MacRAG: Compress, Slice, and Scale-up for Multi-scale Adaptive Context RAG

Anonymous ACL submission

Abstract

Long-context large language models (LC LLMs) combined with retrieval-augmented generation (RAG) hold strong potential for complex multi-hop and large-document tasks. However, existing RAG systems often suffer from imprecise retrieval, incomplete context coverage under constrained windows, and fragmented information from suboptimal context construction. We introduce **Multi-scale Adaptive Context RAG (MacRAG)**, a hierarchical RAG framework that compresses and partitions documents into coarse-to-fine granularities, then adaptively merges relevant contexts through real-time chunk- and document-level expansions. By initiating with finest-level retrieval and progressively incorporating broader, higher-level context, MacRAG constructs effective query-specific long contexts, optimizing both precision and coverage. Evaluations on challenging LongBench expansions of HotpotQA, 2WikiMultihopQA, and Musique confirm MacRAG consistently surpasses baseline RAG pipelines in single- and multi-step generation using Llama-3.1-8B, Gemini-1.5-pro, and GPT-4o. Our results establish MacRAG as an efficient, scalable solution for real-world long-context, multi-hop reasoning.

1 Introduction

Large language models (LLMs) have significantly advanced complex reasoning, but they still suffer from factual gaps or hallucinations when relying only on internal parameters (Zhao et al., 2024). Retrieval-Augmented Generation (RAG) addresses this by grounding LLMs in external evidence (Guu et al., 2020; Lewis et al., 2020). Long-context (LC) LLMs such as GPT-4o (OpenAI, 2024), Gemini 1.5 (Team et al., 2024), and Llama 3 (Dubey et al., 2024a) offer large input windows but remain limited. They often miss crucial mid-context information (Liu et al., 2024; Xu et al., 2024b) and their performance degrades at extreme context lengths (Leng et al., 2024; Yu et al., 2024).

These issues in RAG systems give rise to three key trade-offs: (1) Context Length vs. Focus, where longer contexts improve recall but may obscure important details, while shorter ones enhance focus but risk omission of essential evidence; (2) Chunking and Indexing, where fine-grained chunks boost retrieval precision but harm coherence, while coarse chunks preserve structure but introduce redundancy; and (3) Coverage vs. Computation, where broader context improves reasoning but increases token cost and latency, especially in iterative or agentic pipelines (Yue et al., 2024; Asai et al., 2024).

To address these interconnected trade-offs systematically, we propose Multi-scale Adaptive Context RAG (MacRAG). MacRAG integrates top-down offline indexing with bottom-up query-time adaptive retrieval. Offline, documents are partitioned into overlapping chunks, their content compressed via abstractive summarization, and these summaries are further sliced for fine-grained indexing. At query time, MacRAG retrieves precise slices, then adaptively reconstructs the context by merging these into parent chunks, incorporating neighboring chunks, and performing document-level expansions. This constructs effective, query-specific, and length-bounded long contexts, optimizing the balance between precision, coverage, and efficiency.

MacRAG's unified approach combines structure-preserving indexing, adaptive multi-scale retrieval, and bounded context assembly, offering a distinct, robust solution for complex multi-hop reasoning. Extensive experiments on challenging LongBench (Bai et al., 2023) datasets show significant gains over strong baselines using Llama-3.1-8B, Gemini-1.5-pro, and GPT-4o. This paper thus introduces a novel multi-scale retrieval architecture, empirically validates its significant benefits for demanding QA tasks, and presents an efficient, modular framework for advanced RAG applications.

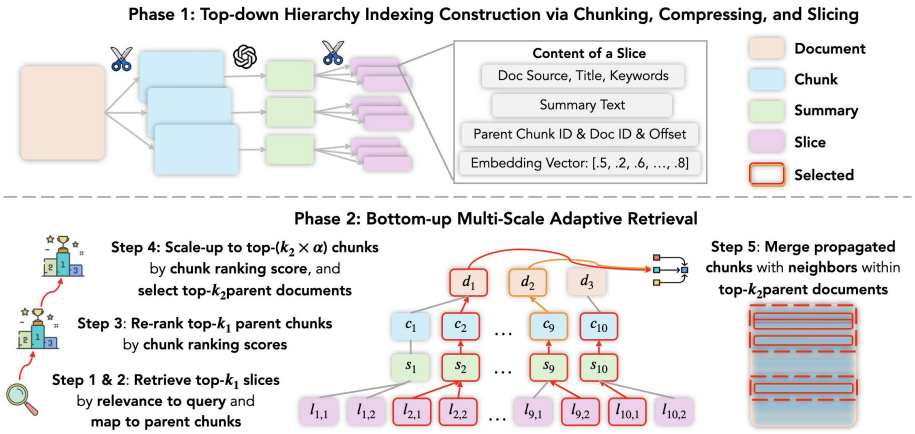

Figure 1: An overview of the MacRAG framework, consisting of two main phases: (1) top-down hierarchical indexing (upper), and (2) bottom-up multi-scale adaptive retrieval on the constructed hierarchy of document-chunk-summary-slice (lower).

2 Related Work

Strategies for RAG with Long Context (LC) LLMs (OpenAI, 2024; Team et al., 2024; Dubey et al., 2024a) largely follow two directions. Post-retrieval context management aims to condense information after initial retrieval by employing methods such as abstractive summarization like RECOMP-Abst (Xu et al., 2024a) or extractive techniques including LLMLingua (Jiang et al., 2023) and RECOMP-Extr (Xu et al., 2024a). While these manage context size, they are bottlenecked by initial retrieval quality, risk information loss, and can be computationally intensive. MacRAG, by contrast, distinctly improves the retrieval phase itself, proactively constructing a high-quality, bounded context.

Orthogonally, hierarchical retrieval approaches organize information during retrieval and ranking for broader coverage. Systems like GraphRAG (Edge et al., 2024) and HippoRAG (Gutiérrez et al., 2024) use symbolic graphs or sentence-level indexing, which can improve recall but often increase the overhead. RAPTOR (Sarthi et al., 2024) recursively summarizes clustered text chunks, potentially missing non-semantic relations, while SIRERAG (Zhang et al., 2025) further integrates relational connectivity at higher computational cost. LongRAG (Zhao et al., 2024) utilizes entire parent documents of top-ranked chunks alongside a multi-step generation scheme, a strategy that can be inefficient with very long document contexts.

Based on Zhang et al. (2025)'s comparative analysis (Table 4 therein), which shows that GraphRAG underperforms on multi-hop QA while RAPTOR, HippoRAG, and SIRERAG attain similar F1-scores, we focus our evaluations on RAPTOR and LongRAG. Although RAPTOR's semantic clustering may fragment knowledge and LongRAG's full-document usage can be costly, both were selected for their promising trade-offs between strong performance and relatively lower overhead compared to other competitive methods, such as the noted underperformance of GraphRAG or the higher operational costs of SIRERAG.

In contrast, MacRAG preserves original document structure through its offline hierarchical indexing and adaptively merges relevant and related contexts via its multi-scale retrieval and ranking at query time. This approach circumvents costly, repeated clustering and the need for explicit symbolic graphs, offering flexible assembly of multi-hop contexts with minimal overhead, and is designed with extendibility towards graph-based enhancements.

3 Multi-scale Adaptive Context RAG

We propose **Multi-scale Adaptive Context RAG (MacRAG)**, a hierarchical multi-scale adaptive retrieval system with three sequential components. First, **top-down hierarchical indexing** compresses documents, partitions them into chunks, and slices them to build a hierarchical index from coarse-grained documents to fine-grained slices with multi-level indices. Second, **bottom-up multi-scale adaptive retrieval** starts from the finest granularity to ensure precision, and progressively expands to broader contexts by merging relevant information to construct **query-specific long contexts dynamically** in real time. Third, the **response generation** leverages these carefully assembled contexts to produce the answer.

Algorithm 1 MacRAG: Multi-scale Adaptive Context RAG

Require: Query q, document corpus $\mathcal{D}$, k_1 (for slice retrieval), k_2 (for final merged chunks), the # of hops h, up-scaling factor α

Ensure: Top-k_2 merged chunks $\mathcal{C}_{\text{final}}$, generated answer

Phase 1: Top-down Hierarchical Indexing

For each document $d \in \mathcal{D}$:

(1.1) Split d into a set of overlapping *chunks* $\mathcal{C}_d$.

(1.2) *Compress* each chunk c_i into a shorter summary s_i via summarization.

(1.3) Split each compressed summary s_i into overlapping *slices* $\mathcal{L}_i$.

(1.4) Encode each slice and store its embedding & metadata (doc ID, chunk ID, offset, etc) in the database.

Phase 2: Bottom-up Multi-scale Adaptive Retrieval and Ranking

(2.1) Retrieve the top-k_1 slices based on similarity to q.

