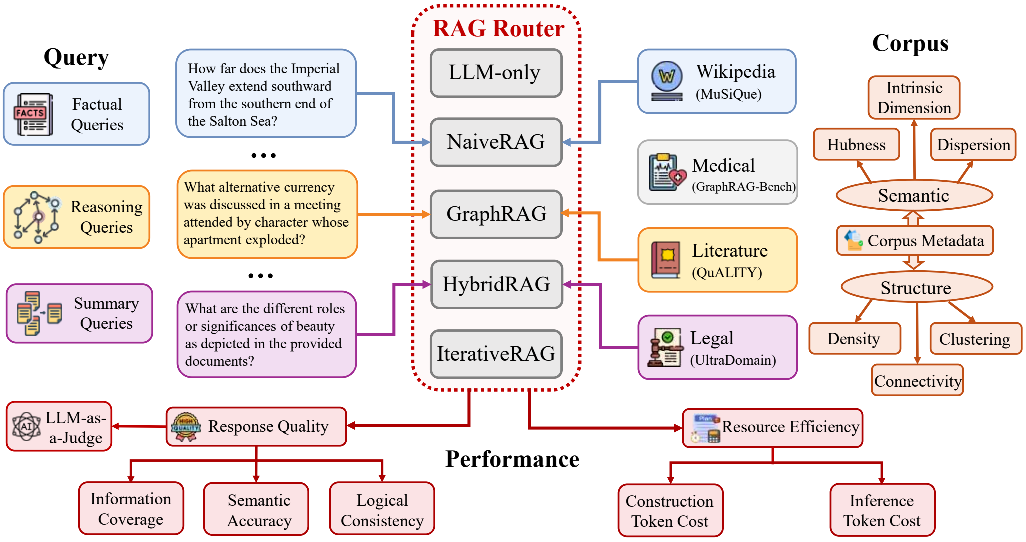

Figure 2: Overview of the RAGRouter-Bench framework. Left: Query types with representative examples. **Center:** Five RAG paradigms as routing targets. **Right:** Multi-domain corpora with structural and semantic characterization. **Bottom:** Dual-axis evaluation covering response quality and resource efficiency.

evaluation of standard RAG paradigms; (ii) We identify query-corpus compatibility as the key to RAG routing, and introduce fine-grained dual-view indicators to characterize the application boundaries; (iii) We propose a unified evaluation protocol grounded in effectiveness-efficiency trade-offs; (iv) We conduct extensive experiments using DeepSeek-V3 and LlaMA-3.1-8B as LLM backbones, offering insights toward adaptive, interpretable, and generalizable next-generation RAG systems.

2 Related Work

Retrieval Paradigms in RAG. RAG has evolved from flat semantic matching to structured integration. Foundational VectorRAG models like DPR (Karpukhin et al., 2020) and REALM (Guu et al., 2020) utilize dense retrieval for semantic similarity but often miss complex structural dependencies, increasing hallucination risk on long-tail knowledge (Ji et al., 2023; Mallen et al., 2023). Conversely, GraphRAG frameworks such as HippoRAG (Jiménez Gutiérrez et al., 2024), G-Retriever (He et al., 2024), and Think-on-Graph (Sun et al., 2024) leverage knowledge graphs for structure-aware reasoning, excelling at entity-centric tasks yet struggling with abstractive queries. Addressing these trade-offs, HybridRAG (Sarmah et al., 2024) fuses vector and graph contexts, while recursive methods like Self-RAG (Asai et al., 2024) handle multi-hop complexity through iterative retrieval. However, most work treats these paradigms as competing alternatives rather than context-dependent choices (Gao et al., 2023).

RAG Benchmarks and Evaluation. RAG evaluation has evolved from basic metrics to multi-dimensional benchmarks. Automated frameworks like RAGAS (Es et al., 2024) and ARES (Saad-Falcon et al., 2024) employ LLM-as-a-Judge for reference-free evaluation. RGB (Chen et al., 2024) tests noise resilience, CRUD-RAG (Lyu et al., 2024) and RAGBench (Friel et al., 2024) categorize diverse retrieval tasks, while MultiHop-RAG (Tang and Yang, 2024) and GraphRAG-Bench (Xiao et al., 2025) address complex reasoning scenarios. Recent work on embedding space analysis, including intrinsic dimensionality estimation (Facco et al., 2017) and hubness characterization (Radovanović et al., 2010), provides tools for corpus-level diagnostics, yet these insights remain disconnected from RAG paradigm selection. Meanwhile, adaptive routing methods (Jeong et al., 2024; Tang et al., 2025) toggle strategies based on query complexity but overlook corpus properties, leaving query-corpus compatibility unexplored.

3 Preliminaries

RAG Framework. Standard RAG follows a retrieve-then-generate pipeline. We adopt a modular view, decomposing retrieval into atomic modules and a scheduling policy that orchestrates them. This abstraction unifies existing paradigms and accommodates emerging variants.

Adaptive RAG Routing. We formalize Adaptive RAG Routing as: given a query-corpus pair $(q, \mathcal{C})$, select the optimal paradigm π^* that maximizes a utility function $\mathcal{U}$:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathcal{U}(\pi; q, \mathcal{C}) \qquad (1)$$

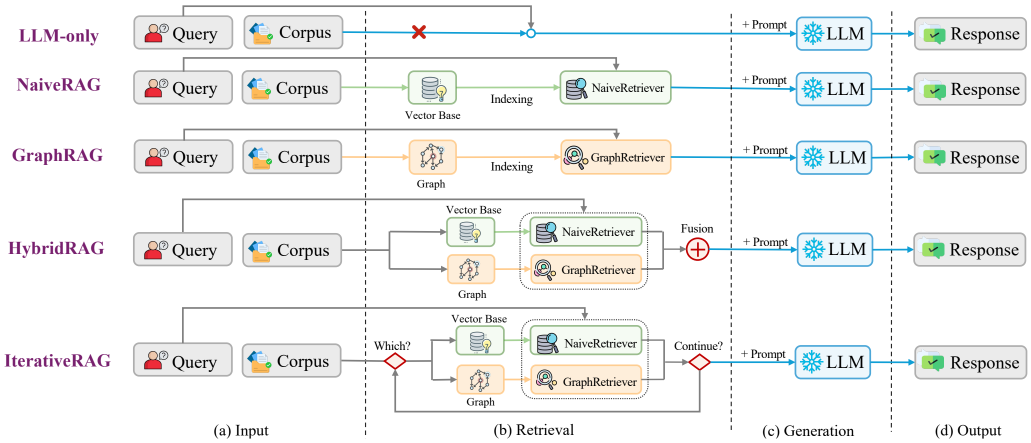

Figure 3: Overview of the five RAG paradigms evaluated in RAGRouter-Bench. Input: Query and corpus shared across all paradigms. **Retrieval:** Paradigm-specific pipelines differing in index structures and retrieval strategies. **Generation:** Retrieved context combined with query as prompt to LLM. **Output:** Final response.

where Π is the candidate paradigm space, and $\mathcal{U}$ captures task-specific criteria such as answer quality, efficiency, or their trade-off. This formulation frames RAG routing as context-dependent paradigm selection, the central challenge motivating RAGRouter-Bench.

4 The RAGRouter-Bench

We construct RAGRouter-Bench to investigate how query and corpus characteristics jointly influence paradigm selection, comprising a diverse dataset and an effectiveness-efficiency evaluation framework (Figure 2).

4.1 RAG Paradigm Instantiation

To enable cross-paradigm comparison, we define two base retrievers and instantiate five RAG paradigms from these components.

Base Retrievers. Rather than evaluating specific implementations, we decompose RAG methods into atomic modules (e.g., vector search, entity extraction, graph traversal). This enables fair cross-paradigm comparison while covering mainstream real-world approaches. We define two atomic retrieval modules as building blocks. NaiveRetriever performs dense vector-based semantic retrieval, which encodes queries and chunks into latent vectors and retrieves top-K segments via cosine similarity (Karpukhin et al., 2020). GraphRetriever operates on knowledge graphs, which extracts seed entities from the query, propagates relevance scores via Personalized PageRank (PPR) (Haveliwala, 2002), and returns text from high-relevance nodes (Edge et al., 2024).

RAG Paradigm Instances. Building on these retrievers, we define five paradigms spanning a broad spectrum: LLM-only bypasses retrieval, prompting the LLM with the query alone as a retrieval-free baseline (Petroni et al., 2019). NaiveRAG invokes NaiveRetriever once and concatenates retrieved chunks as context (Lewis et al., 2020). GraphRAG applies GraphRetriever to retrieve high-relevance nodes, then extracts their associated triplets and text chunks as context for generation (Edge et al., 2024). HybridRAG invokes both retrievers in parallel and merges results via Reciprocal Rank Fusion (RRF) (Cormack et al., 2009; Sarmah et al., 2024). IterativeRAG employs a Retrieve-Generate-Evaluate feedback loop, decomposing complex queries into sub-queries and iterating until an LLM-based evaluator confirms answer completeness (Asai et al., 2024; Trivedi et al., 2023).

These paradigms, spanning retrieval-free to dynamic iteration, enable systematic comparison across retrieval strategies (Figure 3). Implementation details are provided in Appendix B.2.

4.2 Data Curation

Existing benchmarks use homogeneous data sources and query types, obscuring how corpus characteristics affect retrieval (Chen et al., 2024; Friel et al., 2024). We address this through diverse corpus sourcing and query generation.

Corpus Sourcing. We integrate datasets from four domains: wikipedia (MuSiQue (Trivedi et al., 2022)), literature (QuALITY (Pang et al., 2022)), legal documentation (UltraDomain_legal (Qian et al., 2024)), and medical textbooks (GraphRAG-Bench_medical (Xiao et al., 2025)), totaling 21,460 documents.

Query Generation. These original benchmarks exhibit skewed query distributions, for example,

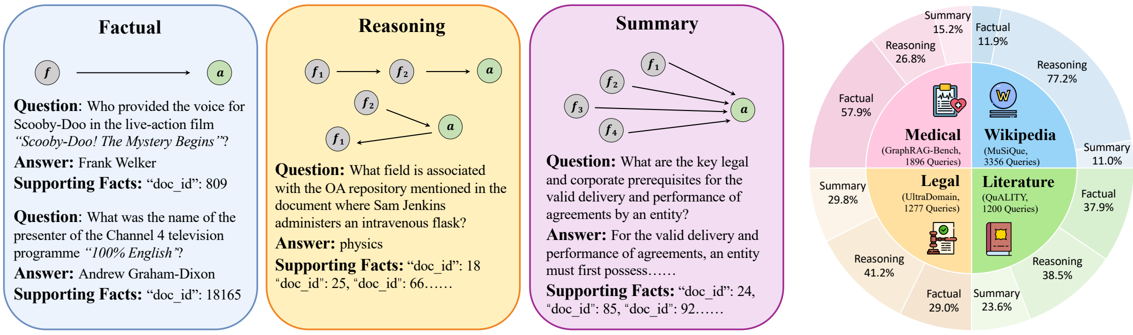

Figure 4: Query type taxonomy and dataset composition in RAGRouter-Bench. Left: Three query types definition. **Right:** Query type distribution across four datasets.

MuSiQue contains only reasoning queries, while QuALITY is over 90% factual (Trivedi et al., 2022; Pang et al., 2022). To enable cross-type comparison within each corpus, we augment queries via LLM-based generation guided by strcuture-aware expanding (Xiao et al., 2025). This workflow is validated using a verify-then-filter protocol (Chen et al., 2024) to mitigate potential bias in LLM-based evaluation. Specifically, human verification is conducted on a stratified sample of 200 queries (50 per corpus), achieving a 94% agreement rate with automated judgments.

To decouple query characteristics from corpus structure, we generate three query categories across the unified corpora: Factual queries target single entities with answers retrievable from one segment; Reasoning queries require inference across multiple segments; Summary queries demand aggregation from multiple sources. The dataset finally comprises 7,727 queries: 4,086 reasoning (52.9%), 2,320 factual (30.0%), and 1,321 summary (17.1%). Each dataset covers all three types, enabling cross-type comparison (Figure 4). By integrating multi-domain corpora with multi-type queries, RAGRouter-Bench decouples query typologies from corpus characteristics, establishing a foundation for analyzing their effects. Detailed preprocessing and generation protocols are provided in Appendix A.1 and A.2.

4.3 Dual-View Analysis

RAG efficacy hinges on query-corpus interplay rather than query complexity alone. Drawing on prior work in graph topology and embedding quality assessment (Newman, 2010; Ethayarajh, 2019), we establish a dual-view analytical framework in corpus constraints and query demands.

Corpus Analysis Dimension. The corpus serves as the physical environment dictating retrieval feasibility. We characterize corpus constraints along two axes: structural topology and semantic space.

Structural Topology Metrics. Structural metrics characterize knowledge graph topology along three dimensions. Connectivity captures global reachability via two measures: LCC Ratio quantifies the proportion of nodes in the main subgraph, where low values imply fragmentation that severs multi-hop paths (Newman, 2010); Relation Type Diversity measures edge semantic variety for precise graph traversal. Density quantifies edge saturation: Average Degree reflects connection intensity, while Maximum Degree Centrality identifies hub nodes with disproportionate connectivity, excessive sparsity lacks relational bridges, whereas hub dominance introduces bias (Sun et al., 2024). Clustering Coefficient assesses local cohesiveness, with high values indicating tight communities that facilitate evidence aggregation (Watts and Strogatz, 1998).

