DMG-RAG: Dynamic Multi-Grained Retrieval Augmented Generation for Multi-Hop QA

Yu Pei[1,2,3,4], Mieradilijiang Maimaiti[1,2,3,4,*], Yi Chen[1,2,3,4], Wu Le[5], Zhuofei Xie[5], Jiawei Chen[5]
[1]School of Computer Science and Technology, Xinjiang University
[2]Xinjiang Laboratory of Multi-Language Information Technology, Xinjiang University
[3]Xinjiang Multilingual Information Technology Research Center
[4]Joint International Research Laboratory of Silk Road Multilingual Cognitive Computing
No. 777, Huarui Street, Shuimogou District, Urumqi 830017, Xinjiang, China
[5]Integrated Laboratory for Space, Air, and Ground Systems
Email: peiyu@stu.xju.edu.cn, miradeljan51@xju.edu.cn, 107552501327@stu.xju.edu.cn,
alaia@richnetworks.hk, zhx34@pitt.edu, syscjw@zohomail.cn

***Abstract*—Retrieval-Augmented Generation (RAG) improves multi-hop question answering by grounding large language models (LLMs) in retrieved evidence, yet its effectiveness is highly sensitive to chunk granularity and context construction under a fixed token budget. Sentence-level chunks can be precise but often fragment evidence chains, while paragraph- and document-level chunks provide broader coverage at the risk of introducing redundancy and irrelevant content. We propose DMG-RAG, a query-aware multi-grained RAG framework that coordinates chunking, retrieval, and budgeted context construction with two lightweight modules. We build aligned indices on the same corpus at three simple granularities (sentence, paragraph, and document). A query-conditioned *Router* dynamically allocates stage-1 retrieval quota set across granularities to shape an adaptive candidate pool for each query. Given the candidate pool, a *Budgeter* performs budget-constrained evidence selection by jointly considering relevance, chunk length, and redundancy, and assembles a compact evidence set via a deterministic greedy packing procedure guided by a learned scoring function. Experiments on HotpotQA, 2WikiMultiHopQA, and MuSiQue demonstrate that DMG-RAG consistently outperforms strong baselines under EM and F1, yielding substantial average gains (up to 7–8 points). The code is publicly available[1].**

***Index Terms*—multi-grained RAG, evidence set selection, multi-hop QA**

I. Introduction

Retrieval-Augmented Generation (RAG) has emerged as a practical framework that couples Large Language Models (LLMs) with an external retrieval module to tackle knowledge-intensive tasks such as question answering, translation, summarization and information extraction [1]. By retrieving task-relevant evidence from a corpus (often via dense retrieval [2]), RAG enables LLMs to access up-to-date and task-specific information beyond their fixed parametric knowledge, improving factuality and controllability in real-world scenarios.

Despite substantial progress, RAG performance is highly sensitive to the "chunking–retrieval–selection" pipeline: how the corpus is segmented, how candidates are retrieved, and how evidence is assembled for generation. Many systems rely on fixed chunking and retrieval settings with simple heuristics that overlook question heterogeneity. Fine-grained queries often need sentence-level precision, while broader or multi-hop queries benefit from paragraph- or document-level context to preserve coherence. A single strategy can therefore introduce redundant, low-density context or fragment key evidence and break reasoning chains. This motivates query-aware schemes that coordinate retrieval across granularities and construct more appropriate supporting evidence per question.

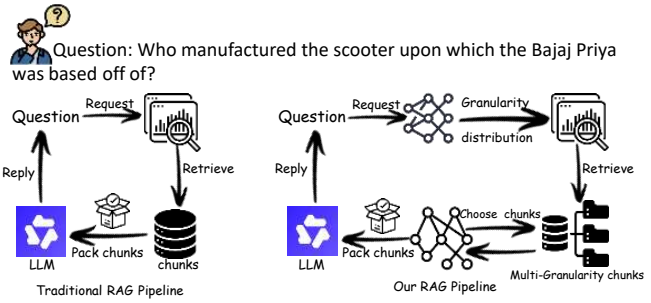


Fig. 1: Pipeline comparison between a traditional retrieve-then-read RAG pipeline and our Dynamic Multi-Grained RAG pipeline.

As illustrated in Fig. 1, we view the performance of RAG as the outcome of a coupled pipeline: how the corpus is fragmented, how candidates are retrieved, and how evidence is selected and organized in the final context for the LLM. A conventional retrieve-then-read design uses a fixed setting for all queries (e.g., one granularity and rank-based concatenation), which may be less aligned with heterogeneous multi-hop needs on HotpotQA [3], 2WikiMultiHopQA [4], and MuSiQue [5]. Previous work also suggests that the most suitable retrieval unit can vary between queries [6], motivating a more query-adaptive design for both evidence units and composition.