(2.2) Map retrieved slices to their parent chunks to obtain $\mathcal{C}'_{k_1}$.

(2.3) *Rerank* chunks in $\mathcal{C}'_{k_1}$ to refine their scores r'_{q,c_i}.

(2.4) *Scale-up* by selecting the top $(k_2 \times \alpha)$ chunks and ranking their source documents by aggregated scores to choose k_2 distinct documents.

(2.5) *Merge* the top-ranked chunks' h-hop neighbors to form the final top-k_2 chunks, $\mathcal{C}_{\text{final}}$.

Phase 3: Response Generation

Provide $\mathcal{C}_{\text{final}}$ as context to an LLM (or any downstream model) to generate the final answer.

Our main contribution lies in the retrieval, ranking, and adaptive input context construction pipeline (phases 1 and 2), which serves as a modular foundation for integrating any response generation method in phase 3. Figure 1 provides an overview of MacRAG, and Algorithm 1 outlines the methods along with equations in subsequent sections.

3.1 Document Hierarchical Indexing

Given a document corpus $\mathcal{D} = \{d_1, \ldots, d_N\}$, MacRAG performs offline processing to construct a hierarchy through chunking (§3.1.1), compression (§3.1.2), and slicing (§3.1.3), while building a *slice-level* vector DB at the finest granularity to empower precise retrieval. This strategy aims to reduce redundancy and sharpen semantic focus, thereby enhancing retrieval precision and minimizing noise during the subsequent multi-scale expansion phase.

3.1.1 Chunking

Each document $d \in \mathcal{D}$ is split into partially overlapping chunks, $\mathcal{C}_d = \{c_1, c_2, \ldots, c_m\}$, where each chunk c_i spans approximately 200 to 500 tokens, with a fixed overlap of 10–200 tokens between consecutive chunks. This controlled overlap preserves semantic continuity across chunk boundaries. Although the chunking is token-based by default, it can alternatively be configured at the character level. We applied the similar range of chunk sizes used by LongRAG (Zhao et al., 2024), more details in Appendix A.5.

3.1.2 Compression

Each chunk c_i is transformed into a more compressed summary s_i, yielding a set of summaries $\mathcal{S}_d = \{s_1, \ldots, s_m\}$, where $s_i = \text{Compress}(c_i)$. The $\text{Compress}(\cdot)$ represents a summarization model. It can be either extractive (selecting salient sentences) or abstractive (using a generative model to generate a summary). By default, we use an LLM to generate the abstractive summarization. This step effectively reduces redundancy while retaining core factual information.

3.1.3 Slicing

To facilitate finer-grained retrieval, each summary s_i is further split into overlapping slices: $\mathcal{L}_i = \{\ell_{i,1}, \ell_{i,2}, \ldots, \ell_{i,j}, \ldots\}$, where each slice $\ell_{i,j}$ typically spans 50-200 tokens with partial overlap as in the standard overlap chunking, and our parameter details are in Appendix A.5. Each slice is then embedded via $\mathbf{e}_{i,j} = \text{Embed}(\ell_{i,j})$, where $\text{Embed}(\cdot)$ represents a text encoder (*e.g.*, `multilingual-e5-large`). The slice-level embeddings with metadata (document ID, chunk ID, offsets) are indexed in the vector DB. MacRAG can optionally index chunk and summary embeddings for multi-view retrieval.

3.2 Multi-scale Adaptive Retrieval and Ranking

Given an input query q, MacRAG implements a multi-scale adaptive retrieval pipeline over the constructed hierarchy through five sequential steps, ultimately selecting the top-k_2 merged chunks with contextually relevant neighbor expansions for enhanced both precision and recall.

3.2.1 Initial Slice-level Retrieval

With slices serving as the finest granular units in our hierarchical system and vector DB, we first compute the query-slice relevance scores for each slice $\ell_{i,j}$ in the vector DB: $r_{q,\ell_{i,j}} = \text{Rel}(q, \ell_{i,j})$, where $\text{Rel}(\cdot,\cdot)$ can be implemented with relevance model, *e.g.*, cosine similarity and L_2 distance in dense retrieval, sparse retrieval, hybrid retrieval, and so on. We then retrieve the top-k_1 most relevant slices. This fine-grained retrieval ensures high precision by identifying slices with strong semantic alignment to the query q.

3.2.2 Parent Chunk Mapping

Each retrieved slice corresponds to a parent chunk c_i. To prevent redundancy when multiple retrieved slices originate from the same chunk, we perform a unique mapping: $\mathcal{C}^{'}_{k_1} = \text{unique}\big(c_i \mid \ell_{i,\cdot} \in \text{top-}k_1(r_{q,\ell})\big)$, where we identify and consolidate the parent chunks of all retrieved slices and obtain a set of selected chunks used for the next steps.

3.2.3 Chunk-level Re-Ranking

For each identified unique relevant chunk from the mapping as $c_i \in \mathcal{C}^{'}_{k_1}$ to the query, we compute a chunk-level ranking score for the query: $r_{q,c_i} = \text{ReRank}\big(q,\, c_i\big)$, utilizing a cross-encoder (*e.g.*, `marco-miniLM`) or advanced ranking model that evaluates the complete chunk content, mitigating potential "information fragmentation" inherent in slice-level retrieval. The chunks in $\mathcal{C}^{'}_{k_1}$ are then reordered according to their ranking scores to the query q.

3.2.4 Scaling-Up & Document-level Ranking

This step aims to select optimal content segments for constructing k_2 precise, bounded long contexts for the final LLM generation, ensuring both relevance and sufficient coverage. It incorporates top-ranked chunks and their context neighbors in the same document to mitigate information fragmentation and support multi-hop reasoning.

Scaled Top Chunk Selection. Recognizing that initial re-ranking scores r_{q,c_i} may not be perfect, we consider a broader set of borderline candidates rather than strictly picking the top-k_2 chunks. MacRAG employs a scaling factor $\alpha \geq 1$ (where $(k_2 \times \alpha) \leq |\mathcal{C}^{'}_{k_1}|$) to select the top-$(k_2 \times \alpha)$ chunks from $\mathcal{C}^{'}_{k_1}$: $\mathcal{C}^{'}_{q,k_2} = \text{TopK}_{(k_2 \times \alpha)}\big(r_{q,c_i}\big)$ for $c_i \in \mathcal{C}^{'}_{q,k_1}$. $\alpha = 1$ does not scale-up the number of borderline candidates, but moderate higher α (*e.g.*, 3 or 4) can provide higher probabilities to include borderline but crucial chunks to improve coverage while maintaining a certain level of quality for candidates. To balance among precision, recall, and context lengths, $\alpha \in \{2, 3, 4\}$ would be promising where $(k_2 \times \alpha) \leq |\mathcal{C}^{'}_{k_1}|$, and we will also demonstrate that selection α is not sensitive, but robust with handling trade-off problems well in Section 4. This approach can also potentially facilitate bridging multi-hop queries.

Document Selection. From the scaled set of $(k_2 \times \alpha)$ chunks in $\mathcal{C}^{'}_{q,k_2}$, we identify their unique source documents. To select the top-k_2 documents, we compute a document-level rank score $r_{q,d}$ for each candidate document d. This score $r_{q,d}$ leverages the chunk-level scores r_{q,c_i} of document's all constituent chunks c_i that are also present in $\mathcal{C}^{'}_{q,k_2}$: $r_{q,d} = \text{ReRank}_{\text{Doc}}\Big(\{c_i \in d \text{ and } c_i \in \mathcal{C}^{'}_{q,k_2}\}\Big)$. Document ranking functions can include `mean`, `max`, `sum` of chunk scores, or an LLM-based ranker (with $\max(r_{q,c_i})$ being a suitable base if using cross-encoder scores, which can be either positive or negative). Based on $r_{q,d}$, we form $\mathcal{D}_{q,k_2}$, a set of top-k_2 *distinct* documents. Ensuring document diversity, especially with appropriate α values (*e.g.*, 3 or 4 depending on the overlapped ratio of original chunks), mitigates the risk of concentrating on too few sources and is critical when evidence is scattered across multiple documents.