Semantic Space Metrics. Semantic metrics capture embedding space properties along a causal chain from space complexity to retrieval effects. Intrinsic Dimension, estimated via TwoNN, measures effective degrees of freedom, high dimensionality exacerbates the curse of dimensionality, diminishing distance-based similarity (Facco et al., 2017). Dispersion characterizes distribution uniformity through three centroid-based distances: average distance captures overall spread, minimum distance identifies the closest (most confusable) cluster pair, and standard deviation reveals distributional imbalance, collectively, low dispersion causes semantic crowding that hinders hard-negative discrimination (Wang and Isola, 2020). Hubness, measured as the skewness of k-occurrence distribution, quantifies retrieval interference from hub vectors that dominate nearest-neighbor lists, causing systematic bias toward frequently retrieved but potentially irrelevant passages (Radovanović et al., 2010).

Query Analysis Dimension. We categorize queries by reasoning complexity into three types:

Dataset	NaiveRAG					GraphRAG					HybridRAG					IterativeRAG				
	Fac.	Rea.	Sum.	**Avg.**	Tok.	Fac.	Rea.	Sum.	**Avg.**	Tok.	Fac.	Rea.	Sum.	**Avg.**	Tok.	Fac.	Rea.	Sum.	**Avg.**	Tok.
Model: DeepSeek-V3																				
MuSiQue	11.1	28.3	29.4	26.4	13k	**90.2**	22.5	20.6	30.3	9k	83.7	**32.8**	**30.2**	**38.6**	22k	10.3	21.8	21.2	20.4	20k
QuALITY	**83.7**	**33.8**	17.0	**48.8**	50k	70.7	20.4	**19.8**	39.3	50k	80.0	20.4	14.8	41.6	99k	67.0	17.1	16.2	35.8	21k
Legal	54.9	11.2	**39.1**	32.2	46k	61.6	6.7	29.1	29.3	184k	**72.2**	11.8	34.6	**36.1**	230k	49.5	**12.4**	30.4	28.5	20k
Medical	63.1	63.5	49.1	61.1	51k	55.0	56.0	43.6	53.5	38k	**67.8**	64.0	54.0	**64.7**	74k	62.1	**67.8**	**56.1**	62.7	7k
Model: LLaMA-3.1-8B																				
MuSiQue	10.3	7.9	12.0	8.6	13k	**84.4**	9.7	11.4	18.7	9k	79.9	**12.1**	**13.9**	**20.3**	22k	12.3	6.0	6.8	6.8	20k
QuALITY	69.2	10.9	1.4	30.7	50k	44.7	9.5	**2.5**	21.2	50k	**70.3**	**15.6**	**2.5**	**33.2**	99k	62.3	11.5	1.4	28.4	21k
Legal	50.3	10.7	22.8	25.8	46k	55.4	7.4	19.7	25.0	184k	**63.8**	**13.1**	**24.2**	**31.1**	230k	48.7	10.1	12.6	22.0	20k
Medical	52.1	37.9	30.1	44.9	51k	48.1	33.6	31.8	41.7	38k	**55.7**	39.9	**33.9**	**48.2**	74k	47.5	**44.6**	27.3	43.6	7k

Table 2: Main evaluation results across RAG paradigms, datasets, and backbone LLMs. Each paradigm reports LLM-as-a-Judge accuracy (%) by query type (Factual, Reasoning, Summary) and overall average, along with average token consumption per query. Bold indicates best-performing paradigm per dataset.

Factual queries require single-hop lookup from one fact (Chen et al., 2024); Reasoning queries demand multi-hop inference across chained facts (Yang et al., 2018); Summary queries involve global aggregation over dispersed information (Edge et al., 2024). To ensure every corpus covers all three types, we augment queries via the LLM-based pipeline described in Section 4.2.

This dual-view framework quantitatively assesses how corpus attributes and query demands jointly shape RAG paradigm effectiveness. Details are provided in Appendix A.3.

4.4 Evaluation Protocol

Uni-dimensional evaluation fails to capture practical RAG performance. To evaluate effectiveness-efficiency trade-offs across paradigms, we construct a protocol in two dimensions: generation quality and resource consumption (Jin et al., 2024).

Generation Quality Evaluation. We assess generation outcomes across answer quality, factual faithfulness, and holistic correctness. Semantic F1 calculates token-level similarity between generated responses and gold standards via BERTScore (Zhang et al., 2020). Coverage quantifies the extent to which answers cover key information using sentence embeddings (Reimers and Gurevych, 2019). Faithfulness measures average support strength between answers and retrieved content (Es et al., 2024). LLM-as-a-Judge employs an LLM for ternary classification, providing correctness aligned with human judgment (Zheng et al., 2023).

Resource Consumption Evaluation. We adopt token consumption as the efficiency metric, decomposing total cost into retrieval and generation components. Retrieval Cost captures LLM invocation overhead during retrieval, including entity extraction and multi-turn queries. Generation Cost measures overhead during generation, predominantly governed by context length. Construction costs for one-time graph building in GraphRAG and HybridRAG are amortized over queries and incorporated into retrieval cost (Edge et al., 2024). In practice, generation input is truncated to 8k tokens to accommodate LLM context limits; cost metrics report full retrieval output, as retrieved content is relevance-ranked and truncation removes only lower-ranked passages with minimal impact on response quality.

Our evaluation protocol considers effectiveness-efficiency trade-offs across candidate paradigms to inform optimal RAG routing decisions. Formal definitions are provided in Appendix C.1 and C.2.

5 Experiments

We conduct experiments to evaluate how query and corpus jointly influence paradigm selection. We compare paradigm performance across datasets first, then analyze corpus-driven and query-driven effects, finally assess cost-performance trade-offs.

5.1 Experimental Setup

RAG Paradigms. We standardize infrastructure across all paradigms for fair comparison. We use DeepSeek-V3 (DeepSeek-AI, 2024) and LLaMA-3.1-8B (Team, 2024) as generators, and text-embedding-3-small for vectorization (OpenAI, 2024), with a unified 8k token context budget. For retrieval, NaiveRAG retrieves top-100 chunks via cosine similarity (Karpukhin et al., 2020); GraphRAG extracts 20 seed entities and propagates relevance via PPR ($\alpha = 0.85$) (Haveliwala, 2002) to retrieve top-100 nodes (Edge et al., 2024); HybridRAG combines both retrievers (Sarmah et al., 2024); IterativeRAG performs up to 3 retrieve-generate-evaluate iterations (Trivedi et al., 2023). Details are provided in Appendix B.1.

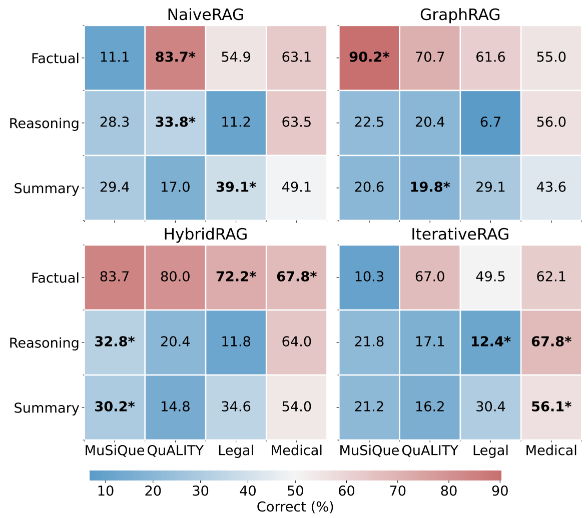

Figure 5: Paradigm performance across datasets and query types. Each panel shows one RAG paradigm's LLM-as-a-Judge accuracy (Correct%), with rows as query types and columns as datasets. Asterisk (*) marks the best-performing paradigm for each combination.

Evaluation. We evaluate following the protocol in Secti 4.4 across three dimensions. For corpus analysis, we compute structural metrics (LCC Ratio, Density, Clustering Coefficient) (Newman, 2010; Sun et al., 2024; Watts and Strogatz, 1998) and semantic metrics (Intrinsic Dimension, Dispersion, Hubness) (Facco et al., 2017; Wang and Isola, 2020; Radovanović et al., 2010). For generation quality, we measure Semantic F1 (Zhang et al., 2020), Coverage, Faithfulness (Es et al., 2024), and LLM-as-a-Judge accuracy using GPT-4o as the evaluator (OpenAI, 2023). For efficiency, we track token consumption decomposed into retrieval and generation costs (Jin et al., 2024). Implementation details are provided in Appendix C.1.

5.2 Main Results

Comparative Paradigm Analysis. No universal retrieval paradigm exists across query-corpus combinations, as shown in Table 2 and Figure 5. On the same Factual query type, GraphRAG achieves 90.2% on MuSiQue but only 70.7% on QuALITY, while NaiveRAG shows the opposite pattern (11.1% on MuSiQue vs 83.7% on QuALITY). This reversal demonstrates that optimal strategy depends jointly on corpus structure and query type. Each paradigm exhibits distinct strengths. NaiveRAG excels in implicit narratives (83.7% on QuALITY Factual), where preserving continuous context outweighs structured retrieval. GraphRAG dominates entity-centric tasks in explicit graphs (90.2% on MuSiQue Factual), leveraging entity linking to overcome vector ambiguity. HybridRAG achieves the highest average accuracy on 3 of 4 datasets by combining semantic coverage with structural reasoning. IterativeRAG underperforms expectations; without topology-aligned paths, its refinement cannot correct initial retrieval drift.

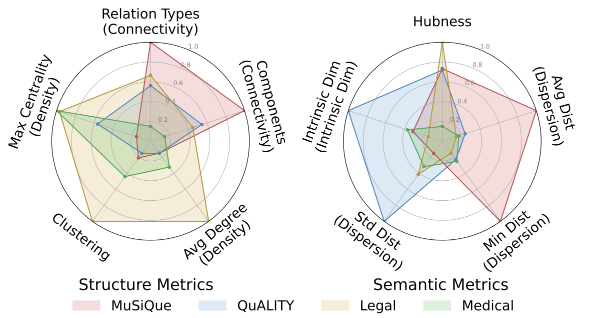

Figure 6: Corpus features across structural and semantic dimensions. Left: Graph topology metrics capturing knowledge graph properties. **Right:** Embedding space metrics characterizing semantic distribution.

Corpus-Driven Performance Constraints. The performance reversal in Paradigm Analysis stems from distinct corpus characteristics. Figure 6 reveals that each corpus exhibits a unique fingerprint across structural and semantic dimensions. Structural properties determine graph retrieval effectiveness. MuSiQue's high relation diversity and explicit entity links enable GraphRAG to achieve 90.2% on Factual queries via precise graph traversal. In contrast, QuALITY's linear narrative structure yields sparse, fragmented graphs where forced graph construction introduces noise, explaining why NaiveRAG outperforms GraphRAG (83.7% vs 70.7% on Factual). Semantic properties constrain vector retrieval precision. Legal corpus exhibits high hubness and low semantic dispersion, causing vector space congestion that limits NaiveRAG's discrimination ability. This explains its moderate performance (54.9%) despite rich textual content, while HybridRAG bypasses this bottleneck through structured paths (72.2%). Complex corpora require complementary retrieval. Medical corpus combines moderate structural density with semantic complexity. Neither single-modality retriever suffices, HybridRAG consistently leads by fusing semantic coverage with structural reasoning (67.8% vs 63.1% NaiveRAG, 55.0% GraphRAG).

Adaptability across Query Types. Query type adds another dimension to paradigm selection. Figure 7 reveals distinct response distributions across query complexities. Factual queries demand precise anchoring. GraphRAG achieves the highest accuracy (84.4%) by leveraging entity linking in explicit graphs, but suffers high Refused rates (22%) when graph structure is sparse. NaiveRAG

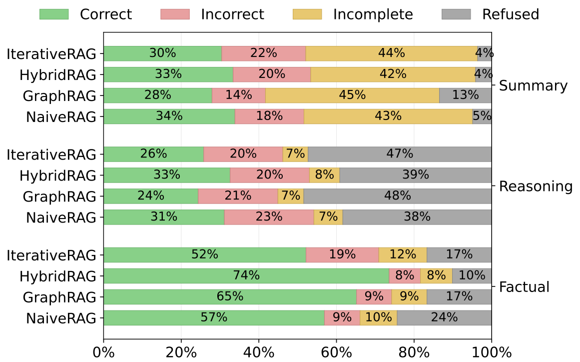

Figure 7: Response quality distribution across RAG paradigms and query types. Each bar shows the breakdown of LLM-as-a-Judge outcomes across all datasets. **Left axis:** Paradigms; **Right axis:** Query type.

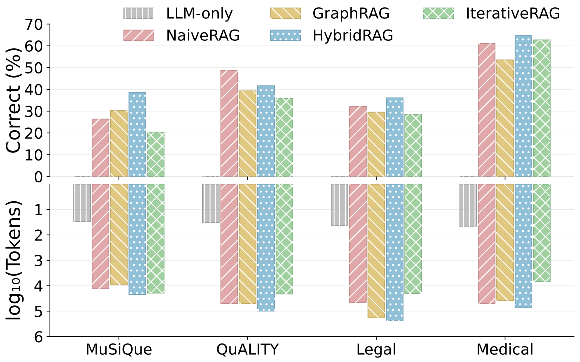

Figure 8: Cost-performance trade-off across RAG paradigms and datasets. Top: LLM-as-a-Judge accuracy (Correct%). **Bottom:** average token consumption per query (log scale).

provides more stable performance across corpora (avg. 52%) through direct semantic matching. Reasoning queries require link completion. Multi-hop reasoning exposes single-modality limitations, both NaiveRAG and GraphRAG show elevated Incorrect rates (20-25%) due to incomplete evidence chains. HybridRAG leads by combining vector entry points with graph-based path completion, reducing Incomplete responses by 8% compared to alternatives. Summary queries need coverage matching. Performance depends on corpus-query alignment: NaiveRAG preserves coherent context for narrative synthesis (69.2% on QuALITY), while HybridRAG provides broader coverage for attribute aggregation in structured domains (67.8% on Medical). These confirm that query characteristics interact with corpus properties, neither dimension alone determines the optimal paradigm.