Concretely, our pipeline (right of Fig. 1) maintains three aligned corpus views — sentence, paragraph, and document —built from the same evidence collection, keeping chunking

*Corresponding author.
[1]https://github.com/zza-zaa/DMG-RAG

simple and update-friendly. A lightweight query-conditioned *Router* outputs a granularity distribution and maps it to stage-1 retrieval quota set across the three indices, shaping the candidate pool without selecting specific chunks. Given this pool, a *Budgeter* constructs the evidence set under a fixed context budget by combining relevance signals, chunk length, and redundancy, and applying a learned scorer with deterministic greedy packing to select a compact, complementary context. This "routing then packing" decomposition separates candidate formation from within-budget evidence composition, improving interpretability and reproducibility. We evaluate the system with `Qwen2.5-14B-Instruct` as generator [7] and compare against four strong baselines.

The main contributions are as follows:

- We propose DMG-RAG, a query-adaptive multi-grained RAG framework for multi-hop QA (sentence/paragraph/document).
- We introduce a lightweight Router+Budgeter: the Router allocates stage-1 top-k quota set across granularities, and the Budgeter scores candidates for deterministic budget-aware greedy packing.
- We conduct EM/F1 experiments and controlled ablations on HotpotQA, 2WikiMultiHopQA, and MuSiQue to isolate when router and budgeter help.

II. Related Work

A. Chunking and Granularity for RAG

Chunking defines the unit of indexing and retrieval in RAG, directly shaping what evidence can be surfaced for answering. Early RAG systems retrieve Wikipedia passages and generate condition on them [1], a practice echoed by dense retrievers such as DPR that segment documents into passages before encoding and indexing [2]. Previous analyses further show that segmentation strategies and granularity choices can substantially affect the effectiveness of RAG [8]. Granularity also impacts how evidence is distributed (fragmented facts vs. broader context): overly fine units may split co-referent facts, while overly coarse units may introduce weakly related content and blur salient cues. Recent work explores multi-granularity designs, including hierarchical retrieval with recursive clustering/summarization [9], query-conditioned granularity routing [6], and structure-aware fragmentation with LMs [10]. Cross-granularity frameworks such as FreeChunker further aim to standardize fragmentation across granularities for evaluation and reuse [11]. Although promising, these approaches can add preprocessing or hierarchical structure that complicates maintenance and updates as corpora evolve [8].

B. Retrieval, Reranking, and Chunk Selection

Modern RAG pipelines are typically multi-stage: a fast retriever builds a candidate pool, and reranking/selection determines what evidence is retained. Recent work shows that stronger selection can improve downstream generation, including structured evidence composition [12], information-gain-based reranking and filtering, and zero-shot reranking via "answer scent" signals [13]. Complementary studies analyze when and why selection helps [14] and empirically evaluate LLM rerankers [15]. Methodologically, reranking continues to evolve through multi-view guidance [16], fine-grained relevance scoring for zero-shot rankers [17], and efficient listwise reranking [18]. Beyond reranking, context construction can be improved via compression or budget-aware subset selection: EXIT performs context-aware extractive compression to reduce noise under limited context budgets [19], and AdaGReS proposes redundancy-aware scoring with greedy selection for token-budgeted RAG [20]. Overall, these advances motivate the treatment of selection as a first-class component, though many systems still optimize retrieval and selection separately, rather than as a unified process.

C. Coordinating Chunking, Retrieval, and Selection

An emerging line of work argues that RAG stages should be coordinated rather than tuned in isolation. DRAGIN links retrieval activation and query formulation with the model's evolving information needs [21], while SeaKR triggers retrieval using self-aware uncertainty signals [22]. For long-context QA, LongRAG shows that fragmentation can interact with retrieval noise and downstream evidence usage [23], and LongRefiner refines long retrieved documents before evidence assembly [24]. Query-conditioned selection mechanisms such as SelectLLM further support inference-time coordination [25]. For multi-hop QA, retrieval-chain failures (e.g., lost-in-retrieval) and mitigation strategies that emphasize structured multi-step evidence acquisition have also been highlighted [26]. Collectively, these studies motivate aligning representation units, retrieval behavior, and evidence construction, yet a single reproducible framework that coordinates all three stages while remaining simple to maintain for multi-hop QA is still an open direction.

III. Method

A. Problem Setup

Given a question q and a corpus $\mathcal{C}$, retrieval-augmented generation (RAG) retrieves an evidence set $\mathcal{S} \subset \mathcal{C}$ and generates an answer $\hat{a}$ conditioned on $(q, \mathcal{S})$. In multi-hop QA, evidence is often distributed across multiple documents. A key design choice is the *chunk granularity* for retrieval. Fine-grained sentence chunks can be precise but may fragment multi-hop evidence, while coarse document chunks improve coverage but introduce noise. Under a fixed context budget B (tokens), our goal is to make the retrieval *query-adaptive* by answering two coupled questions: **(i) granularity allocation**: how to allocate stage-1 retrieval capacity across granularities; **(ii) budget-aware selection**: how to select a final subset of retrieved chunks that best supports answering q within B.