3.2.5 Neighbor Chunk Propagation & Merge for Input Context Construction

Within each of the top-k_2 *distinct* documents, we focus on chunks c_i both in $\mathcal{D}_{q,k_2}$ and $\mathcal{C}^{'}_{q,k_2}$, as these represent the most relevant content segments. For such a chunk c_i, we define $\mathcal{N}_h(c_i)$ which consists of c_i's neighbors within h-hops and c_i itself, and process extension of c_i to get $\mathcal{N}(c_i)$ by using its associated indices, $\mathcal{N}_h(c_i) = \big\{c_{i-h}, \ldots, c_i, \ldots, c_{i+h}\big\}$. We then merge $\mathcal{N}_h(c_i)$ within the same ranked document to form the final merged chunk for adaptive long context to the query q: $c^{merge}_{q,d} = \text{Merge}\big(\mathcal{N}_h(c_i)\big)$, where $d \in \mathcal{D}_{q,k_2}$, $c_i \in d$, and $c_i \in \mathcal{C}^{'}_{q,k_2}$. From the top-$k_2$ documents in $\mathcal{D}_{q,k_2}$, we can select top-k_2 merged chunks $c^{merge}_{q,d}$ by considering their order along with the top document order with $r_{q,d}$. Then, finally, we have top-k_2 merged chunks as $\mathcal{C}_{\text{final}} = \text{TopK}_{k_2}\Big(c^{merge}_{q,d}\Big)$. The final merged chunks $\mathcal{C}_{\text{final}}$ provide diverse views across the top-k_2 documents while optimizing context length (Zhao et al., 2024), combining the most relevant content with minimal redundancy.

3.3 Single/Multi-step Post Retrieval Generation

Our retrieval framework is modular, enabling integration with various post-retrieval generation approaches. These include both vanilla single-step generation directly using the retrieved top-k_2 chunks in a single pass and iterative multi-step generation as in LongRAG (Zhao et al., 2024), which introduces a CoT-guided filter (which checks chunk-level relevance) and an LLM-augmented ex-

tractor (which locates global context in original passages). Following the comprehensive exploration of single and multi-step generation variants in LongRAG, we evaluate MacRAG's performance when combined with different post-retrieval generation methods. Specifically, we investigate seven generation methods, as detailed in Table 4. Five of these methods (`R&B`, `R&L`, `Full_Ext`, `Fil`, and `Full_E&F`) are adopted directly from LongRAG. Among these, `R&L`, `Full_Ext`, and `Full_E&F` process the entire set of top-k_2 documents, which can introduce computational overhead when handling very long texts. To address this limitation while maintaining an optimal balance between precision and recall, we introduce two new variants: `R&B_Ext` and `R&B_E&F`, to reduce this overhead while maintaining precision-recall balance.

3.4 Implementation Complexity and System Efficiency

MacRAG enhances simple overlapping-chunk indexing by first chunking each document with partial overlaps, then compressing each chunk through summarization, and finally slicing it. These processes are performed once during offline preparation, eliminating query-time overhead. Moreover, this document-level independence simplifies updates and streamlines database operations, avoiding the complexity associated with hierarchical multi-clustering (Sarthi et al., 2024; Zhang et al., 2025).

At inference time, MacRAG constructs a real-time *effective long context* by mapping slices back to their parent chunks and documents using lightweight index-based lookups rather than costly tree traversals. This design leverages metadata dictionaries (document ID, chunk ID, token offsets) to enable modular neighbor merging with minimal overhead. A bounded context size is enforced through parameters $(k_2 \times \alpha)$ and chunk dimensions, ensuring scalability to enterprise-level corpora without degrading performance. This controlled scalability and real-time context assembly underscore MacRAG's value for massive long-document collections, "almost infinite context" scenarios in advanced personalization, and as a robust foundation for iterative RAG or agentic systems.

4 Experiments

4.1 Experimental Setup

We evaluate MacRAG on three challenging multi-hop question answering benchmarks from LongBench (Bai et al., 2023): HotpotQA, 2WikiMultihopQA, and Musique. These datasets, with their tangential passages and obscured connecting information, specifically test a system's ability to handle the "Lost in the Middle" phenomenon and perform robust multi-hop reasoning—precisely the challenges MacRAG addresses. Section A.2 in the Appendix provides a detailed explanation about these datasets.

Models and Evaluations. Our evaluations employ three prominent LC LLMs: GPT-4o (OpenAI, 2024), Gemini-1.5-pro (Team et al., 2024), and the open-source Llama-3.1-8B-instruct (Dubey et al., 2024a). Performance is measured using Exact Match (EM), F1-score, Precision, and Recall. We primarily compare MacRAG against strong baselines RAPTOR (Sarthi et al., 2024) and LongRAG (Zhao et al., 2024), selected for their representative state-of-the-art performance and efficiency balance in hierarchical RAG (as discussed in Section 2). To ensure rigorous comparative analysis across all RAG methods (MacRAG, LongRAG, and RAPTOR) use the same reranker (`marco-miniLM`), and LongRAG's reported optimal hyperparameters ($k_1 = 100$ initial chunks, $k_2 = 7$ final chunks, unless in ablation). We also applied seven different generation methods described in Appendix A.1, and Table 4 for detailed comparisons.

MacRAG-specific parameters are scale-up factor $\alpha \in \{1, 4\}$ and the neighbor hop count $h \in \{0, 1\}$, except during ablation studies. The ablation study and performance analysis in Section 4.2 demonstrate both robustness and the soundness of the architecture design of MacRAG, enhancing multi-hop reasoning capability.

This controlled setup ensures that performance differences result from MacRAG's architectural innovations, particularly its hierarchical representation and adaptive multi-scale retrieval, rather than from variations in hyperparameters or retrieval components. By isolating architectural factors, we can clearly assess the impact of multi-scale retrieval on multi-hop reasoning.

4.2 Main Results

Comparison with RAPTOR. While RAPTOR (Sarthi et al., 2024), a summarization and hierarchical clustering-based approach, shows gains on standard datasets, it struggles with retrieval quality on LongBench versions (Bai et al., 2023) as shown in

Model	HotpotQA	2WikimultihopQA	Musique	Average
Long Context (LC) LLM without RAG				
Gemini-1.5-pro (Team et al., 2024)	36.79	38.31	20.09	31.73
GPT-4o (OpenAI, 2024)	46.76	40.62	30.76	39.38
Agentic RAG Method				
CRAG (*GPT-3.5-Turbo*) (Yan et al., 2024)	52.04	41.13	25.34	39.50
Self-RAG (*GPT-3.5-Turbo*) (Asai et al., 2024)	50.51	46.75	24.62	40.63
RAG with Reranking (R&B, Base Version)				
Llama3-8B-8k (Dubey et al., 2024b)	48.25	43.47	19.66	37.13
GPT-3.5-Turbo	52.31	43.44	25.22	40.32
Llama-3.1-8B-instruct (abbr. Llama-3.1-8B)	52.50	46.33	26.70	41.84
RAPTOR (Sarthi et al., 2024) (Llama-3.1-8B)	52.30 (-0.20↓)	43.61 (-2.72↓)	23.79 (-2.91↓)	39.90 (-1.94↓, -4.64%↓)
MacRAG (Ours) (Llama-3.1-8B)	**57.39** (**+4.89**↑)	**44.87** (**-1.46**↓)	**30.38** (**+3.68**↑)	**44.21** (**+2.37**↑, **+5.66%**↑)
LongRAG (Zhao et al., 2024) (Multi-step Generation, `Full_E&F` in Table 4)				
LongRAG (GPT-3.5-Turbo)	56.17	51.37	32.83	46.79
LongRAG (Llama-3.1-8B)	58.49	55.90	32.73	49.04
MacRAG (Ours) (Llama-3.1-8B)	**61.10** (**+2.61**↑)	**57.74** (**+1.84**↑)	**34.43** (**+1.70**↑)	**51.09** (**+2.05**↑, **+4.18%**↑)
LongRAG (Gemini-1.5-pro)	61.95	58.17	34.04	51.39
MacRAG (Ours) (Gemini-1.5-pro)	**64.39** (**+2.44**↑)	**64.75** (**+6.58**↑)	**43.34** (**+9.30**↑)	**57.49** (**+6.10**↑, **+11.87%**↑)
LongRAG (GPT-4o)	66.20	65.89	43.83	58.64
MacRAG (Ours) (GPT-4o)	**68.52** (**+2.32**↑)	**73.19** (**+7.30**↑)	**50.09** (**+6.26**↑)	**63.93** (**+5.29**↑, **+9.02%**↑)

Table 1: Experimental results (F1-score) comparing **RAPTOR**, **LongRAG** and **MacRAG** on HotpotQA, 2WikimultihopQA, and MuSiQue datasets from LongBench (Bai et al., 2023). Gains are displayed in parentheses with absolute number and its relative percentage.