Cost-Performance Trade-off Analysis. Beyond accuracy, practical deployment requires considering computational cost. Figure 8 reveals that token consumption spans 3 orders of magnitude across paradigms. LLM-only serves as the minimal-cost baseline but achieves zero accuracy without retrieval, confirming retrieval augmentation is essential for knowledge-intensive tasks. NaiveRAG and GraphRAG occupy the mid-cost tier with comparable token consumption ($\sim 10^4$ per query). The choice between them depends on corpus characteristics rather than cost: GraphRAG for explicit structures, NaiveRAG for implicit narratives. HybridRAG incurs the highest cost by combining both retrievers, but consistently achieves top accuracy across datasets. This trade-off is justified in high-stakes scenarios where correctness outweighs computational expense. IterativeRAG presents flexible cost depending on iteration count and base retriever configuration. While potentially efficient for simple queries that terminate early, it offers no consistent cost-accuracy advantage in our experiments. These trade-offs show that resource-constrained deployments should select between NaiveRAG and GraphRAG based on corpus type, while accuracy-critical applications benefit from HybridRAG despite higher cost. Representative case studies are provided in Appendix C.3.

6 Limitations

Our analysis centers on paradigm-level mechanistic differences rather than exhaustive benchmarking of specific implementations, aiming to elucidate compatibility trends between paradigms and corpora. Additionally, while our query generation approach guarantees logical soundness, synthetic queries may not fully capture the noise distribution characteristic of real-world interactions.

7 Conclusion

In this work, we present RAGRouter-Bench, the first dataset and benchmark explicitly designed for adaptive RAG routing. By revisiting retrieval from a query–corpus compatibility perspective, RAGRouter-Bench enables systematic comparison of representative RAG paradigms under unified effectiveness–efficiency evaluation. Extensive experiments demonstrate that RAG performance is highly context-dependent, shaped jointly by query characteristics, corpus properties, and retrieval strategies, and that more complex methods do not necessarily yield better trade-offs. These findings highlight retrieval as a critical decision point rather than a fixed design choice. We believe RAGRouter-Bench will facilitate principled routing research and support next-generation RAG systems.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Bowen Cao, Deng Cai, Leyang Cui, Xuxin Cheng, Wei Bi, Yuexian Zou, and Shuming Shi. 2024. Retrieval is accurate generation. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 411–418.

DeepSeek-AI. 2024. DeepSeek-V3 technical report. *Preprint*, arXiv:2412.19437.

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

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2024. RAGAS: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2024)*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 4809–4818.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2017. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7(1):1–8.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. RAGBench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning (ICML)*, pages 3929–3938.

Taher H Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th International Conference on World Wide Web*, pages 517–526.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2024)*, pages 1–15.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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

Jiajie Jin, Yutao Zhu, Guanting Dong, Yuyao Zhang, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, Zhicheng Dou, and Ji-Rong Wen. 2024. FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Yuanjie Lyu, Zhiyi Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Peng Wu, Ke Liu, Min Chen, and Xing Wan. 2024. CRUD-RAG: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *arXiv preprint arXiv:2401.17043*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Mark Newman. 2010. *Networks: An Introduction*. Oxford University Press.

OpenAI. 2023. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024. New embedding models and API updates. `https://openai.com/blog/new-embedding-models-and-api-updates`.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Arpit Phan, Jason andoba, Phu Mon Htut, Jheng-Hong Xie, Samuel R. Bowman, and He He. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. MemoRAG: Moving towards next-gen RAG via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 3982–3992.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*.

Bhaskarjit Sarmah, Benika Hall, Rohan Rao, Sunil Patel, Stefano Pasquali, and Dhagash Mehta. 2024. HybridRAG: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. *arXiv preprint arXiv:2408.04948*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Xiaqiang Tang, Qiang Gao, Jian Li, Nan Du, Qi Li, and Sihong Xie. 2025. MBA-RAG: a bandit approach for adaptive retrieval-augmented generation through question complexity. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025)*, pages 3248–3254.

Yixuan Tang and Yi Yang. 2024. MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*, pages 2369–2380.

Llama Team. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nandan Thakur, Nils Reimers, Andreas Rücklë, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Advances in Neural Information Processing Systems*, volume 34, pages 24453–24466.

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 (ACL 2023)*, pages 10014–10029.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning (ICML)*, pages 9929–9939.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442.

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*.

Yilin Xiao, Junnan Dong, Chuang Zhou, Su Dong, Qian-Wen Zhang, Di Yin, Xing Sun, and Xiao Huang. 2025. GraphRAG-Bench: Challenging domain-specific reasoning for evaluating graph retrieval-augmented generation. *arXiv preprint arXiv:2506.02404*.

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.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTscore: Evaluating text generation with BERT. In *International Conference on Learning Representations (ICLR 2020)*.

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

Appendix

A Data Construction & Corpus Analysis

A.1 Corpus Statistics & Preprocessing.

Data Overview To establish a benchmark encompassing diverse retrieval environments, we integrate four representative datasets spanning encyclopedic knowledge (MuSiQue, 21,100 Wikipedia articles), long-form narratives (QuALITY, 265 Gutenberg novels), specialized legal corpora (UltraDomain_Legal, 94 contract documents), and medical literature (GraphRAGBench_Medical, a single comprehensive textbook). As shown in Table 3, these datasets exhibit extreme disparities in both document count (ranging from 1 to 21,100) and average length (107.9 to 221,495 tokens), thereby serving as an ideal testbed for evaluating the adaptability of RAG paradigms across distinct scales and structural settings. The complete preprocessing pipeline is formalized in Algorithm 1.

Chunking Strategy. We employ a sliding window chunking strategy to process the raw corpora (see Table 4 for all hyperparameters). Specifically, utilizing the `cl100k_base` encoder from `tiktoken`, each document (concatenated title and content) is segmented into fixed-size chunks with a window size of 512 tokens and a 100-token overlap to preserve contextual coherence. This configuration strikes a balance between retrieval granularity and contextual completeness. Following this segmentation, the four datasets yield distinct chunk inventories: 21,153 (MuSiQue), 3,822 (QuALITY), 11,632 (Legal), and 538 (Medical).

Knowledge Graph Construction. To facilitate the structure-aware retrieval required by GraphRAG and HybridRAG, we extract knowledge triplets from each individual text chunk. We employ DeepSeek-V3 as the underlying extraction engine, with hyperparameters detailed in Table 4, setting the temperature to 0.0 to ensure deterministic generation. To optimize processing throughput, we implement an asynchronous parallelization strategy configured with a maximum concurrency of 15, a 60-second request timeout, and a retry mechanism allowing up to 3 attempts with exponential backoff. As illustrated in Figure 9, the extraction prompt instructs the model to identify `(Subject, Relation, Object)` triplets within the input text and format the output as a JSON array. The scale of the resulting knowledge graphs is detailed in Table 7. Specifically, MuSiQue comprises 206,738 entity nodes and 276,898 edges; QuALITY contains 90,088 nodes and 120,611 edges; the Legal corpus yields 135,231 nodes and 261,207 edges; and the Medical textbook generates 14,712 nodes and 21,480 edges. Graph density varies significantly from 6.0×10^{-6} (MuSiQue) to 9.9×10^{-5} (Medical), reflecting inherent disparities in structural sparsity across domains. All constructed graphs are represented as undirected graphs to facilitate bidirectional traversal.

Vectorization & Indexing. To facilitate the dense retrieval mechanism of NaiveRAG, we utilize the OpenAI `text-embedding-3-small` model to encode each text chunk into a 1,536-dimensional vector representation. We employ a batch size of 30 to mitigate API rate-limiting constraints. All generated vectors undergo L2 normalization and are subsequently indexed using `FAISS` (`IndexFlatIP`) to enable efficient retrieval based on cosine similarity. Concurrently, to support the entity-level retrieval required by GraphRAG, we generate distinct embeddings for all unique entities within the knowledge graphs, specifically, 206,738 for MuSiQue, 90,088 for QuALITY, 135,231 for Legal, and 14,712 for Medical. These entity embeddings serve as the foundation for precise entity matching and the selection of seed nodes for graph traversal during the query execution phase.

A.2 Query Generation Pipeline

Generation Overview. To construct a query set encompassing varying degrees of cognitive complexity, we devise three distinct query generation strategies, as detailed in Algorithm 2. All generation processes utilize DeepSeek-V3 as the backbone model, with a temperature setting of 0.7 to strike a balance between diversity and quality.

Factual Queries: We perform uniform random sampling of text segments from the corpus and employ the prompt illustrated in Figure 10 to steer the LLM in generating factual QA pairs. Crucially, the prompt enforces a *self-contained* constraint, ensuring that questions are semantically independent of context and devoid of ambiguous pronominal references.

Reasoning Queries: Leveraging the topological structure of the knowledge graph, we identify document chains linked via shared entities. Specifically, for k-hop inquiries, we execute random walks on the graph to locate k documents connected by

Dataset	Domain	Num. Docs	Avg. Tokens	Total Tokens
MuSiQue	Wikipedia	21,100	107.9	2,276,013
QuALITY	Narrative	265	5,741.1	1,521,395
UltraDomain_legal	Legal	94	50,785.0	4,773,793
GraphRAGBench_medical	Medical Textbook	1	221,495.0	221,495

Table 3: Raw Corpus Statistics. Overview of the source documents illustrating extreme variations in document length (Avg. Tokens) and scale (Num. Docs), ranging from fragmented Wikipedia articles to monolithic textbooks.

Algorithm 1 Unified Corpus Preprocessing Pipeline.

Require: Raw Corpus $\mathcal{C} = \{d_1, d_2, ..., d_n\}$, where each $d_i = (\text{title}, \text{text})$
Ensure: Knowledge Graph $\mathcal{G} = (V, E)$, Dense Vector Index $\mathcal{I}_{\text{vec}}$, Entity Embeddings $\mathbf{E}_{\text{ent}}$
Hyperparameters: Chunk Size $L = 512$, Overlap $O = 100$, Embedding Model $\mathcal{M}_{\text{emb}}$, LLM $\mathcal{M}_{\text{gen}}$

// Phase 1: Sliding Window Chunking
$\mathcal{S}_{\text{chunks}} \leftarrow \emptyset$
for each document $d \in \mathcal{C}$ **do**
  Tokens $\leftarrow$ Tokenize(d.title $\oplus$ d.text) ▷ Using `tiktoken` (cl100k_base)
  $ptr \leftarrow 0$
  while $ptr <$ len(Tokens) **do**
    $c_{\text{raw}} \leftarrow$ Tokens$[ptr : ptr + L]$
    $c_{\text{text}} \leftarrow$ Decode(c_{raw})
    $\mathcal{S}_{\text{chunks}} \leftarrow \mathcal{S}_{\text{chunks}} \cup \{(d.\text{id}, c_{\text{text}})\}$
    $ptr \leftarrow ptr + (L - O)$
  end while
end for
// Phase 2: Graph Construction & Entity Extraction
Initialize $V \leftarrow \emptyset, E \leftarrow \emptyset$
for each chunk $c \in \mathcal{S}_{\text{chunks}}$ **do**
  $\mathcal{T} \leftarrow \mathcal{M}_{\text{gen}}(\text{Prompt}_{\text{extract}}, c)$ ▷ Extract triplets (s, r, o) via DeepSeek
  for each triplet $(s, r, o) \in \mathcal{T}$ **do**
    $V \leftarrow V \cup \{s, o\}$
    $E \leftarrow E \cup \{(s, r, o)\}$
  end for
end for
// Phase 3: Vectorization & Indexing
$\mathbf{X}_{\text{chunks}} \leftarrow \emptyset$
for each chunk $c \in \mathcal{S}_{\text{chunks}}$ **do**
  $\mathbf{v}_c \leftarrow \mathcal{M}_{\text{emb}}(c)$ ▷ Dimension $d = 1536$
  $\mathbf{X}_{\text{chunks}}$.append($\mathbf{v}_c$)
end for
$\mathcal{I}_{\text{vec}} \leftarrow$ FAISS.Index($\mathbf{X}_{\text{chunks}}$) ▷ Build dense retrieval index
$\mathbf{E}_{\text{ent}} \leftarrow$ EmbedEntities(V) ▷ Embed unique entities for GraphRAG
return $\mathcal{G}, \mathcal{I}_{\text{vec}}, \mathbf{E}_{\text{ent}}$

bridge entities. Figure 11 depicts the generation prompt, which centers on a Reverse Substitution strategy: starting from the target answer, bridge entities are iteratively replaced with functional descriptions derived from preceding documents. This mechanism ensures that the resulting questions necessitate traversing the complete reasoning chain for resolution. We generate reasoning queries at both 2-hop and 3-hop complexity levels.

Summary Queries: We cluster documents by entity, selecting those with high connectivity within the graph as summarization targets. As shown in Figure 12, the prompt mandates an initial *consistency check*, verifying that multiple documents refer to the same entity rather than homonyms, followed by the synthesis of information from at least two documents. This process yields summarization questions that explicitly require cross-document integration for a complete response.