Dataset	Avg. Performance on Seven Gen. Settings	LongRAG (Gemini-1.5-pro)	MacRAG (Gemini-1.5-pro)	LongRAG (GPT-4o)	MacRAG (GPT-4o)
HotpotQA	**Exact Match**	46.48	**48.36** (**+1.88**↑, **+4.04%**↑)	51.85	**53.60** (**+1.75**↑, **+3.37%**↑)
	F1-score	61.45	**63.97** (**+2.52**↑, **+4.10%**↑)	66.17	**67.51** (**+1.34**↑, **+2.03%**↑)
	Precision	65.93	**67.98** (**+2.05**↑, **+3.11%**↑)	68.66	**70.10** (**+1.44**↑, **+2.10%**↑)
	Recall	61.59	**64.47** (**+2.88**↑, **+4.68%**↑)	68.10	**69.36** (**+1.26**↑, **+1.85%**↑)
2WikimultihopQA	**Exact Match**	49.93	**53.69** (**+3.76**↑, **+7.53%**↑)	52.90	**57.60** (**+4.69**↑, **+8.87%**↑)
	F1-score	57.73	**61.97** (**+4.24**↑, **+7.34%**↑)	62.69	**67.45** (**+4.76**↑, **+7.59%**↑)
	Precision	57.97	**61.69** (**+3.72**↑, **+6.41%**↑)	62.22	**66.48** (**+4.26**↑, **+6.85%**↑)
	Recall	60.17	**65.10** (**+4.93**↑, **+8.20%**↑)	66.47	**72.00** (**+5.53**↑, **+8.32%**↑)
Musique	**Exact Match**	26.05	**33.75** (**+7.70**↑, **+29.57%**↑)	31.30	**36.48** (**+5.18**↑, **+16.55%**↑)
	F1-score	33.92	**42.84** (**+8.93**↑, **+26.34%**↑)	41.43	**47.91** (**+6.48**↑, **+15.64%**↑)
	Precision	34.69	**43.29** (**+8.60**↑, **+24.78%**↑)	40.80	**47.07** (**+6.27**↑, **+15.37%**↑)
	Recall	35.69	**45.16** (**+9.46**↑, **+26.50%**↑)	45.06	**51.67** (**+6.61**↑, **+14.67%**↑)

Table 2: Consolidated results for four metrics across three datasets, comparing LongRAG and MacRAG using the Gemini-1.5-pro and GPT-4o models. The results are averaged over seven generation settings in Table 4.

Table 1. Specifically, RAPTOR with Llama-3.1-8B shows performance decreases in 2WikimultihopQA and Musique, potentially associated with retrieval difficulty on Musique-Normal in (Yue et al., 2024). These results highlight the importance of both precise retrieval and effective long-context construction for achieving consistent gains across various multi-hop reasoning and long-context datasets.

Integration with Multi-step Generation Method. Integrating MacRAG with the LongRAG (`E&F`) generation method across multiple LLMs (Llama-3.1-8B, Gemini-1.5-pro, and GPT-4o) yields substantial performance improvements, as shown in Table 1. This underscores MacRAG's versatility and robustness as a retrieval approach when paired with the LongRAG generation method. By constructing an effective long context, MacRAG demonstrates advantages in multi-step generation with LongRAG (`E&F`), producing consistent and meaningful gains across three datasets and three LLMs. In contrast, it is not straightforward to extend RAPTOR in combination with LongRAG (`E&F`). As further illustrated in Table 1, the performance gains for Gemini-1.5-pro and GPT-4o indicate that pairing MacRAG with more powerful LLMs can achieve even greater improvements, despite already high F1-scores obtained by other frameworks under these settings. We further observe that summarization during indexing improves retrieval quality. Our preliminary experiments show a 7.2% gain in precision when using summaries instead of raw slices.

Robustness across Generation Methods. In addition to MacRAG's robust gains in both vanilla single-step (`R&B`) and multi-step generation (`E&F`), MacRAG consistently shows strong advantages across the four metrics listed in Table 2, covering all seven generation modes in Table 4. These substantial improvements across various datasets,

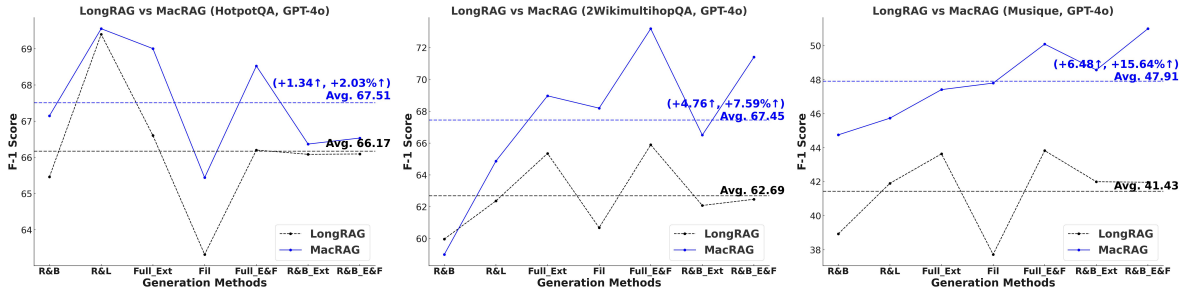

Figure 2: Performance comparison of GPT-4o using LongRAG and MacRAG across the seven settings in Table 4, showing F1-scores for three datasets (HotpotQA, 2WikiMultihopQA, and MuSiQue). Complete results for all metrics and LLMs are provided in the Appendix.

(k_2=7, *marco-miniLM*)	R&B	R&L	Full_Ext	Fil	Full_E&F	R&B_Ext	R&B_E&F	Average
HotpotQA								
LongRAG	65.46	69.40	66.60	63.32	66.20	66.08	66.09	66.16
MacRAG	67.15 (**+1.69**↑)	69.55 (**+0.15**↑)	69.00 (**+2.40**↑)	65.44 (**+2.12**↑)	68.52 (**+2.32**↑)	66.36 (**+0.28**↑)	66.53 (**+0.44**↑)	**67.51** (**+1.35**↑, **+2.04%**↑)
- Propagation&Merging	64.17 (**-1.29**↓)	68.91 (**-0.49**↓)	67.82 (**+1.22**↑)	68.06 (**+4.74**↑)	68.44 (**+2.24**↑)	63.77 (**-2.31**↓)	65.57 (**-0.52**↓)	**66.68** (**+0.52**↑, **+0.79%**↑)
- Scaling up	64.12 (**-1.34**↓)	67.57 (**-1.83**↓)	67.33 (**+0.73**↑)	64.41 (**+1.09**↑)	67.26 (**+1.06**↑)	64.22 (**-1.86**↓)	63.84 (**-2.44**↓)	**65.54** (**-0.62**↓, **-0.94%**↓)
2WikimultihopQA								
LongRAG	59.97	62.37	65.35	60.68	65.89	62.08	62.47	62.69
MacRAG	59.00 (**-0.97**↓)	64.87 (**+2.50**↑)	68.97 (**+3.62**↑)	68.20 (**+7.52**↑)	73.19 (**+7.30**↑)	66.50 (**+4.42**↑)	71.40 (**+8.93**↑)	**67.45** (**+4.76**↑, **+7.59%**↑)
- Propagation&Merging	59.67 (**-0.30**↓)	65.06 (**+2.69**↑)	68.48 (**+3.13**↑)	67.57 (**+6.89**↑)	72.20 (**+6.31**↑)	65.13 (**+3.05**↑)	72.43 (**+9.96**↑)	**67.22** (**+4.53**↑, **+7.23%**↑)
- Scaling up	59.00 (**-0.97**↓)	61.89 (**-0.48**↓)	67.63 (**+2.28**↑)	60.41 (**-0.27**↓)	67.03 (**+1.14**↑)	66.37 (**+4.29**↑)	65.77 (**+3.30**↑)	**64.01** (**+1.32**↑, **+2.11%**↑)
Musique								
LongRAG	38.98	41.90	43.64	37.72	43.83	42.00	41.97	41.43
MacRAG	44.76 (**+5.78**↑)	45.74 (**+3.84**↑)	47.42 (**+3.78**↑)	47.80 (**+10.08**↑)	50.09 (**+6.26**↑)	48.57 (**+6.57**↑)	51.00 (**+9.03**↑)	**47.91** (**+6.48**↑, **+15.63%**↑)
- Propagation&Merging	41.61 (**+2.63**↑)	45.43 (**+3.53**↑)	48.00 (**+4.36**↑)	47.48 (**+9.76**↑)	52.26 (**+8.43**↑)	40.09 (**-1.91**↓)	45.84 (**+3.87**↑)	**45.82** (**+4.39**↑, **+10.58%**↑)
- Scaling up	41.65 (**+2.67**↑)	45.87 (**+3.97**↑)	46.80 (**+3.16**↑)	41.83 (**+4.11**↑)	46.59 (**+2.76**↑)	39.40 (**-2.60**↓)	39.63 (**-2.34**↓)	**43.11** (**+1.68**↑, **+4.04%**↑)

Table 3: Ablation study of MacRAG with GPT-4o and F1-score, using the same k_1, k_2, and reranker.

metrics, and LLMs demonstrate the effectiveness of MacRAG's long-context retrieval in enhancing both precision and recall for RAG. Notably, MacRAG achieves significant gains on the Musique datasets, which (Yue et al., 2024) identifies as having challenging retrieval conditions. Beyond the average gains in these four metrics, Figure 2 further confirms MacRAG's robust benefits in all seven single-/multi-step generation schemes in Table 4 in Appendix A.1. Finally, regardless of the choice of k_2 or reranker, MacRAG maintains substantial advantages in all test settings in Table 5.