Verify-then-Filter Validation. Raw queries undergo a rigorous three-tiered verification protocol before inclusion in the final benchmark (Algorithm 2, Phase 2):

Grounding Check: We task an LLM with generating answers derived exclusively from the supporting facts. An LLM-as-a-Judge then validates the semantic consistency between the generated response and the expected answer. This step guarantees that the question is rigorously *answerable* given the provided documents.

Stage	Parameter	Value	Description
Chunking	Size	512	Fixed-size segment
	Overlap	100	Sliding window
	Tokenizer	cl100k_base	OpenAI encoding
Extraction	LLM	DeepSeek-V3	Base Model
	Temp.	0.0	Deterministic
	Concur.	15	Parallel requests
	Timeout	60s	API limit
	Retries	3	Fault tolerance
Graph	Directed	False	Undirected edges
Embedding	Model	text-emb-3-small	OpenAI Model
	Dim.	1536	Vector size
	Batch	30	API batch size

Table 4: Hyperparameters for Corpus Preprocessing.

Dataset	Type	Gen.	Pass	Rate
MuSiQue	Single-hop	700	398	56.9%
	2-hop	400	84	21.0%
	3-hop	400	89	22.2%
	Summary	527	368	69.8%
	Total	2,027	939	*46.3%*
QuALITY	Single-hop	500	454	90.8%
	2-hop	561	212	37.8%
	3-hop	600	249	41.5%
	Summary	789	283	35.9%
	Total	2,450	1,198	*48.9%*
Legal	Single-hop	400	370	92.5%
	2-hop	784	238	30.4%
	3-hop	800	288	36.0%
	Summary	983	381	38.8%
	Total	2,967	1,277	*43.0%*
Total		7,444	3,414	**45.9%**

Table 5: Verify-then-Filter validation statistics.

Shortcut Detection: We evaluate whether any single supporting fact suffices to answer the question in isolation. If an individual document yields the correct answer, the query is identified as containing a "shortcut", violating the intrinsic requirement of multi-hop reasoning, and is subsequently filtered out.

Knowledge Leakage Check: We screen for two forms of information leakage: (1) Lexical Leakage, where the answer appears as a substring within the question itself; and (2) Parametric Leakage, where the LLM can answer correctly relying solely on pre-trained knowledge. The latter is assessed via a closed-book test; if the model succeeds without retrieval context, the query is deemed ineffective for evaluating retrieval capabilities.

Human Verification: To address potential LLM self-validation bias in the above automated checks, we conduct manual verification on a stratified sample of 50 queries per corpus (N=200 total). Two annotators independently assess answerability and information leakage, achieving 91% inter-annotator agreement (Cohen's κ=0.85). The human-LLM agreement rate of 94% confirms the reliability of the automated filtering pipeline.

Validation Statistics. Table 5 presents the validation statistics across the constituent datasets. We generated a total of 7,444 candidate queries, of which 3,414 were retained following the *Verify-then-Filter* process, yielding an overall acceptance rate of 45.9%.

Pass rates exhibit significant variance across distinct query types. Single-hop queries achieve pass rates exceeding 90% on QuALITY and Legal datasets, yet only 56.9% on MuSiQue. This disparity is primarily attributed to MuSiQue's foundation in Wikipedia, where extensive factual overlap with the LLM's pre-training corpus frequently triggers the Knowledge Leakage filter. Multi-hop queries register the lowest pass rates (21%–42%), with the vast majority discarded by Shortcut Detection, underscoring the inherent challenge in generating questions that genuinely necessitate multi-step reasoning. Summary queries exhibit pass rates ranging from 36% to 70%, with failures predominantly stemming from the Grounding Check, specifically, semantic deviations between the response derived from the provided document set and the expected gold standard.

The final benchmark comprises validated queries from MuSiQue (939), QuALITY (1,198), and Legal (1,277). Integrating the Medical dataset from GraphRAGBench (1,896 pre-annotated questions), the final corpus totals 5,310 high-quality queries, spanning four domains and three levels of cognitive complexity.

A.3 Corpus Evaluation Metrics

Table 6 summarizes the core metrics employed to characterize the retrieval environment. By quantifying corpus properties along the dual dimensions of *structural topology* and *semantic space*, these metrics provide a quantitative foundation for delineating the applicability boundaries of distinct RAG paradigms.

Structural Topology Metrics. We employ three graph-theoretic metrics to evaluate the topological structure of the knowledge graphs (Table 7):

LCC Ratio (Largest Connected Component Ratio): This metric measures global reachability. It is defined as the ratio of nodes in the largest con-

Dimension	Metric	Physical Interpretation	Constraint Mechanism on Retrieval
Structural	LCC Ratio	Global Reachability	Fragmentation into isolated components severs reasoning paths for multi-hop retrieval.
	Density	Edge Saturation	Sparse graphs lack sufficient relational bridges; overly dense graphs introduce noise.
	Clustering Coeff.	Local Coherence	Facilitates evidence aggregation within topical neighborhoods.
Semantic	Intrinsic Dim.	Effective Degrees of Freedom	High dimensionality exacerbates the curse of dimensionality, degrading similarity metrics.
	Dispersion	Semantic Spread	Low dispersion causes semantic crowding, hindering distinction of hard negatives.
	Hubness	Retrieval Interference	Hub embeddings dominate neighbor lists, causing systematic retrieval bias.

Table 6: Key corpus metrics characterizing the retrieval environment.

Dataset	Nodes	Edges	Density	Rel. Types	Avg. Deg.	Comp.	LCC Ratio	Cluster. Coeff.
MuSiQue	206,738	276,898	6.00e-06	44,766	2.68	7,722	0.882	0.0213
QuALITY	90,088	120,611	1.50e-05	23,828	2.68	3,997	0.883	0.0177
Legal	135,231	261,207	1.40e-05	28,799	3.86	3,204	0.933	0.0701
Medical	14,712	21,480	9.90e-05	4,169	2.92	741	0.861	0.0357

Table 7: Full Structural Statistics. Detailed graph topology metrics including node/edge counts, graph density, number of unique relation types, average node degree, number of connected components (Comp.), ratio of the largest connected component (LCC Ratio), and average clustering coefficient.

nected component to the total number of nodes:

$$\text{LCC Ratio} = \frac{|V_{\text{LCC}}|}{|V|} \tag{2}$$

where V_{LCC} denotes the node set of the largest connected component and V represents the total node set. A lower ratio indicates severe graph fragmentation, increasing the risk that multi-hop reasoning paths are physically severed.

Density: This metric measures edge saturation. For an undirected graph, it is defined as:

$$D = \frac{2|E|}{|V|(|V|-1)} \tag{3}$$

where $|E|$ is the number of edges and $|V|$ is the number of nodes. Excessively low density implies a lack of sufficient relational bridges between entities, while excessively high density introduces noise, thereby degrading graph traversal efficiency.

Clustering Coefficient: This metric measures local cohesiveness. The clustering coefficient for a node v is defined as the ratio of actual edges between its neighbors to the number of possible edges:

$$C_v = \frac{2 \cdot |\{e_{jk} : v_j, v_k \in N(v), e_{jk} \in E\}|}{k_v(k_v - 1)} \tag{4}$$

where $N(v)$ is the neighborhood set of node v, and $k_v = |N(v)|$ represents the node degree. The global clustering coefficient is the average over all nodes. A high coefficient indicates the presence of tight-knit thematic communities, facilitating local evidence aggregation.

Semantic Space Metrics We employ three vector space metrics to assess embedding quality (Table 8):

Intrinsic Dimension: This metric quantifies the effective degrees of freedom within the embedding space. We estimate it using the **TwoNN** algorithm: for each data point, we calculate the distance to its nearest neighbor (r_1) and second-nearest neighbor (r_2). Letting $\mu = r_2/r_1$, the intrinsic dimension is defined as:

$$d_{\text{int}} = \frac{1}{\mathbb{E}[\ln \mu]} \tag{5}$$

High intrinsic dimensionality exacerbates the *curse of dimensionality*, rendering distance-based similarity metrics ineffective.

Dispersion: This metric measures the uniformity of semantic distribution. We compute the cosine distance of each embedding vector from the global centroid:

$$\text{dist}(x_i) = 1 - \frac{x_i \cdot \bar{x}}{\|x_i\|\|\bar{x}\|} \tag{6}$$

where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ denotes the centroid vector. Table 8 reports the mean, standard deviation, minimum, and maximum of these distances. Low dispersion results in *semantic crowding*, hinder-

Dataset	Chunks	Int. Dim.	Hubness	Avg. Dist.	Std. Dist.	Min. Dist.	Max. Dist.
MuSiQue	21,153	8.17	1.27	0.708	0.049	0.552	0.924
QuALITY	3,822	10.75	1.26	0.345	0.119	0.186	0.805
Legal	11,632	7.56	1.46	0.300	0.071	0.147	0.792
Medical	538	8.39	0.86	0.312	0.063	0.196	0.700

Table 8: Full Semantic Statistics. Metrics covering vector space properties: total number of text chunks, intrinsic dimension (Int. Dim.), Hubness score (interference), and centroid distance statistics (Average, Standard Deviation, Minimum, Maximum).

ing the retriever's ability to distinguish between semantically similar yet factually unrelated "hard negatives."

Hubness: This metric quantifies the extent of retrieval interference. Defining $N_k(i)$ as the number of times point i appears in the k-nearest neighbor lists of all other points (k-occurrence), Hubness is calculated as the skewness of this distribution:

$$S_k = \frac{\mathbb{E}[(N_k - \mu_{N_k})^3]}{\sigma_{N_k}^3} \quad (7)$$

Positive skewness indicates the presence of "hub" embeddings, vectors that frequently appear in the nearest neighbor lists of others. This phenomenon causes a systematic bias in retrieval results towards these hubs, thereby reducing both retrieval diversity and accuracy.

B RAG Paradigm Implementation

Category	Parameter	Value	Description
LLM	Model	DeepSeek-V3	Generation backbone
	Temperature	0.3	Focused generation
	Max Tokens	1000	Response length limit
	Timeout	120s	API request limit
Retrieval	Token Budget	8000	Context length limit
	Similarity	Cosine	Distance metric
	Min Threshold	0.4	Relevance filter

Table 9: Shared hyperparameters across all RAG paradigms.

Paradigm	Parameter	Value	Description
LLM-only	Temperature	0.7	Creative generation
	Max Tokens	1000	Response length
NaiveRAG	Top-K	100	Max chunks retrieved
GraphRAG	Seed Entities	20	Initial anchors
	PPR Alpha	0.85	Damping factor
	PPR Max Nodes	100	Subgraph size limit
	Max Triplets	500	Serialization limit
IterativeRAG	Base Retriever	GraphRAG	Underlying method
	Max Iterations	3	Reasoning loop limit
	Eval Temp.	0.1	Evaluator setting

Table 10: Paradigm-specific hyperparameters.

B.1 Unified Hyperparameters

To guarantee a fair comparison across distinct RAG paradigms, we standardize the core hyperparameter configurations. Table 9 enumerates the foundational settings shared across all methodologies: DeepSeek-V3 serves as the uniform generation backbone, with the temperature set to 0.3 to elicit stable outputs. All retrieval-augmented approaches are constrained by a shared context budget of 8,000 tokens, employ cosine similarity as the distance metric, and enforce a minimum relevance threshold of 0.4 to filter out low-quality evidence.

Table 10 details the paradigm-specific parameters. LLM-only, serving as the retrieval-free baseline, utilizes a higher temperature (0.7) to encourage the model to fully leverage its internal parametric knowledge. NaiveRAG retrieves a maximum of 100 text chunks, truncating the selection based on similarity ranking to fit within the token budget. GraphRAG initiates from 20 seed entities and expands the subgraph using Personalized PageRank ($\alpha = 0.85$), retaining a maximum of 100 nodes and 500 triplets. IterativeRAG employs GraphRAG as the base retriever, executing up to 3 rounds of iterative refinement, with the evaluator operating at a low temperature (0.1) to ensure decisional consistency. Finally, HybridRAG inherits the parameter settings of both NaiveRAG and GraphRAG, fusing their respective retrieval results.

B.2 Retrieval Paradigm Implementation

Table 11 contrasts the five paradigms from a methodological perspective, characterizing their retrieval substrates, information granularity, and search mechanisms. This section elaborates on the specific implementation details of each paradigm. To ensure a fair comparison, all methodologies employ a uniform generation prompt, as illustrated in Figure 16.

LLM-only and NaiveRAG. LLM-only operates as the retrieval-free baseline, generating responses

Paradigm	Retrieval Substrate	Info. Granularity	Search Mechanism	Optimal Use Cases
LLM-only	Parametric Weights (Implicit)	Internal Knowledge	Next-token Prediction	General chit-chat; Creative writing; Tasks requiring no external facts (Mallen et al., 2023).
NaiveRAG	Flat Vector Space	Coarse-grained (Passage/Chunk)	Semantic Similarity (Dense Retrieval)	Explicit fact retrieval; Simple QA; Queries with high semantic overlap (Karpukhin et al., 2020).
HybridRAG	Dual-Pathway Space (Dense + Sparse)	Multi-granular (Keyword + Chunk)	Hybrid Fusion (BM25 + Vector)	Precision-critical search; Low-frequency entity lookup; Exact matching (Thakur et al., 2021; **?**).
GraphRAG	Graph Topology (Knowledge Graph)	Fine-grained (Entity/Relation)	Structure-aware Traversal	Multi-hop reasoning (Yasunaga et al., 2021); Global summarization (Edge et al., 2024); Connecting disparate information.
IterativeRAG	Dynamic Context	Adaptive (Coarse to Fine)	Multi-step Feedback (Reasoning Loop)	Ambiguous queries; Complex research; Tasks needing progressive clarification (Trivedi et al., 2023; Asai et al., 2024).