Enhanced Gains with Stronger Models. When applied to the more powerful Gemini-1.5-pro and GPT-4o models, MacRAG showcases its capacity to further improve performance by optimizing the retrieval process and building more relevant, informative contexts. Its use of hierarchical chunking and slicing, coupled with adaptive propagation and merging, maintains high precision while expanding coverage. This leads to more efficient handling of long documents, reduced computational overhead, and higher-quality generated answers overall.

Ablation Study. Table 3 presents an ablation study demonstrating the effectiveness of MacRAG's key components: Propagation and Merging, Scaling-up, and Hierarchical Slicing Retrieval. Each component contributes to cumulative performance gains. The combination of propagation and merging with the scaling-up mechanism increases the coverage of relevant contexts for multi-hop reasoning tasks. Figure 3 further illustrates the benefits of scaling up promising candidates with $\alpha = [2, 3, 4]$, showing how this approach effectively balances context lengths while incorporating additional relevant candidates. We observe that performance gains remain consistent across different k_2 values and rerankers, indicating robustness. Although the parameters (k_1, k_2, α, h) are fixed, MacRAG adaptively merges nearby slices depending on partial relevance signals. Removing this adaptivity leads to a 5% F1 drop, as shown in our ablations.

4.3 Generation Schemes and Input Lengths

While multi-step generation methods (e.g., LongRAG's Full_E&F) improve answer quality, they often increase cumulative input length. Figure 4 compares cumulative input lengths for single and multi-step generation methods in Table 4, given single retrieval with specific settings, highlighting

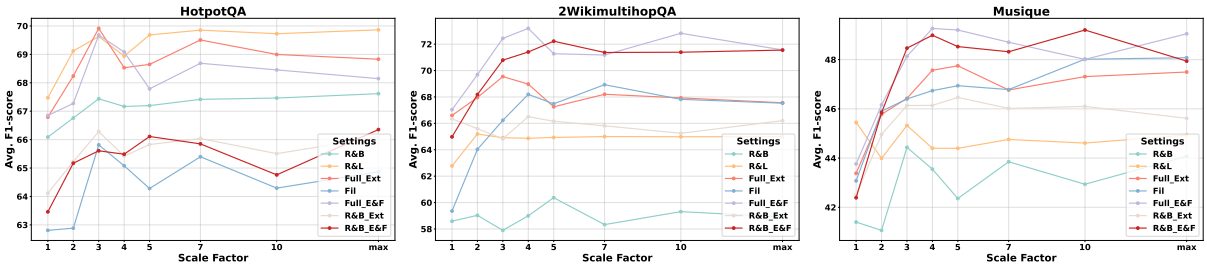

Figure 3: Performance trends across datasets for scale factors (α).

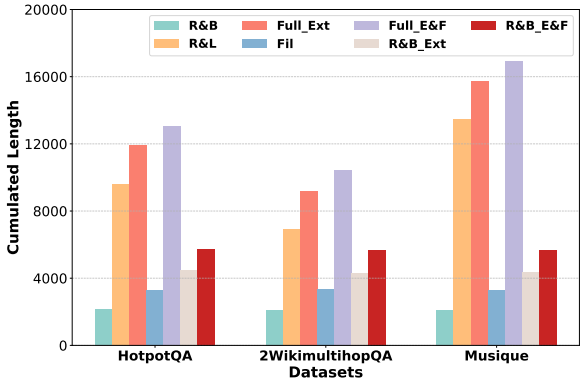

Figure 4: Cumulative LLM's input context lengths for single/multi-step generation methods in Table 4.

that R&L, Full_Ext, and Full_E&F accumulate the most overhead by relying on entire documents. In contrast, MacRAG's final k_2 chunks, though potentially larger than basic R&B chunks, preserve bounded context well below full-document with long lengths.

As Figure 2 and Table 3 illustrate, MacRAG consistently outperforms baselines under both R&B_Ext and R&B_E&F settings. Due to MacRAG's advantage in effective and stable long context construction, even our intermediate R&B_E&F variant demonstrates significant improvements as shown in Table 3. For example, on 2WikiMultihopQA (GPT-4o), MacRAG (R&B_E&F) achieves an F1 of 72.43 versus LongRAG (Full_E&F) at 65.89 (+9.93% relative gain), while using 8.19% less context. On Musique, the intermediate variant of MacRAG (R&B_E&F) reaches 45.84 compared to LongRAG (Full_E&F)'s 43.83 (+4.58% relative) with a 45.87% reduction in context length. This highlights MacRAG's efficiency in evidence gathering and context construction.

4.4 Efficiency and Latency Analysis

MacRAG maintains sub-second efficiency despite constructing richer multi-scale contexts. On average, retrieval and reranking take 0.23 seconds per query, which is 38% faster than RAPTOR's 0.37 seconds. With Llama 3.1 8B, total inference latency is 0.99 seconds (0.23s retrieval and 0.76s generation), compared to 0.80 seconds for RAPTOR (0.37s and 0.43s, respectively). This slight increase in generation time is offset by improved retrieval precision and higher answer quality on R&B queries.

This efficiency is enabled by MacRAG's lightweight, index-based context construction and shared reranking modules, which preserve the same level of computational complexity as baseline retrieval pipelines. MacRAG introduces a one-time offline summarization step during indexing, similar to RAPTOR and SIRERAG. This step runs in parallel with standard RAG indexing, adding no runtime latency and supporting efficient index maintenance. As shown in Table 3, Figure 4, and Subsection 4.3, MacRAG achieves a better trade-off than LongRAG between input context length, answer quality, and overall computational cost.

5 Conclusion

In this work, we introduced Multi-scale Adaptive Context RAG (MacRAG), a novel framework designed to address key trade-offs in RAG involving precision, coverage, and computational efficiency for long-context question answering and multi-hop reasoning. By dynamically constructing query-adaptive long contexts through real-time, multi-scale retrieval, MacRAG demonstrably improves both precision and recall while maintaining operational efficiency. Our extensive experiments validated MacRAG's advantages across diverse datasets and both single- and multi-step generation paradigms, showcasing consistent and notable performance gains. These findings establish MacRAG as a potent and efficient core module for advancing next-generation iterative and agentic RAG systems, with significant potential for robust enterprise-level applications.

Limitations

While MacRAG achieves strong performance with a modular and efficient design, there remain opportunities for further extension. Its current offline summarization and indexing strategy, though effective and amortized, may benefit from adaptive or online variants in evolving corpora. The use of fixed hyperparameters like chunk size and expansion scale has shown robustness across settings, but dynamic adjustment based on query complexity is a promising direction. In addition, integrating MacRAG more tightly with agentic or multi-round retrieval workflows could enhance its utility in interactive and real-time RAG applications.

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*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. 2025. Improving retrieval-augmented generation through multi-agent reinforcement learning. *Preprint*, arXiv:2501.15228.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024a. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024b. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

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 (Volume 1: Long Papers)*, pages 7029–7043.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Quinn Leng, Jacob Portes, Sam Havens, Matei Zaharia, and Michael Carbin. 2024. Long context rag performance of large language models. *arXiv preprint arXiv:2411.03538*.

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

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

OpenAI. 2024. Hello gpt-4o. *OpenAI Blogs*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025. Chain-of-retrieval augmented generation. *Preprint*, arXiv:2501.14342.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2024b. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Tan Yu, Anbang Xu, and Rama Akkiraju. 2024. In defense of rag in the era of long-context language models. *arXiv preprint arXiv:2409.01666*.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference scaling for long-context retrieval augmented generation. *arXiv preprint arXiv:2410.04343*.

Nan Zhang, Prafulla Kumar Choubey, Alexander Fabbri, Gabriel Bernadett-Shapiro, Rui Zhang, Prasenjit Mitra, Caiming Xiong, and Chien-Sheng Wu. 2025. SireRAG: Indexing similar and related information for multihop reasoning. In *The Thirteenth International Conference on Learning Representations*.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. 2024. Longrag: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22600–22632.

A Appendix

A.1 Generation Modes

Table 4 summarizes different generation modes. In addition to MacRAG's robust gains in both vanilla single-step (R&B) and multi-step generation (E&F), MacRAG consistently shows strong advantages across the four metrics listed in Table 2, covering all seven generation modes in Table 4. These substantial improvements across various datasets, metrics, and LLMs demonstrate the effectiveness of MacRAG's long-context retrieval in enhancing both precision and recall for RAG. Notably, MacRAG achieves significant gains on the Musique datasets, which (Yue et al., 2024) identifies as having challenging retrieval conditions. Beyond the average gains in these four metrics, Figure 2 further confirms MacRAG's robust benefits in all seven single-/multi-step generation schemes in Table 4 in Appendix A.1. Finally, regardless of the choice of k_2 or reranker, MacRAG maintains substantial advantages in all test settings in Table 5.