Table 11: A Methodology Perspective on RAG Paradigms. We categorize existing paradigms by their *Retrieval Substrate* (data structure), *Information Granularity*, and *Search Mechanism*. Each paradigm imposes different trade-offs between retrieval cost and reasoning capability, highlighting that no single strategy fits all scenarios.

by directly querying the model and thereby relying exclusively on its internal parametric knowledge (Figure 15). NaiveRAG follows the standard dense retrieval protocol, retrieving the top-100 semantically similar text chunks and subsequently truncating the concatenated context to adhere to the 8,000-token budget limit.

GraphRAG. GraphRAG leverages knowledge graphs to perform structure-aware retrieval (Algorithm 3). The process initiates by extracting query entities via an LLM (Figure 13) and linking them to graph entities to establish a seed node set S. Subsequently, Personalized PageRank (PPR) is executed over the global graph topology. We construct the personalization vector $\mathbf{p}$ based on semantic similarity to the seeds:

$$\mathbf{p}[v] = \frac{\text{sim}(v)}{\sum_{u \in S} \text{sim}(u)}, \quad v \in S \tag{8}$$

The PPR iterative update rule is defined as:

$$\boldsymbol{\pi}^{(t+1)} = \alpha \cdot \mathbf{A} \cdot \boldsymbol{\pi}^{(t)} + (1-\alpha) \cdot \mathbf{p} \tag{9}$$

where $\mathbf{A}$ denotes the column-normalized adjacency matrix of the graph, and $\alpha = 0.85$ serves as the damping factor. Upon convergence, we identify the top-100 nodes with the highest PPR scores to construct a salient subgraph. The associated triplets are then extracted and mapped back to their original textual source to serve as the generation context.

HybridRAG. HybridRAG integrates the retrieval outputs from both NaiveRAG (vector-based) and GraphRAG (graph-based). Following the independent acquisition of ranked lists from both pathways, we employ Reciprocal Rank Fusion (RRF) to merge the rankings:

$$\text{RRF}(d) = \sum_{r \in \mathcal{R}} \frac{1}{k + \text{rank}_r(d)} \tag{10}$$

where $\mathcal{R} = \{\text{Naive}, \text{Graph}\}$ represents the set of retrievers, $\text{rank}_r(d)$ denotes the rank position of document d within retriever r, and $k = 60$ serves as the smoothing constant. Post-fusion, documents are sorted in descending order of their RRF scores. We subsequently remove duplicates and truncate the sequence to adhere to the strict 8,000-token context budget.

IterativeRAG. IterativeRAG implements a "Retrieve-Generate-Evaluate" feedback loop, as detailed in Algorithm 4. In the initial phase (Round 0), the system attempts a direct response using the LLM. If the evaluator (Figure 14) deems this response insufficient, it generates targeted sub-questions to trigger the retrieval cycle. In each subsequent iteration, newly retrieved evidence is aggregated with the cumulative context to synthesize an updated answer, which is then re-evaluated. This cycle persists until one of the following termination criteria is met: (i) the answer is judged sufficient; (ii) the maximum iteration count ($T = 3$) is reached; (iii) no new sub-questions are generated; or (iv) generated sub-questions are repetitive. We instantiate the framework using either NaiveRAG or GraphRAG as the underlying base retriever.

C Evaluation & Analysis Details

C.1 Metric Implementation

Metric Categories. To comprehensively assess the generation quality of RAG systems, we devise a multi-dimensional evaluation framework (Table 12). The metrics are categorized into three distinct classes: (1) Answer Quality Metrics, which quantify the semantic similarity and informational completeness of the generated response relative to the gold standard; (2) Faithfulness Metrics, which verify whether the response is strictly grounded in the retrieved context, serving as a mechanism to identify hallucinations; and (3) LLM-as-a-Judge, which provides a holistic assessment of correctness that aligns with human judgment.

Answer Quality Metrics. Semantic F1 calculates the token-level semantic similarity between the generated response and the ground truth, derived from BERTScore. Let $\hat{y} = \{\hat{x}_1, \ldots, \hat{x}_m\}$ denote the predicted answer and $y = \{x_1, \ldots, x_n\}$ denote the reference answer. We first extract contextual embeddings utilizing a pre-trained language model, specifically `microsoft/deberta-xlarge-mnli`, and subsequently compute the precision (P_{BERT}), recall (R_{BERT}), and F1 score:

$$P_{\text{BERT}} = \frac{1}{|\hat{y}|} \sum_{\hat{x}_i \in \hat{y}} \max_{x_j \in y} \cos(\mathbf{h}_{\hat{x}_i}, \mathbf{h}_{x_j}) \tag{11}$$

$$R_{\text{BERT}} = \frac{1}{|y|} \sum_{x_j \in y} \max_{\hat{x}_i \in \hat{y}} \cos(\mathbf{h}_{\hat{x}_i}, \mathbf{h}_{x_j}) \tag{12}$$

$$\text{Semantic F1} = 2 \cdot \frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{13}$$

Here, $\mathbf{h}$ represents the contextual embedding vector of a token, and $\cos(\cdot, \cdot)$ signifies cosine similarity. This metric exhibits robustness against synonym substitution and paraphrastic variations.

Soft Coverage quantifies the extent to which the generated response encapsulates the information present in the gold standard. We utilize `SentenceTransformer (all-MiniLM-L6-v2)` to segment both the reference and the prediction into individual sentences and encode them into sentence-level embeddings. For each sentence s_i^{gt} in the ground truth, we compute its maximum similarity with respect to all sentences in the prediction:

$$\text{Coverage} = \frac{1}{|S^{gt}|} \sum_{s_i^{gt} \in S^{gt}} \max_{s_j^{pred} \in S^{pred}} \cos(\mathbf{e}_{s_i^{gt}}, \mathbf{e}_{s_j^{pred}}) \tag{14}$$

where S^{gt} and S^{pred} denote the sentence sets of the ground truth and the prediction, respectively, and $\mathbf{e}$ represents the sentence embedding. A higher coverage value indicates that the generated response has successfully captured a greater proportion of the critical information contained in the reference.

Faithfulness Metrics. The Faithfulness metric evaluates whether the generated response is faithful to the retrieved context, serving as a primary mechanism for detecting hallucinations. We segment the generated response into individual sentences and calculate the semantic support for each sentence against the retrieved content.

Faithfulness (Hard) utilizes a strict threshold to determine the proportion of sentences in the answer that are supported by the retrieval:

$$\text{Faith}_{\text{hard}} = \frac{1}{|S^{ans}|} \sum_{s_i \in S^{ans}} \mathbb{1}\left[\max_{c_j \in C} \cos(\mathbf{e}_{s_i}, \mathbf{e}_{c_j}) \geq \tau\right] \tag{15}$$

where S^{ans} denotes the set of answer sentences, C denotes the set of retrieved context sentences, $\tau = 0.7$ serves as the similarity threshold, and $\mathbb{1}[\cdot]$ is the indicator function. This metric strictly quantifies the fraction of the response that possesses explicit grounding within the retrieved results.

Faithfulness (Soft). employs a continuous calculation to measure the average support strength between the answer and the retrieved content:

$$\text{Faith}_{\text{soft}} = \frac{1}{|S^{ans}|} \sum_{s_i \in S^{ans}} \max_{c_j \in C} \cos(\mathbf{e}_{s_i}, \mathbf{e}_{c_j}) \tag{16}$$

The Soft version provides a more granular measure of grounding, reflecting partial support even when the strict threshold is not met.

It is important to note that Faithfulness metrics are calculated exclusively for NaiveRAG and GraphRAG. We exclude HybridRAG and IterativeRAG from this specific evaluation, as their complex retrieval formats, involving multi-turn interactions or hybrid sources, may introduce bias into the direct calculation.

LLM-as-a-Judge. Complementing automated metrics, we employ the LLM-as-a-Judge method-

Table 12: Multi-dimensional metrics for evaluating RAG generation quality.

Category	Metric	Focus	Description & Rationale
Answer Quality	LLM-as-a-Judge	Answer Correctness	LLM classifies answers as *correct*, *incorrect*, or *incomplete*, providing human-aligned judgment.
	Semantic F1	Reference Similarity	BERTScore-based token-level semantic similarity between prediction and ground truth, robust to paraphrase variations.
	Soft Coverage	Completeness	Maximum cosine similarity between GT embedding and any prediction sentence, measuring information recall.
Grounding	Faithfulness (Hard)	Hallucination	Fraction of answer sentences with retrieval support above threshold ($\tau = 0.7$), detecting unsupported claims.
	Faithfulness (Soft)	Support Strength	Mean of max similarities between answer sentences and retrieval content, measuring grounding degree.

ology to conduct human-aligned correctness evaluation. As illustrated in Figure 17, we devise a structured evaluation prompt instructing the evaluator model (GPT-4o-mini) to classify generated responses into three mutually exclusive categories. (1) Correct: The response is logically accurate and encapsulates the core information of the ground truth. (2) Incorrect: The response contains erroneous information that contradicts the ground truth. (3)Incomplete: The response is partially correct yet lacks critical details, or the model refuses to generate an answer.

This evaluation paradigm mitigates the limitations of similarity-based metrics, which often fail to capture logical inconsistencies, thereby providing a quality assessment that more closely aligns with human judgment.

Evaluation Results. Table 13 and Table 14 present the comprehensive evaluation results for DeepSeek-V3 and Llama 3 8B across the four constituent datasets. The key findings are summarized as follows:

Overall Performance: HybridRAG achieves optimal or near-optimal performance across the majority of datasets, particularly excelling in Semantic F1 and Coverage metrics. For instance, on the Medical dataset, DeepSeek-V3 combined with HybridRAG attains an LLM accuracy of 64.7%, surpassing both NaiveRAG (61.1%) and GraphRAG (53.5%).

Model Disparity: DeepSeek-V3 significantly outperforms Llama 3 8B. Taking the MuSiQue dataset as an example, DeepSeek-V3 with HybridRAG achieves an LLM accuracy of 38.6%, whereas Llama 3 8B reaches only 20.3%, a substantial performance gap of 18.3 percentage points.

Faithfulness Analysis: Faithfulness scores for NaiveRAG and GraphRAG are generally low (mostly below 0.25), indicating that even with retrieval augmentation, models continue to generate content unsupported by the retrieved context. Conversely, the Medical dataset exhibits relatively higher faithfulness (GraphRAG reaches 0.250 Hard / 0.588 Soft), potentially attributable to the medical domain's strict reliance on retrieved factual evidence for answer formulation.

Table 15 further provides a breakdown of LLM evaluation results by query type. Key observations include:

Single-hop Dominance: All methods yield their best performance on single-hop queries. GraphRAG achieves accuracies of 90.2% (DeepSeek) and 84.4% (Llama) on MuSiQue's single-hop questions, significantly outperforming other query types. This aligns with GraphRAG's entity-based retrieval mechanism, where single-entity queries facilitate the precise localization of relevant information.

Multi-hop Challenge: Multi-hop queries prove universally challenging across all methods, with accuracy generally remaining below 35%. The Legal dataset is particularly demanding, where the best-performing method (IterativeRAG) attains only 12.4% accuracy on multi-hop tasks.

Summary Dilemma: Summary-type queries exhibit a high rate of incorrect responses (exceeding 30% for most methods on QuALITY), suggesting a model tendency to generate summaries that are either over-generalized or deviate from the source text.

C.2 Cost Calculation

Cost Components. We decompose the computational overhead of the RAG system into two primary phases (Table 16): the Retrieval phase and the Generation phase. The total cost is formally defined as:

$$C_{\text{total}} = C_{\text{retrieval}} + C_{\text{generation}} \tag{17}$$

Dataset	Method	Sem-F1	COV	Faith-H	Faith-S	LLM-Cor%
MuSiQue	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.503	0.362	0.143	0.486	26.4
	GraphRAG	0.510	0.386	0.114	0.439	30.3
	HybridRAG	0.613	0.472	-	-	38.6
	Iterative (Naive)	0.469	0.320	-	-	20.4
	Iterative (Graph)	0.000	0.400	-	-	29.2
QuALITY	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.858	0.627	0.009	0.404	48.7
	GraphRAG	0.794	0.546	0.009	0.377	39.3
	HybridRAG	0.738	0.553	0.438	0.146	41.7
	Iterative (Naive)	0.724	0.506	-	-	35.8
	Iterative (Graph)	0.657	0.446	-	-	28.7
Legal	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.568	0.469	0.145	0.537	32.2
	GraphRAG	0.530	0.443	0.094	0.510	29.3
	HybridRAG	0.617	0.520	0.589	0.326	36.1
	Iterative (Naive)	0.572	0.466	-	-	28.5
	Iterative (Graph)	0.534	0.439	-	-	26.0
Medical	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.770	0.599	0.207	0.583	61.1
	GraphRAG	0.691	0.541	0.250	0.588	53.5
	HybridRAG	0.792	0.620	0.767	0.358	64.7
	Iterative (Naive)	0.826	0.595	-	-	62.7
	Iterative (Graph)	0.801	0.575	-	-	59.8

Table 13: Evaluation results on DeepSeek-V3 across all datasets and RAG paradigms. Sem-F1: Semantic F1 (BERTScore-based), COV: Coverage, Faith-H/S: Faithfulness Hard/Soft, LLM-Cor%: LLM-as-a-Judge correct rate. "-" indicates metric not applicable or not computed.

where the cost for each phase comprises both input and output token consumption:

$$C_{\text{retrieval}} = T_{\text{ret}}^{\text{in}} + T_{\text{ret}}^{\text{out}} \tag{18}$$

$$C_{\text{generation}} = T_{\text{gen}}^{\text{in}} + T_{\text{gen}}^{\text{out}} \tag{19}$$

In the generation phase, the input token volume is predominantly governed by the aggregate context length:

$$T_{\text{gen}}^{\text{in}} \approx N \times (L_{\text{prompt}} + L_{\text{context}} + L_{\text{query}}) \tag{20}$$

where N denotes the total number of queries, L_{prompt} represents the fixed length of the system prompt, L_{context} is the average length of the retrieved context (denoted as *Avg-Ctx* in the table), and L_{query} is the query length.