A.2 Benchmark Datasets and Challenges

The LongBench variants of HotpotQA, 2WikimultihopQA, and Musique datasets have become the standard evaluation suite for hierarchical retrieval research, including recent systems such as SIRERAG (Zhang et al., 2025), RAPTOR (Sarthi et al., 2024), and LongRAG (Zhao et al., 2024). They specifically intensify the "Lost in the Middle" phenomenon by introducing near-duplicate and tangential passages, creating challenging retrieval environments where critical connecting information becomes obscured by surrounding context. Each dataset introduces distinct multi-hop reasoning challenges that systematically test different aspects of retrieval capability: HotpotQA features bridging and comparison queries requiring evidence synthesis across multiple paragraphs. 2WikiMultihopQA expands partial excerpts to complete articles, significantly increasing the risk that essential connecting information becomes buried within expansive context. Musique embeds nested sub-questions within tangential sections, challenging systems to identify and connect scattered bridging facts across document boundaries. These characteristics directly test a retrieval system's ability to handle fragmented evidence, reconcile overlapping information, and identify critical bridging facts despite contextual noise (Yue et al., 2024; Leng et al., 2024), precisely the challenges that MacRAG's multi-scale architecture addresses through hierarchical indexing and adaptive retrieval.

A.3 Target Baseline Methods

Based on SIRERAG's comparative analysis (Table 4 in Zhang et al. (2025)), we focus our evaluation on RAPTOR (Sarthi et al., 2024) and LongRAG (Zhao et al., 2024) as representative state-of-the-art hierarchical RAG systems. SIRERAG's analysis establishes that GraphRAG underperforms on these specific multi-hop tasks, while RAPTOR, HippoRAG, and SIRERAG achieve comparable performance in terms of F1-score on multiple multi-hop question answering datasets (Zhang et al., 2025). Among these top-performing systems, we selected RAPTOR and LongRAG because they balance performance and computational efficiency better, offering lower computational costs and easier maintenance than SIRERAG's higher overhead.

A.4 Adaptive Retrieval Mechanism Analysis

A key strength of MacRAG lies in its adaptive retrieval mechanism, which dynamically constructs query-specific contexts despite using fixed common hyperparameters (k_1, k_2) and MacRAG's α, and h. This adaptivity operates through several complementary mechanisms:

Query-dependent Adaptive Context Expansion: For complex multi-hop queries, MacRAG naturally extends its context boundaries to encompass connecting information. For simpler queries, expansion remains more focused, selectively around the most promising content regions, but avoiding unnecessary content. Our ablation studies quantify the impact of this adaptivity. When the adaptivity of query-adaptive expansion of neighborhoods is removed by eliminating the propagation and merging steps (Table 3), performance drops by up to around 5% F1-score on datasets. This degradation is most pronounced on Musique, where connecting multiple scattered pieces of evidence is crucial. Table 3 additionally reveals another important aspect of query-adaptive expansion of borderline candidates "Scaling-up", and if we eliminate its corresponding "Scaling-up" component, then performance drops up to around 6% F1-score on datasets. This aligns with our expectation that adaptive expansion becomes increasingly valuable as query complexity grows.

Mode	Description
`R&B`	Single-step generation using retrieved top-k_2 chunks, which is the vanilla basic version.
`R&L`	Single-step generation using complete documents from top-k_2 chunks. Prone to the "Lost in the Middle" effect and inefficient for long documents.
`Full_Ext`	Multi-step generation: (1) LLM extracts query-relevant content from full documents of top-k_2 chunks, (2) generates answer using both chunks and extracted content.
`Fil`	Filtering re-ranked top-k_2 chunks, *e.g.*, via LLM. Improves precision but may lose borderline info.
`Full_E&F`	Multi-step generation combining `Full_Ext`'s extracted content and `Fil`'s filtered chunks. Balances coverage and precision, but is expensive for long documents.
`R&B_Ext`	Multi-step generation with content extraction limited to top-k_2 chunks, reducing computational overhead.
`R&B_E&F`	Multi-step generation combining `R&B_Ext`'s extracted content and `Fil`'s filtered chunks. Optimizes precision-recall trade-off w/o processing the full document.

Table 4: Seven different generation methods with single-step and multi-step generation schemes using retrieved top-k_2 chunks.

A.5 Preprocessing Efficiency of MacRAG

Chunk Size and Token Budget: MacRAG uses original chunk sizes of approximately 400 tokens (around 1500 characters), which fall within the 200–500 token range used in prior work such as LongRAG even for re-ranking on retrieved chunks. The average sizes of summarized chunks are around 800-1000 characters. We applied slices into summarized chunks for initial retrieval to improve precision further with two slice sizes around 450 characters with 300 character overlaps or 600 characters with 450 character overlaps. These sizes allow MacRAG to maintain sub-second retrieval and reranking times while keeping the context structure manageable for downstream language models.

Offline Summarization Cost: MacRAG includes a one-time summarization step during indexing, which does not affect query-time latency. This step is comparable in cost to offline embedding computations in standard RAG pipelines. All runtime efficiency measurements reported here exclude this indexing cost, as it is amortized over the use of the document collection.

A.6 Latency Comparison between MacRAG and RAPTOR

Retrieval and Reranking Latency: MacRAG maintains sub-second latency for retrieval and reranking across all evaluated datasets. Using dense retrieval with $k_1 = 100$ followed by cross-encoder reranking with $k_2 = 7$, the measured latency is 0.23 seconds on HotpotQA, 0.22 seconds on 2WikimultihopQA, and 0.24 seconds on Musique. In contrast, RAPTOR reports 0.43s, 0.24s, and 0.44s on the same datasets, averaging 0.37s. This represents a 38% speedup on average, even though both methods operate on the same top-100 retrieved chunks and use the same reranker (marco-miniLM). MacRAG achieves this by leveraging index-based merging after reranking, which reuses precomputed relevance scores and adds only a few milliseconds per query.

Generation Time and Trade-offs: While retrieval and reranking are efficient, generation time varies depending on the context length produced by each method. MacRAG's adaptive multi-scale expansion typically results in longer but more relevant contexts. With Llama 3.1 8B, the average generation time with `R&B` is 0.76 seconds, compared to 0.43 seconds for RAPTOR with `R&B`. This increase reflects MacRAG's strategy of incorporating broader evidence to improve answer accuracy. Users seeking faster inference can reduce the expansion parameter α to trade off coverage for speed, though at the cost of some performance.

Total Latency Comparison: Combining retrieval, reranking, and generation with `R&B`, **the total end-to-end latency of MacRAG is 0.99 seconds (0.23s + 0.76s), compared to RAPTOR's 0.80 seconds (0.37s + 0.43s)**. While MacRAG incurs a small additional cost, it consistently yields higher accuracy across multiple datasets and models, as shown in Table 1.

A.7 Efficiency Comparison between MacRAG and LongRAG

MacRAG and LongRAG both rely on index-based hierarchical retrieval and reranking over the same number of k_1 chunks. As a result, their retrieval and reranking steps show similar efficiency and latency. However, for the generation step, according to Table 3, Figure 4, and Subsection 4.3, MacRAG achieves improved efficiency compared to LongRAG in terms of the trade-off between final an-

swer quality, input context length, and overall computational cost.

A.8 Discussion on Effectiveness of MacRAG's Retrieval and Constructed Long Context

Enhanced Precision and Coverage: MacRAG employs a hierarchical multi-scale strategy that begins with fine-grained slice retrieval and chunk-level re-ranking for precise identification of relevant content, then systematically incorporates broader document-level context through h-hop neighbor expansions. To preserve coherence without overwhelming the LLM, it strategically upscales via the $(\alpha \times k_2)$ factor, capturing borderline yet crucial segments. Rather than relying on compressed text or entire documents, MacRAG focuses on real-time construction of effective long contexts from promising original chunks while enforcing an upper bound on context length. By removing irrelevant portions from documents and maintaining a continuous, coherent subset of text, MacRAG avoids excessive token consumption, mitigates the "lost in the middle" phenomenon, and minimizes hallucinations, ultimately ensuring high recall for complex multi-hop queries.