For GraphRAG and HybridRAG, the retrieval cost incorporates a one-time graph construction overhead, which is amortized over the query set:

$$C_{\text{retrieval}}^{\text{graph}} = \underbrace{C_{\text{construction}}}_{\text{one-time, amortized}} + \underbrace{C_{\text{entity\_extraction}}}_{\text{per-query}} \tag{21}$$

The construction cost encompasses the input tokens required for processing the raw corpus via the LLM (T_{corpus}) and the resulting output tokens for the extracted triplets (T_{triplets}).

Method Comparison. Table 16 reveals substantial disparities in computational cost across the five distinct methodologies:

LLM-only incurs minimal overhead (0.04–0.10M tokens), as it bypasses retrieval and context injection, consuming tokens solely for the prompt, query input, and response generation.

NaiveRAG's cost footprint is predominantly driven by the generation phase ($T_{\text{gen}}^{\text{in}}$ accounts for $> 99\%$), necessitated by the inclusion of extensive retrieved chunks within the context window. For instance, on the Medical dataset, the average context length reaches 50,504 tokens, resulting in a total expenditure of 95.86M tokens.

GraphRAG generally exhibits lower generation costs compared to NaiveRAG, as graph-based retrieval yields more precise and concise contexts. On MuSiQue, GraphRAG records an Avg-Ctx of 8,472 (vs. 13,139 for NaiveRAG), translating to a total cost of 31.14M (vs. 44.22M), a reduction of approximately 30%. However, the Legal dataset presents an exception; here, GraphRAG's Avg-Ctx surges to 179,728. This anomaly arises from the dense entity interconnectivity characteristic of legal documents, where graph traversal retrieves a voluminous amount of associated content.

HybridRAG incurs the highest computational burden, as it utilizes results from both vector and

Dataset	Method	Sem-F1	COV	Faith-H	Faith-S	LLM-Cor%
MuSiQue	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.249	0.161	0.194	0.540	8.6
	GraphRAG	0.374	0.262	0.185	0.494	18.7
	HybridRAG	0.406	0.284	0.582	0.178	20.3
	Iterative (Naive)	0.289	0.169	-	-	6.8
	Iterative (Graph)	0.032	0.271	-	-	16.1
QuALITY	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.620	0.475	0.011	0.414	30.7
	GraphRAG	0.501	0.376	0.014	0.385	21.2
	HybridRAG	0.643	0.493	0.283	0.201	33.2
	Iterative (Naive)	0.583	0.435	-	-	28.4
	Iterative (Graph)	0.528	0.378	-	-	20.4
Legal	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.587	0.473	0.164	0.531	25.8
	GraphRAG	0.536	0.445	0.146	0.508	25.0
	HybridRAG	0.622	0.512	0.627	0.349	31.1
	Iterative (Naive)	0.577	0.453	-	-	22.0
	Iterative (Graph)	0.013	0.444	-	-	21.0
Medical	LLM-only	-	-	-	-	0.0
	NaiveRAG	0.732	0.574	0.175	0.575	44.9
	GraphRAG	0.673	0.532	0.214	0.572	41.7
	HybridRAG	0.759	0.600	0.813	0.374	48.2
	Iterative (Naive)	0.802	0.582	-	-	43.6
	Iterative (Graph)	0.818	0.595	-	-	43.6

Table 14: Evaluation results on Llama 3 8B across all datasets and RAG paradigms. Sem-F1: Semantic F1 (BERTScore-based), COV: Coverage, Faith-H/S: Faithfulness Hard/Soft, LLM-Cor%: LLM-as-a-Judge correct rate. “-” indicates metric not applicable or not yet computed.

graph retrieval, resulting in a context length approximating the sum of both. On the Legal dataset, it peaks at 293.10M tokens, marking the maximum consumption across all evaluated methods.

IterativeRAG exhibits a distinct cost structure characterized by high retrieval overhead (due to multi-turn LLM invocations for judgment and subquery generation) but low generation cost (owing to the refined conciseness of the context). On the Medical dataset, despite a substantial $T_{\text{ret}}^{\text{in}}$ of 8.67M, the Avg-Ctx remains merely 2,231, yielding a total cost of 13.27M, the lowest among all RAG paradigms.

Dataset Variation. The cost variations across datasets are predominantly governed by document length and corpus scale:

The Legal dataset incurs significantly higher costs compared to other benchmarks. Specifically, GraphRAG’s total expenditure on Legal (234.55M) is 7.5 times that on MuSiQue (31.14M). This disparity stems from the extensive length and complex entity interrelations inherent in legal documents, which result in: (1) elevated graph construction overhead; and (2) substantially longer contexts yielded by graph traversal (with Avg-Ctx reaching 179,728).

For the QuALITY dataset, the costs of NaiveRAG and GraphRAG are comparable (59.31M vs. 59.81M), indicating that graph-based retrieval fails to effectively reduce context length in this setting. This is attributable to QuALITY’s long-document characteristic, where each question corresponds to a complete article, resulting in high inter-chunk correlation.

On the Medical dataset, IterativeRAG demonstrates superior cost-efficiency (13.27M), amounting to merely 14% of the cost incurred by NaiveRAG (95.86M). Medical QA typically involves explicit information needs, allowing iterative retrieval to rapidly localize critical content.

Cost-Performance Trade-off. Synthesizing the cost profiles in Table 16 with the performance metrics in Table 13, we analyze the cost-performance trade-offs:

HybridRAG: High Cost, High Performance. On MuSiQue, HybridRAG attains an LLM accuracy of 38.6% at a cost of 75.22M tokens. Compared to NaiveRAG (44.22M, 26.4%), this represents a 70% cost increase yielding a 46% performance gain. Conversely, on **Medical**, HybridRAG (140.11M, 64.7%) incurs a 46% cost hike over NaiveRAG (95.86M, 61.1%) for a mere 6% performance improvement, indicating diminishing marginal returns.

GraphRAG: Dataset-Dependent Cost-Efficiency. On MuSiQue, GraphRAG (31.14M, 30.3%) delivers superior performance at a lower cost than NaiveRAG, emerging as the optimal choice. However, on QuALITY, GraphRAG (59.81M, 39.3%) offers no advantage, incurring costs comparable to NaiveRAG while yielding inferior performance (48.7%).

IterativeRAG: Low Cost, Variable Performance. On Medical, IterativeRAG (Naive-base) achieves the highest cost-effectiveness, reaching 62.7% accuracy at a minimal cost of 13.27M. Yet, on MuSiQue, the same approach yields only 20.4% accuracy, underperforming other more resource-intensive methods.

Practical Recommendations: (1) For domains with explicit entity relations (e.g., Medical), IterativeRAG offers the best cost-performance ratio; (2) For complex QA requiring the synthesis of multi-source information, HybridRAG delivers optimal performance despite its high cost; and (3) For long-document comprehension tasks (e.g., QuALITY), NaiveRAG remains the most straightforward and effective solution.

C.3 Case Studies

Comparative Analysis. Table 17 presents qualitative analysis of representative cases, comparing paradigm performance on multi-hop reasoning and cross-document summarization tasks.

Error Analysis. Drawing upon the aforementioned case studies and the comprehensive evaluation results, we categorize the primary failure modes as follows:

Retrieval Imprecision. The failure of NaiveRAG and HybridRAG in Case 1 stems from the fact that while the retrieved content possessed thematic relevance (cervical cancer surgery), it failed to precisely hit the pivotal term “trachelectomy.” This error is particularly prevalent in specialized domains (e.g., Medical, Legal), where the semantic similarity of domain-specific terminology may be lower than that of generic descriptions. GraphRAG, through its entity matching mechanism, demonstrates superior precision in localizing such specialized terms.

Context Overload. When retrieval yields a high volume of relevant yet redundant content, the LLM is prone to overlooking critical information. In Case 1, HybridRAG’s average context length (Avg-Ctx) reached 73,628 tokens. Such an excessively long context can cause the model to become “lost in the middle,” resulting in performance inferior to that of GraphRAG, which utilizes a more concise context (Avg-Ctx: 37,513).

Hallucination. In Case 2, GraphRAG generated concepts absent from the source documents (e.g., “transphasia”). This indicates that when the alignment between retrieved content and the query is poor, the model may resort to its internal parametric knowledge to “complete” the answer, leading to fabrication. This issue is particularly pronounced in Summary-type queries, as summarization necessitates cross-document synthesis which fragmented retrieval results often fail to fully cover.

Retrieval Mismatch. In Case 2, HybridRAG output “James I”, an answer entirely unrelated to spacecraft, suggesting that the retrieval module returned completely irrelevant document fragments. Such severe mismatches likely stem from a semantic gap between the query and the corpus, or from errors in entity extraction within the graph retrieval process.

Reasoning Chain Failure. For questions necessitating multi-step reasoning (e.g., the multi-hop query in Case 1), even if partially relevant content is retrieved, the model may fail to correctly link the information. IterativeRAG, via its multi-turn “Retrieve-Generate-Evaluate” loop, effectively patches the reasoning chain step-by-step, thereby exhibiting relatively stable performance on complex queries.

Triplets Extraction

- Instructions:

You are an expert Knowledge Graph Engineer. Your task is to extract structured triplets (Subject, Relation, Object) from the provided text.

1. Coreference Resolution (MUST DO): You MUST resolve pronouns (he, she, it, they, the company) to their specific canonical names.
   - BAD: ["He", "founded", "it"]
   - GOOD: ["Elon Musk", "found", "SpaceX"]
   - If a pronoun refers to an entity mentioned earlier in the text, use the Entity Name.
2. Entity Normalization:
   - Use the most complete name: "NYC" -> "New York City", "Apple" -> "Apple Inc."
   - Split compound entities: "Steve Jobs and Wozniak" -> Extract two separate triplets.
   - Keep entities atomic: Extract "New York City" not "the city of New York"
3. Relation Normalization:
   - Use active, base-form verbs: "was born in" -> "born_in", "is composed of" -> "comprise"
   - Keep relations simple and snake_case
   - Use lowercase for all relations
4. Edge Cases:
   - Multiple subjects: "Jobs and Wozniak founded Apple" -> Create two triplets
   - Nested relationships: Extract all levels (e.g., both "Paris located_in France" and "Eiffel Tower located_in Paris")
   - Negations: Include if semantically important ("does not contain", "is not part of")

- Examples:

1. Temporal + Coreference:
   - Input: "Steve Jobs co-founded Apple in 1976. He was later fired from the company but returned to lead it."
   - Output: [["Steve Jobs", "co_found", "Apple Inc."],
     ["Apple Inc.", "founded_in", "1976"],
     ["Steve Jobs", "was_fired_from", "Apple Inc."],
     ["Steve Jobs", "lead", "Apple Inc."]]
2. Pronoun Resolution:
   - Input: "The medication Aspirin is used to treat headaches. It can also reduce fever."
   - Output: [["Aspirin", "treat", "headache"],
     ["Aspirin", "reduce", "fever"]]
3. Nested Relations:
   - Input: "Paris is the capital of France and is known for the Eiffel Tower."
   - Output: [["Paris", "is_capital_of", "France"],
     ["Paris", "known_for", "Eiffel Tower"],
     ["Eiffel Tower", "locate_in", "Paris"]]
4.Multiple Entities:
   - Input: "Einstein and Bohr debated quantum mechanics in the 1920s."
   - Output: [["Albert Einstein", "debate_with", "Niels Bohr"],
     ["Albert Einstein", "research_topic", "quantum mechanics"],
     ["Niels Bohr", "research_topic", "quantum mechanics"],
     ["quantum mechanics", "debated_in", "1920s"]]

- Output Format:

Return ONLY a valid JSON list of lists. No markdown, no explanations.

Figure 9: The prompt template for extracting structured knowledge triplets from text chunks.