A.9 Discussion on Single, Multi-Step, Iterative, and Agentic Generation with MacRAG

MacRAG's modular design seamlessly supports single and multi-step generation, iterative retrieval, and agentic pipelines. By combining its core multi-scale retrieval with standard generation methods, MacRAG dynamically adapts context formation and refinement across multiple rounds without overwhelming the LLM. In single and multi-step modes (*e.g.*, LongRAG (Zhao et al., 2024) and standard multi-step QA in Section 3.3), MacRAG selects and assembles relevant text from large corpora while minimizing noise and retaining crucial connections, thereby improving both recall and precision for multi-hop reasoning. For iterative scenarios, such as IterDRAG (Yue et al., 2024) or chain-of-RAG (Wang et al., 2025), MacRAG can support efficient updates on the evidence set via $\mathcal{C}_t \leftarrow \text{Merge}\Big(\mathcal{C}_{t-1}, \text{Retrieval}(q_t)\Big)$ to unify old and newly retrieved content, promoting consistent coverage of bridging facts while discarding irrelevant material. In agentic RAG settings (Asai et al., 2024; Jeong et al., 2024; Chen et al., 2025), MacRAG likewise prevents context explosion by focusing specifically on segments required for each action, thereby improving precision and recall over multiple steps. Our experiments (Section 4) confirm that integrating MacRAG with single and multi-step generation methods, potential core modules for multi-round advanced systems, consistently enhances complex multi-hop reasoning tasks and reduces error rates in long-context RAG.

A.10 Discussion on Graph-Enhanced MacRAG

A promising direction is extending MacRAG with a two-stage reranking strategy incorporating graph-based knowledge structures. After initial chunk-level reranking, MacRAG could perform a second reranking phase by expanding top candidates through both local index-based extensions and pre-constructed graph neighbors, enabling efficient coverage of both local and global relationships. This approach would leverage MacRAG's bounded context guarantees at each phase while allowing controlled exploration of knowledge-guided connections, managing computational efficiency through bounded candidate sets during reranking. By applying MacRAG's document-oriented indexing or relationship-based thresholding to merge expanded candidates, this strategy would enhance retrieval quality for complex queries requiring both precise local context and broad knowledge integration.

Model	k_2	Reranker	R&B	R&L	Full_Ext	Fil	Full_E&F	R&B_Ext	R&B_E&F	Average
HotpotQA										
LongRAG	7	bge-m3	67.67	67.99	68.96	64.30	68.49	66.73	66.10	67.18
MacRAG	7	bge-m3	67.59 (**-0.08**↓)	68.63 (**+0.64**↑)	70.53 (**+1.57**↑)	65.56 (**+1.26**↑)	70.29 (**+1.80**↑)	66.59 (**-0.14**↓)	67.32 (**+1.22**↑)	**68.07** (**+0.89**↑, **+1.32%**↑)
LongRAG	12	bge-m3	68.57	67.65	70.31	64.74	70.14	67.63	67.49	68.08
MacRAG	12	bge-m3	67.88 (**-0.69**↓)	69.87 (**+2.22**↑)	70.72 (**+0.41**↑)	65.98 (**+1.24**↑)	70.66 (**+0.52**↑)	69.05 (**+1.42**↑)	67.84 (**+0.35**↑)	**68.86** (**+0.78**↑, **+1.15%**↑)
2WikimultihopQA										
LongRAG	7	bge-m3	59.36	65.56	68.27	55.31	67.36	64.42	63.88	64.45
MacRAG	7	bge-m3	62.32 (**+2.96**↑)	66.34 (**+0.78**↑)	71.63 (**+3.36**↑)	69.72 (**+14.41**↑)	73.98 (**+6.62**↑)	66.61 (**+2.19**↑)	72.63 (**+8.75**↑)	**69.03** (**+4.58**↑, **+7.11%**↑)
LongRAG	12	bge-m3	60.08	66.77	69.28	58.50	69.39	64.90	65.28	64.89
MacRAG	12	bge-m3	64.29 (**+4.21**↑)	67.20 (**+0.43**↑)	70.95 (**+1.67**↑)	69.46 (**+10.96**↑)	73.90 (**+4.51**↑)	67.49 (**+2.59**↑)	71.80 (**+6.52**↑)	**69.30** (**+4.41**↑, **+6.80%**↑)
Musique										
LongRAG	7	bge-m3	42.34	48.08	47.88	42.17	47.92	43.70	44.06	45.16
MacRAG	7	bge-m3	45.54 (**+3.20**↑)	46.68 (**-1.40**↓)	49.58 (**+1.70**↑)	46.76 (**+4.59**↑)	49.53 (**+1.61**↑)	47.94 (**+4.24**↑)	49.14 (**+5.08**↑)	**47.88** (**+2.72**↑, **+6.02%**↑)
LongRAG	12	bge-m3	41.53	46.96	49.58	41.60	49.57	46.76	46.33	46.05
MacRAG	12	bge-m3	46.44 (**+4.91**↑)	45.81 (**-1.15**↓)	51.02 (**+1.44**↑)	46.70 (**+5.10**↑)	51.54 (**+1.97**↑)	50.25 (**+3.49**↑)	49.50 (**+3.17**↑)	**48.75** (**+2.70**↑, **+5.86%**↑)

Table 5: Robust test of MacRAG with various k_2 an alternative re-ranker on three multi-hop QA datasets, using GPT-4o and F1-score. The experiments were conducted with the same hyperparameters as reported in LongRAG (Zhao et al., 2024).

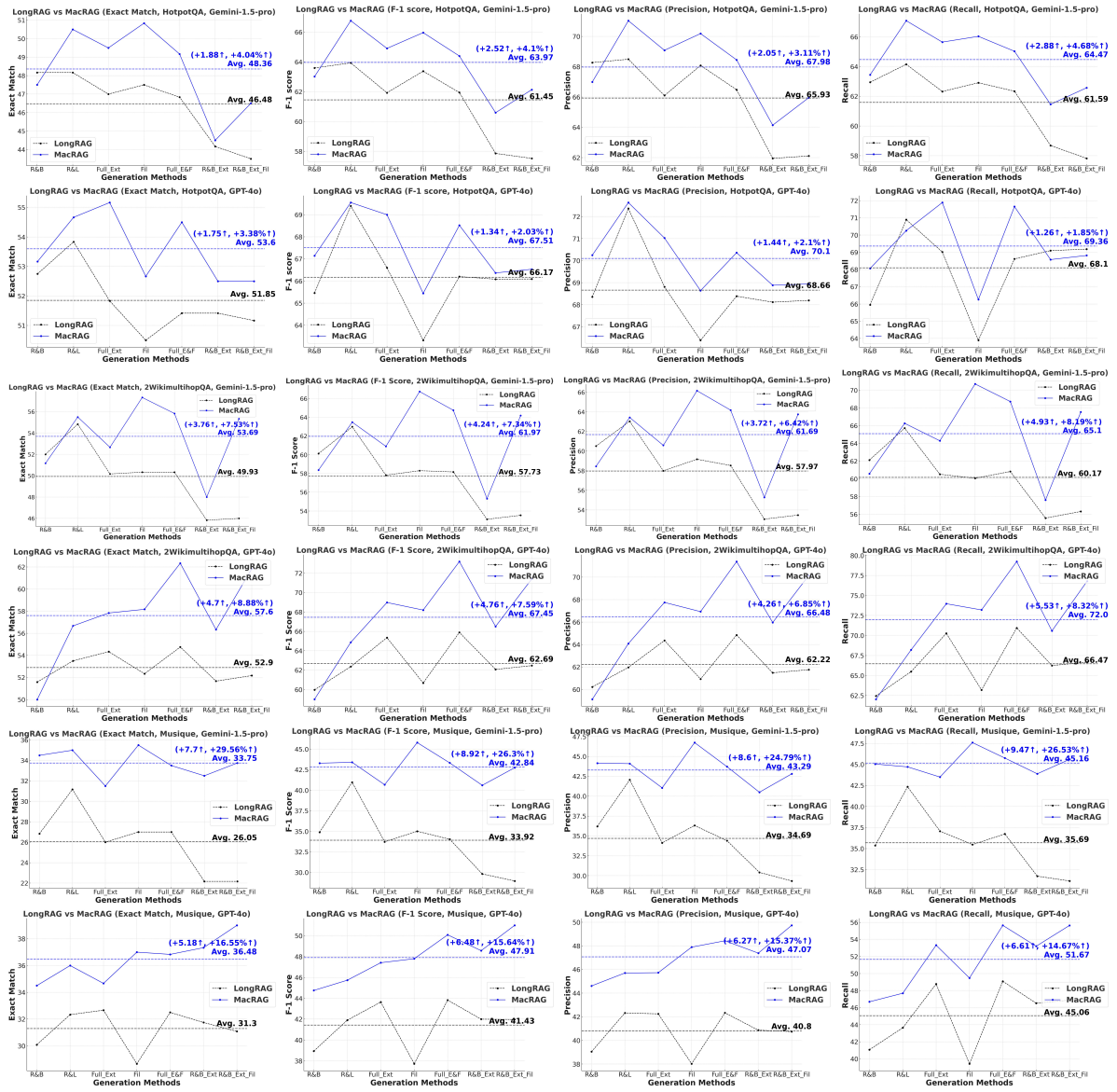

Figure 5: Performances of LongRAG and MacRAG regarding the fours metrics (Exact Match, F1-score, Precision, Recall) for three datasets (HotpotQA, 2WikimultihopQA, and Musique) and two LLMs (Gemini-1.5-pro and GPT-4o). Each row corresponds to a combination of dataset and LLM, and each column represents one of the metrics.

HotpotQA								
Method ($k_2 = 7$)	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	63.59	63.93	61.93	63.38	61.95	57.86	57.53	61.45
MacRAG	63.02 (**-0.57**↓)	66.76 (**+2.83**↑)	64.90 (**+2.97**↑)	65.96 (**+2.58**↑)	64.40 (**+2.45**↑)	60.59 (**+2.73**↑)	62.14 (**+4.61**↑)	**63.97** (**+2.52**↑, **+4.10%**↑)
2WikimultihopQA								
Method ($k_2 = 7$)	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	60.13	62.99	57.82	58.31	58.17	53.12	53.54	57.73
MacRAG	58.38 (**-1.75**↓)	63.48 (**+0.49**↑)	60.90 (**+3.08**↑)	66.74 (**+8.43**↑)	64.75 (**+6.58**↑)	55.32 (**+2.20**↑)	64.19 (**+10.65**↑)	**61.85** (**+4.12**↑, **+7.14%**↑)
Musique								
Method ($k_2 = 7$)	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	34.90	40.97	33.73	35.01	34.04	29.82	28.96	33.92
MacRAG	43.31 (**+8.41**↑)	43.41 (**+2.44**↑)	40.68 (**+6.96**↑)	45.81 (**+10.80**↑)	43.34 (**+9.30**↑)	40.61 (**+10.79**↑)	42.76 (**+13.80**↑)	**42.84** (**+8.92**↑), **+26.30%**↑)

Table 6: Experimental results of Gemini-1.5-pro with F1-score for comparing LongRAG vs MacRAG across HotpotQA, 2WikimultihopQA, and Musique datasets. The experiments conducted with the same hyper-parameter (k1=100 , k2=7) which is the reported best parameter of LongRAG (Zhao et al., 2024). The columns represent various evaluation settings: R&B (Retrieval and Base), R&L (Retrieval and Long), Full_Ext (Extraction from Full Document), Fil (Filtering), Full_E&F (Extraction and Filtering combined), R&B_Ext (Extraction from Top-k_2 Chunks), and R&B_Ext_Fil (Extraction and Filtering from Top-k_2 Chunks). The absolute gains and relative percentage improvements from applying MacRAG to LongRAG are displayed in parentheses.

HotpotQA								
$k_2 = 7$, **marco-miniLM**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	65.46	69.40	66.60	63.32	66.20	66.08	66.09	66.16
MacRAG	67.15 (**+1.69**↑)	69.55 (**+0.15**↑)	69.00 (**+2.40**↑)	65.44 (**+2.12**↑)	68.52 (**+2.32**↑)	66.36 (**+0.28**↑)	66.53 (**+0.44**↑)	**67.22** (**+1.06**↑)
$k_2 = 12$, **marco-miniLM**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	67.97	69.80	67.91	63.56	67.82	66.91	66.44	67.20
MacRAG	68.41 (**+0.44**↑)	69.64 (**-0.16**↓)	69.95 (**+2.04**↑)	65.42 (**+1.86**↑)	69.09 (**+1.27**↑)	67.57 (**+0.66**↑)	66.95 (**+0.51**↑)	**68.00** (**+0.80**↑)
$k_2 = 7$, **bge-m3**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	67.67	67.99	68.96	64.30	68.49	66.73	66.10	67.18
MacRAG	67.59 (**-0.08**↓)	68.63 (**+0.64**↑)	70.53 (**+1.57**↑)	65.56 (**+1.26**↑)	70.29 (**+1.80**↑)	66.59 (**-0.14**↓)	67.32 (**+1.22**↑)	**68.07** (**+0.89**↑)
$k_2 = 12$, **bge-m3**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	68.57	67.65	70.31	64.74	70.14	67.63	67.49	68.08
MacRAG	67.88 (**-0.69**↓)	69.87 (**+2.22**↓)	70.72 (**+0.41**↑)	65.98 (**+1.24**↑)	70.66 (**+0.52**↑)	69.05 (**+1.42**↑)	67.84 (**+0.35**↑)	**68.86** (**+0.78**↑)
2WikimultihopQA								
$k_2 = 7$, **marco-miniLM**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	59.97	62.37	65.35	60.68	65.89	62.08	62.47	62.69
MacRAG	59.00 (**-0.97**↓)	64.87 (**+2.50**↑)	68.97 (**+3.62**↑)	68.20 (**+7.52**↑)	73.19 (**+7.30**↑)	66.50 (**+4.42**↑)	71.40 (**+8.93**↑)	**67.45** (**+4.76**↑)
$k_2 = 12$, **marco-miniLM**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	62.64	65.43	70.67	62.06	70.66	67.56	66.60	66.52
MacRAG	63.47 (**+0.83**↑)	67.06 (**+1.63**↑)	70.61 (**-0.06**↓)	67.84 (**+5.78**↑)	72.24 (**+1.58**↑)	67.83 (**+0.27**↑)	72.23 (**+5.63**↑)	**68.75** (**+2.23**↑)
$k_2 = 7$, **bge-m3**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	59.36	65.56	68.27	55.31	67.36	64.42	63.88	64.45
MacRAG	62.32 (**+2.96**↑)	66.34 (**+0.78**↑)	71.63 (**+3.36**↑)	69.72 (**+14.41**↑)	73.98 (**+6.62**↑)	66.61 (**+2.19**↑)	72.63 (**+8.75**↑)	**69.03** (**+5.58**↑)
$k_2 = 12$, **bge-m3**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	60.08	66.77	69.28	58.50	69.39	64.90	65.28	64.89
MacRAG	64.29 (**+4.21**↑)	67.20 (**+0.43**↑)	70.95 (**+1.67**↑)	69.46 (**+10.96**↑)	73.90 (**+4.51**↑)	67.49 (**+2.59**↑)	71.80 (**+6.52**↑)	**69.30** (**+4.41**↑)
Musique								
$k_2 = 7$, **marco-miniLM**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	38.98	41.90	43.64	37.72	43.83	42.00	41.97	41.43
MacRAG	44.76 (**+5.78**↑)	45.74 (**+3.84**↑)	47.42 (**+3.78**↑)	47.80 (**+10.08**↑)	50.09 (**+6.26**↑)	48.57 (**+6.57**↑)	51.00 (**+9.03**↑)	**47.77** (**+6.34**↑)
$k_2 = 12$, **marco-miniLM**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	41.20	43.85	48.19	40.48	47.22	44.66	44.51	44.30
MacRAG	45.85 (**+4.65**↑)	47.66 (**+3.81**↑)	48.27 (**+0.08**↑)	49.01 (**+8.53**↑)	48.99 (**+1.77**↑)	49.35 (**+4.69**↑)	50.57 (**+6.06**↑)	**48.53** (**+4.23**↑)
$k_2 = 7$, **bge-m3**	**R&B**	**R&L**	**Full_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	42.34	48.08	47.88	42.17	47.92	43.70	44.06	45.16
MacRAG	45.54 (**+3.20**↑)	46.68 (**-1.40**↓)	49.58 (**+1.70**↑)	46.76 (**+4.59**↑)	49.53 (**+1.61**↑)	47.94 (**+4.24**↑)	49.14 (**+5.08**↑)	**47.88** (**+2.72**↑)
$k_2 = 12$, **bge-m3**	**R&B**	**R&L**	**Ful_Ext**	**Fil**	**Full_E&F**	**R&B_Ext**	**R&B_Ext_Fil**	**Average**
LongRAG	41.53	46.96	49.58	41.60	49.57	46.76	46.33	46.05
MacRAG	46.44 (**+4.91**↑)	45.81 (**-1.15**↓)	51.02 (**+1.44**↑)	46.70 (**+5.10**↑)	51.54 (**+1.97**↑)	50.25 (**+3.49**↑)	49.50 (**+3.17**↑)	**48.75** (**+2.70** ↑)

Table 7: Extensive experimental results for comparing LongRAG vs. MacRAG+LongRAG across HotpotQA, 2WikimultihopQA, and Musique datasets with two rerankers "*marco-miniLM*" and "*bge-m3*" via GPT-4o and F1-score. The experiments conducted with the same hyper-parameter ($k_1 = 100$, $k_2 = 7$) and ($k_1 = 100$, $k_2 = 12$) which is the reported best parameter of LongRAG (Zhao et al., 2024). The columns represent various evaluation settings: R&B (Retrieval and Base), R&L (Retrieval and Long), Full_Ext (Extraction from Full Document), Fil (Filtering), Full_E&F (Extraction and Filtering combined), and R&B_Ext (Extraction from Top-k_2 Chunks). Gains from applying MacRAG to LongRAG (E&F) are displayed in parentheses with absolute gains on F1-scores and relative percentages of the improvements.