Algorithm 2 Query Generation & Validation Pipeline

Require: Corpus $\mathcal{C}$, Knowledge Graph $\mathcal{G}$, Target Counts $N_{\text{fact}}, N_{\text{hop}}, N_{\text{sum}}$
Ensure: Validated Query Set $\mathcal{Q}_{\text{final}}$
Hyperparameters: Generator $\mathcal{M}_{\text{gen}}$, Evaluator $\mathcal{M}_{\text{eval}}$
// Phase 1: Diversity-Driven Generation
$\mathcal{Q}_{\text{raw}} \leftarrow \emptyset$
(1) Factual Queries (Single-hop):
for $i \leftarrow 1$ **to** N_{fact} **do**
    $c \sim \text{Uniform}(\mathcal{C})$ ▷ Sample random chunk
    $(q, a) \leftarrow \mathcal{M}_{\text{gen}}(\text{Prompt}_{\text{fact}}, c)$
    $\mathcal{Q}_{\text{raw}} \leftarrow \mathcal{Q}_{\text{raw}} \cup \{(q, a, \{c\}, \text{"factual"})\}$
end for
(2) Reasoning Queries (Multi-hop):
$\mathcal{P}_{\text{chains}} \leftarrow \text{RandomWalk}(\mathcal{G}, \text{len} = 2)$ ▷ Find connected doc pairs via bridge entities
for $i \leftarrow 1$ **to** N_{hop} **do**
    $D_{\text{chain}} \leftarrow \mathcal{P}_{\text{chains}}[i]$
    $(q, a, \text{reasoning}) \leftarrow \mathcal{M}_{\text{gen}}(\text{Prompt}_{\text{hop}}, D_{\text{chain}})$
    $\mathcal{Q}_{\text{raw}} \leftarrow \mathcal{Q}_{\text{raw}} \cup \{(q, a, D_{\text{chain}}, \text{"multi\_hop"})\}$
end for
(3) Summary Queries (Global):
for $i \leftarrow 1$ **to** N_{sum} **do**
    $e \sim \text{PageRank}(\mathcal{G})$ ▷ Sample important entity
    $D_{\text{cluster}} \leftarrow \text{GetNeighbors}(e, \mathcal{G})$ ▷ Retrieve ego-graph documents
    $(q, a) \leftarrow \mathcal{M}_{\text{gen}}(\text{Prompt}_{\text{sum}}, D_{\text{cluster}})$
    $\mathcal{Q}_{\text{raw}} \leftarrow \mathcal{Q}_{\text{raw}} \cup \{(q, a, D_{\text{cluster}}, \text{"summary"})\}$
end for
// Phase 2: The "Verify-then-Filter" Validation Loop
$\mathcal{Q}_{\text{final}} \leftarrow \emptyset$
for each query instance $\mathbf{x} = (q, a, D_{\text{supp}}, \text{type}) \in \mathcal{Q}_{\text{raw}}$ **do**
    valid $\leftarrow$ **True**
    Check 1: Grounding (Answerable from Context?)
    $\hat{a} \leftarrow \mathcal{M}_{\text{eval}}(\text{Prompt}_{\text{qa}}, q, D_{\text{supp}})$
    if $\text{Sim}(\hat{a}, a) < \tau_{\text{strict}}$ **then** valid $\leftarrow$ **False**
    end if
    Check 2: Shortcut Detection (Multi-hop Only)
    if type $==$ "multi_hop" **then**
        for $d \in D_{\text{supp}}$ **do**
            if $\mathcal{M}_{\text{eval}}(q, \{d\}) \approx a$ **then** ▷ Can single doc answer it?
                valid $\leftarrow$ **False**; **break**
            end if
        end for
    end if
    Check 3: Knowledge Leakage (LLM Prior Knowledge)
    $a_{\text{prior}} \leftarrow \mathcal{M}_{\text{eval}}(\text{Prompt}_{\text{closed\_book}}, q)$ ▷ Ask without context
    if $\text{Sim}(a_{\text{prior}}, a) > \tau_{\text{leak}}$ **then**
        valid $\leftarrow$ **False** ▷ Reject if LLM already knows the answer
    end if
    if valid **then** $\mathcal{Q}_{\text{final}} \leftarrow \mathcal{Q}_{\text{final}} \cup \{\mathbf{x}\}$
    end if
end for
return $\mathcal{Q}_{\text{final}}$

Factual Query Generation

- Instructions:

1. The question should be answerable by a short phrase or sentence from the passage.
2. The answer must be explicitly stated in the passage.
3. Avoid yes/no questions.
4. The question must be understandable WITHOUT seeing the passage.
    - BAD: "When was he born?" (Who is 'he'?)
    - GOOD: "When was Barack Obama born?" (Specific Entity Name)
    - Action: You MUST replace pronouns (he, she, it, they) with the actual entity names found in the text.

- Output Format:

{"question": "The self-contained question", "answer": "The extractable answer"}

Figure 10: The prompt template for generating Factual queries.

Reasoning Query Generation

- Instructions:

Generate a strict {num_hops}-hop question based on the provided {num_docs}-document chain.

- Generation Guidelines:

1. Analyze: Identify the Start Entity (Doc 1), Bridge Entities (Intermediate), and Final Answer (Last Doc).
2. Recursive Masking: Starting from the end, replace every Bridge Entity's name with a functional description derived strictly from the previous document.
    - Example: Instead of "Hawaii", use "the location where Jurassic Park was filmed".
3. Assembly: The final question must nest these descriptions and contain ONLY the specific name of the Start Entity.

- CRITICAL CONSTRAINTS:

1. Forbidden: NEVER use the actual name of any Bridge Entity in the question.
2. Dependency: The question must become unanswerable if any single document is removed.
3. Answer Isolation: The Final Answer must NOT appear in any document except the last one.
4. Grammar: Ensure natural phrasing for nested structures.

- Output Format:

```
{{
  "reasoning": "Step 1: Identify Bridge [Name] and mask it as '[Description]'; Step 2: Identify Bridge [Name] and mask it as '[Description]'; Step 3: Verify strict dependency.",
  "question": "The final nested question",
  "answer": "The final answer from the last Doc"
}}
```

Figure 11: The instruction set for synthesizing Reasoning queries.

Summary Query Generation

- Instructions:
You are an expert dataset creator.
Your task is to generate a Synthesis Question and a Comprehensive Answer based on a cluster of documents related to the entity: "{entity}".

- Generation Guidelines:
1. Consistency Check: Do these documents talk about the SAME "{entity}"?
   - If one doc is about "Michael Jordan (Basketball)" and another about "Michael Jordan (Professor)", return {{"discard": true}}.
2. Information Synthesis (The Answer):
   - Identify common themes across the documents (e.g., Career, Impact, Controversy, Evolution).
   - Write a high-quality summary (3-5 sentences) that integrates facts from AT LEAST 2 different documents.
   - Constraint: Do not simply list facts. Connect them (e.g., "While Doc A mentions X, Doc B clarifies that Y...").
3. Question Formulation (The Query):
   - Write a question that naturally elicits the summary you just wrote.
   - Specific Theme: Instead of "Summarize X", ask about a specific aspect like "How did X's career evolve?" or "What are the key contributions of X?".
   - Self-Contained: The question MUST contain the full name of "{entity}". Do NOT use pronouns like "he/she" or vague phrases like "the provided text".
     - BAD: "Summarize his achievements mentioned in the text."
     - GOOD: "What were the major scientific achievements of Albert Einstein?"

- Output Format:
{{ "reasoning": "Step 1: Themes identified are... Step 2: Documents used are Doc [x] and Doc [y]...",
 "question": "The self-contained synthesis question",
 "answer": "The synthesized answer covering multiple docs"
}} OR {{"discard": true}}

Figure 12: The prompt designed for Summary query generation.

Entity Extraction

- Instructions:
You are an expert at extracting named entities from text.
1. Identify all entities: people, organizations, locations, dates, events, concepts
2. Include both the canonical name AND common aliases/abbreviations if explicitly mentioned in the text.
   - Include aliases only if they refer to the same real-world entity.
   - Do NOT invent aliases that are not explicitly mentioned.
3. Resolve pronouns to their referent entity (he/she/it → the entity they refer to)
4. Exclude generic pronouns (it, this, that) that don't refer to a specific named entity.
5. Exclude common stop words and articles.

- Examples:
1.Text: "Tony Stark founded Apple. He later sold the company. NYC became its headquarters."
   Output: ["Tony Stark", "Apple", "NYC"]
2.Text: "Apple Inc., commonly known as Apple, was founded by Steve Jobs. The company is based in NYC (New York City)."
   Output: ["Apple Inc.", "Apple", "Steve Jobs", "NYC", "New York City"]

- Output Format:
Return ONLY a JSON list of entities:
["entity1", "entity2", ...]

Figure 13: The prompt template for Entity Extraction.

Algorithm 3 GraphRAG Retrieval & Generation Pipeline

Require: Question q, Knowledge Graph $G = (V, E)$, Entity Index $\mathcal{I}_E$, Token Budget B
Ensure: Generated Answer a
Hyperparameters: Entity Threshold $\tau_{\text{entity}} = 0.4$, PPR Threshold $\tau_{\text{ppr}} = 1e^{-5}$, Damping $\alpha = 0.85$

// Phase 1: Seed Entity Retrieval
$E_{\text{query}} \leftarrow \text{LLM}(\text{Prompt}_{\text{extract}}, q)$
if $E_{\text{query}} = \emptyset$ **then**
    $E_{\text{query}} \leftarrow \{q\}$ ▷ Fallback: use entire question
end if
$S \leftarrow \emptyset$ ▷ Initialize seed set mapping: $id \rightarrow score$
for each $e \in E_{\text{query}}$ **do**
    $v_e \leftarrow \text{Embed}(e)$
    $\mathcal{K} \leftarrow \text{FAISS\_Search}(\mathcal{I}_E, v_e, k = 20)$
    for each $(id, \text{sim}) \in \mathcal{K}$ **do**
        if $\text{sim} > \tau_{\text{entity}}$ **then**
            $S[id] \leftarrow \max(S[id], \text{sim})$ ▷ Max pooling for duplicates
        end if
    end for
end for
$S \leftarrow \text{TopK}(S, k = 20)$

// Phase 2: PPR-Based Subgraph Expansion
$\mathbf{p} \leftarrow \text{Zeros}(|V|)$ ▷ Initialize personalization vector
$Z \leftarrow \sum_{(id,\text{sim}) \in S} \text{sim}$
for each $(id, \text{sim}) \in S$ **do**
    $\mathbf{p}[id] \leftarrow \text{sim}/Z$ ▷ Normalize to probability distribution
end for
$\boldsymbol{\pi} \leftarrow \text{PageRank}(G, \text{personalization} = \mathbf{p}, \alpha = \alpha, \text{iter} = 100)$
$V_{\text{sub}} \leftarrow \{v \mid \boldsymbol{\pi}[v] \geq \tau_{\text{ppr}}\}$
$V_{\text{sub}} \leftarrow \text{TopK}(V_{\text{sub}}, k = 100) \cup \text{Keys}(S)$ ▷ Keep top-100 expanded nodes + seeds

// Phase 3: Context Construction
$T \leftarrow \{(u, r, v) \mid u, v \in V_{\text{sub}}, (u, v) \in E\}$ ▷ Extract triplets from induced subgraph
$\mathcal{C} \leftarrow \emptyset$
Sort T by $\max(\text{sim}(u), \text{sim}(v))$ descending ▷ Prioritize relevance
$ctx \leftarrow$ "", count $\leftarrow 0$
for each $(u, r, v) \in T$ **do**
    sents $\leftarrow$ TripletSourceMap$[(u, r, v)]$
    for each $s \in$ sents **do**
        if count $+$ Len$(s) > B$ **then break**
        end if
        $ctx \leftarrow ctx \oplus s$
        count $\leftarrow$ count $+$ Len(s)
    end for
end for

// Phase 4: Answer Generation
$a \leftarrow \text{LLM}(\text{Prompt}_{\text{RAG}}, ctx, q)$
return a

Algorithm 4 IterativeRAG (Multi-Round Retrieval with Self-Evaluation)

Require: Question q, Base Retriever $\mathcal{R}$ (NaiveRAG or GraphRAG), Max Iterations T
Ensure: Final Answer a

Initialization
$\mathcal{Q}_{\text{history}} \leftarrow \{q\}$ ▷ Track all queries to prevent loops
$\mathcal{C}_{\text{accum}} \leftarrow \emptyset$ ▷ Accumulated retrieved chunks
$H \leftarrow []$ ▷ Reasoning trace

Round 0: Direct LLM Answer (No Retrieval)
$a_0 \leftarrow$ LLM(Prompt$_{\text{QA}}$, q) ▷ Answer without context
eval$_0 \leftarrow$ LLM(Prompt$_{\text{Eval}}$, q, a_0) ▷ {sufficient, reason, sub_question}
H.append($(0, q, a_0, \text{eval}_0)$)
if eval$_0$.sufficient **is True then**
  return a_0 ▷ LLM already knows the answer
end if
$q_{\text{curr}} \leftarrow$ eval$_0$.sub_question **or** q ▷ Get refined query

Round 1+: Iterative Retrieval Loop
for $t \leftarrow 1$ **to** T **do**
  // Step 1: Retrieve new chunks
  $\mathcal{C}_{\text{new}} \leftarrow$ Retrieve($\mathcal{R}$, q_{curr})
  // Step 2: Merge and Deduplicate
  $\mathcal{C}_{\text{accum}} \leftarrow \mathcal{C}_{\text{accum}} \cup \mathcal{C}_{\text{new}}$
  // Step 3: Apply Token Budget
  $\mathcal{C}_{\text{ctx}} \leftarrow$ ApplyTokenBudget($\mathcal{C}_{\text{accum}}$, $B = 8000$)
  ctx $\leftarrow$ Concatenate($\mathcal{C}_{\text{ctx}}$)
  // Step 4: Generate answer with accumulated context
  $a_t \leftarrow$ LLM(Prompt$_{\text{RAG}}$, ctx, q) ▷ Always answer ORIGINAL question
  // Step 5: Evaluate answer sufficiency
  eval$_t \leftarrow$ LLM(Prompt$_{\text{Eval}}$, q, a_t, ctx)
  H.append($(t, q_{\text{curr}}, a_t, \text{eval}_t)$)
  // Step 6: Check termination conditions
  if eval$_t$.sufficient **is True then**
    return a_t ▷ Answer is sufficient
  end if
  $q_{\text{next}} \leftarrow$ eval$_t$.sub_question
  if q_{next} **is null then**
    return a_t ▷ No further refinement possible
  end if
  if $q_{\text{next}} \in \mathcal{Q}_{\text{history}}$ **then**
    return a_t ▷ Prevent query loop
  end if
  // Step 7: Update for next iteration
  $\mathcal{Q}_{\text{history}} \leftarrow \mathcal{Q}_{\text{history}} \cup \{q_{\text{next}}\}$
  $q_{\text{curr}} \leftarrow q_{\text{next}}$
end for
return $H[-1]$.answer ▷ Return last answer if max iterations reached

IterativeRAG Evaluation

- Instructions:

You are an answer evaluation assistant. Assess whether the answer adequately addresses the question.
Evaluate the given answer and determine if it sufficiently answers the question.
1. Set "sufficient" to true if the answer correctly and completely addresses the question
2. Set "sufficient" to false if the answer is wrong, incomplete, or refuses to answer (e.g., "I cannot answer")
3. If insufficient, generate a "sub_question" that targets the specific missing information needed to answer the original question
4. The "sub_question" should be a focused query suitable for document retrieval
5. If sufficient, set "sub_question" to null

- Output Format:

Respond ONLY in JSON with exactly these fields:
{"sufficient": true/false, "reason": "brief explanation", "sub_question": "follow-up question if insufficient, else null"}

Figure 14: The Self-Evaluation prompt for Iterative RAG.

LLM-only Generation

- Instructions:
You are a helpful AI assistant evaluating your internal knowledge base.
Your task is to answer the user's question based PURELY on your pre-trained knowledge.
1. Internal Knowledge: Use your own memory to answer. Do not ask for context.
2. Conciseness: Provide the answer directly and concisely.
 - If the answer is an entity, date, or number, output ONLY that specific string.
 - Do not output full sentences like "The answer is..." unless necessary.
3. Honesty: If you do not know the answer or are unsure, reply EXACTLY with: "I cannot answer" (no explanation, no reasoning).
4. Do NOT hallucinate or make up facts.

- Examples:
1. Q: "What is the capital of France?"
 A: Paris
2. Q: "Who is the director of the movie starring the lead actor of Titanic?"
 A: Christopher Nolan
3. Q: "What is the specific diameter of the hidden pipe in the fictional factory of generic novel X?"
 A: I cannot answer

Figure 15: The prompt for Direct Generation (LLM-only).

Retrieval Answer Generation

- Instructions:
You are a strict and accurate Answer Generator based on retrieved context.
1. Context-Driven: Answer the question using ONLY the provided context snippets. Do not use your own internal knowledge.
2. Refusal: If the provided context does not contain the information needed to answer the question, output EXACTLY: "I cannot answer" (no explanation, no reasoning).
3. No Meta-talk: Do not say "According to the context..." or "The document says...". Just give the answer.

- Adaptive Answer Format:
1. Specific Fact Extraction (e.g., Who, When, Where)
 - Goal: Single-hop / Factual
 - Format: Provide the precise entity, date, number, or name. Be atomic and concise.
 - Example: "Steve Jobs" (NOT "Steve Jobs is the CEO.")
2. Complex Reasoning (e.g., Comparative, Multi-step)
 - Goal: Multi-hop
 - Format: Perform the reasoning across the context and output the direct conclusion.
 - Example: "Paris" (If asked for the capital of the country where X was born)
3. Overview or Explanation (e.g., Summarize, How, Why)
 - Goal: Summarization
 - Format: Synthesize information from multiple parts of the context into a coherent, comprehensive paragraph (3-5 sentences). Do not simply list bullet points unless requested.

- Final Check:
1. Does your answer directly address the prompt?
2. Is it fully supported by the text?

Figure 16: The Context-Aware Generation prompt.

LLM-as-a-Judge

- Instructions:

You are evaluating final answer correctness.
1. Do NOT consider writing quality, fluency, or style.
2. Focus only on whether the final conclusion is correct.
3. Treat the final conclusion as the specific answer that would fully resolve the question.
4. If the answer is partially correct, vague, or missing key information, choose incomplete.
5. If the answer gives a specific wrong conclusion, choose incorrect.
6. If the answer refuses to answer (e.g., "I cannot answer"), choose incomplete.
7. Classify the predicted answer into ONE of the following categories:
    - correct: The answer gives the correct final conclusion required by the ground truth.
    - incorrect: The answer gives a wrong final conclusion that conflicts with the ground truth.
    - incomplete: The answer attempts to answer but does not give the correct final conclusion (includes refusal to answer).

- Examples:

1. GT: "Paris", Pred: "The capital of France is Paris" → correct
2. GT: "1976", Pred: "Apple was founded in 1975" → incorrect
3. GT: "Steve Jobs", Pred: "I cannot answer based on the context" → incomplete

- Output Format:

Respond ONLY in JSON with exactly these fields:
{"label": "<correct/incorrect/incomplete>", "reason": "<one short sentence explaining why>"}

Figure 17: The LLM-as-a-Judge instruction template used for automated evaluation.

Method	Query Type	N	DeepSeek-V3			Llama-3-8B		
			Cor%	Inc%	No-A%	Cor%	Inc%	No-A%
Dataset: MuSiQue								
NaiveRAG	Multi-hop	2590	28.3	24.0	47.7	7.9	9.3	82.7
	Single-hop	398	11.1	5.3	83.7	10.3	5.8	83.9
	Summary	368	29.4	27.5	43.2	12.0	47.8	40.2
GraphRAG	Multi-hop	2590	22.5	20.6	56.9	9.7	13.6	76.7
	Single-hop	398	**90.2**	0.5	9.3	**84.4**	1.3	14.3
	Summary	368	20.7	20.4	59.0	11.4	37.5	51.1
HybridRAG	Multi-hop	2590	32.8	22.7	44.5	12.1	14.9	73.0
	Single-hop	398	83.7	1.5	14.8	79.9	2.3	17.8
	Summary	368	30.2	29.6	40.2	13.9	48.6	37.5
IterativeRAG	Multi-hop	2590	21.8	21.9	56.3	-	-	-
	Single-hop	398	10.3	24.1	65.6	-	-	-
	Summary	368	21.2	34.2	44.6	-	-	-
Dataset: QuALITY								
NaiveRAG	Multi-hop	461	33.8	48.8	17.4	10.9	20.4	68.8
	Single-hop	454	**83.7**	13.2	3.1	**69.2**	13.9	17.0
	Summary	283	17.0	30.7	52.3	1.4	65.4	33.2
GraphRAG	Multi-hop	461	20.4	49.7	29.9	9.5	12.8	77.7
	Single-hop	454	70.7	19.8	9.5	44.7	13.0	42.3
	Summary	283	19.8	24.0	56.2	2.5	67.8	29.7
HybridRAG	Multi-hop	461	20.4	32.1	47.5	15.6	21.5	62.9
	Single-hop	454	80.0	13.7	6.4	70.3	12.1	17.6
	Summary	283	14.8	35.3	49.8	2.5	68.6	29.0
IterativeRAG	Multi-hop	461	17.1	37.3	45.6	-	-	-
	Single-hop	454	67.0	24.5	8.6	-	-	-
	Summary	283	16.3	38.2	45.6	-	-	-
Dataset: Legal								
NaiveRAG	Multi-hop	526	11.2	13.3	75.5	10.7	21.3	68.1
	Single-hop	370	54.9	17.3	27.8	50.3	24.6	25.1
	Summary	381	39.1	11.6	49.3	22.8	27.8	49.3
GraphRAG	Multi-hop	526	6.7	10.7	82.7	7.4	12.2	80.4
	Single-hop	370	61.6	13.5	24.9	55.4	15.7	28.9
	Summary	381	29.1	8.4	62.5	19.7	26.5	53.8
HybridRAG	Multi-hop	526	11.8	14.8	73.4	13.1	19.0	67.9
	Single-hop	370	**72.2**	15.1	12.7	**63.8**	20.8	15.4
	Summary	381	34.7	13.1	52.2	24.2	27.0	48.8
IterativeRAG	Multi-hop	526	12.4	13.9	73.8	-	-	-
	Single-hop	370	49.5	24.6	25.9	-	-	-
	Summary	381	30.5	10.8	58.8	-	-	-
Dataset: Medical								
NaiveRAG	Multi-hop	509	63.5	5.1	31.4	37.9	8.1	54.0
	Single-hop	1098	63.1	6.2	30.7	52.1	12.3	35.6
	Summary	289	49.1	1.0	49.8	30.1	14.5	55.4
GraphRAG	Multi-hop	509	56.0	3.9	40.1	33.6	6.3	60.1
	Single-hop	1098	55.0	6.2	38.8	48.1	8.5	43.4
	Summary	289	43.6	2.4	54.0	31.8	9.7	58.5
HybridRAG	Multi-hop	509	64.1	4.1	31.8	39.9	6.1	54.0
	Single-hop	1098	**67.8**	5.7	26.5	**55.7**	11.2	33.1
	Summary	289	54.0	1.7	44.3	33.9	10.4	55.7
IterativeRAG	Multi-hop	509	**67.8**	4.3	27.9	-	-	-
	Single-hop	1098	62.1	12.5	25.4	-	-	-
	Summary	289	**56.1**	3.8	40.1	-	-	-

Table 15: Comprehensive LLM-as-a-Judge evaluation comparison between DeepSeek-V3 and Llama-3-8B across four datasets. Results report Accuracy (Cor), Incorrectness (Inc), and No-Answer rates (No-A).

Dataset	Method	N	Avg-Ctx	Ret-In	Ret-Out	Gen-In	Gen-Out	Total
MuSiQue	LLM-only	3,356	0	0	0	0.07	0.03	0.10
	NaiveRAG	3,356	13,139	0	0	44.17	0.05	44.22
	GraphRAG	3,356	8,472	2.35	0.24	28.51	0.04	31.14
	HybridRAG	3,356	21,602	2.35	0.24	72.57	0.06	75.22
	IterativeRAG	3,357	6,364	43.24	0.72	21.44	0.05	65.46
QuALITY	LLM-only	1,198	0	0	0	0.03	0.00	0.04
	NaiveRAG	1,198	49,444	0	0	59.27	0.04	59.31
	GraphRAG	1,198	48,502	1.56	0.07	58.14	0.04	59.81
	HybridRAG	1,198	97,789	1.56	0.07	117.19	0.03	118.85
	IterativeRAG	1,198	6,871	16.69	0.25	8.27	0.03	25.24
Legal	LLM-only	1,277	0	0	0	0.05	0.01	0.06
	NaiveRAG	1,277	46,273	0	0	59.14	0.05	59.19
	GraphRAG	1,277	179,728	4.82	0.13	229.56	0.04	234.55
	HybridRAG	1,277	225,571	4.82	0.13	288.10	0.05	293.10
	IterativeRAG	1,278	6,460	16.81	0.29	8.30	0.05	25.45
Medical	LLM-only	1,896	0	0	0	0.03	0.06	0.09
	NaiveRAG	1,896	50,504	0	0	95.78	0.08	95.86
	GraphRAG	1,896	37,513	0.25	0.15	71.15	0.07	71.63
	HybridRAG	1,896	73,628	0.25	0.15	139.63	0.08	140.11
	IterativeRAG	1,897	2,231	8.67	0.26	4.26	0.08	13.27

Table 16: Token consumption breakdown for retrieval and generation across all datasets and methods. All token counts are in millions (M) except Avg-Ctx (average context tokens per question). Ret: Retrieval, Gen: Generation, In: Input, Out: Output.

Case ID	Query & Gold Standard	Paradigm Comparison	Key Analysis
Case 1 *Medical 1244* (Multi-hop)	**Q:** What are the surgical options for early cervical cancer and how do they relate to fertility preservation? **Gold:** Cone biopsy and trachelectomy are surgical options for early-stage disease, with trachelectomy being a fertility-sparing procedure.	**GraphRAG (Correct):** Explicitly states that "fertility-sparing surgical options include cone biopsy or radical trachelectomy," accurately capturing both procedures. **NaiveRAG (Incomplete):** Mentions cone biopsy but fails to explicitly cite "trachelectomy," offering generic "hysterectomy types." **HybridRAG (Incomplete):** Similarly omits "trachelectomy," referencing only distinct hysterectomy types. **IterativeRAG (Correct):** Successfully localizes "cone biopsy, trachelectomy" through iterative retrieval, fully covering the gold answer.	This case demonstrates that for multi-hop queries necessitating precise medical terminology, GraphRAG's entity-oriented retrieval and IterativeRAG's multi-turn refinement are effective. In contrast, semantic-based methods (Naive/Hybrid) lack the required precision.
Case 2 *QuALITY 1034* (Summary)	**Q:** Based on the narrative, what are the common types of extreme adversity faced by spacecraft? What are the crew's survival strategies? **Gold:** Scenarios include physical trauma (crash/flip), atmospheric entry, and combat. Reactions involve assessing reparability or risking hyperdrive escape.	**GraphRAG (Incorrect):** Hallucinates concepts like "transphasia" and "space cafard" absent from the source text. **NaiveRAG (Incorrect):** Lists generic tropes like "sabotage, alien attacks" without addressing specific scenarios. **HybridRAG (Incorrect):** Erroneously outputs "James I," suggesting irrelevant retrieval. **IterativeRAG (Incomplete):** Closest result, citing "mechanical failures" and "damaged hulls," capturing the thematic direction but missing details.	This highlights the challenge of Summary-type queries requiring cross-document synthesis. Even IterativeRAG only achieves Incomplete status, indicating significant room for improvement in long-document summarization tasks.

Table 17: Qualitative analysis of two representative cases.