B. Overview

Figure 2 illustrates the overall architecture of DMG-RAG and the interaction between its two lightweight modules, namely the Router for query-level granularity allocation and the Budgeter for budget-aware evidence selection and packing.

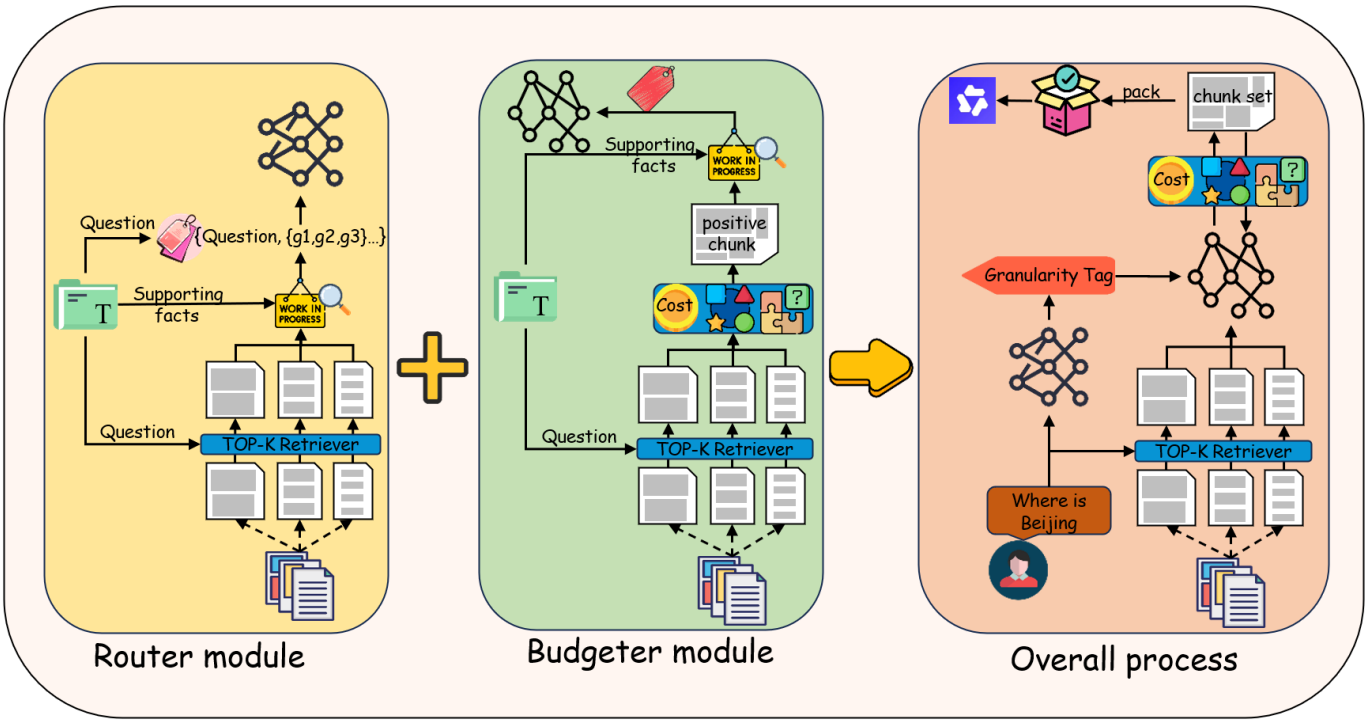


Fig. 2: Overview of our query-adaptive multi-grained RAG pipeline. The Router allocates retrieval quota set across granularities, and the Budgeter performs budget-aware evidence selection/packing before generation.

We maintain three parallel dense indices built from the same corpus at different granularities $\mathcal{G} = \{\texttt{sent}, \texttt{para}, \texttt{doc}\}$. Given q, our inference follows Algorithm 1:

Step 1: Query encoding. We encode q into a dense vector $\mathbf{e}_q \in \mathbb{R}^d$ using a bi-encoder retriever.

Step 2: Router (granularity allocation). The **Router** predicts a distribution on $\mathcal{G}$ and maps it to integer top-k quota set $(k_{\texttt{sent}}, k_{\texttt{para}}, k_{\texttt{doc}})$ for stage-1 retrieval (Algorithm 1, lines 1–5). This corresponds to the left panel "Router module" in Figure 2.

Step 3: Multi-Grained retrieval (and optional reranking). We retrieve candidates from each index according to the quota set, merge them into a global pool $\mathcal{P}(q)$, and optionally rerank the top M items using a cross-encoder (Algorithm 1, lines 6–10).

Step 4: Budgeter (budget-aware evidence selection). Given $\mathcal{P}(q)$ and the budget B, the **Budgeter** scores/filters candidates and produces a final evidence set $\mathcal{S}(q)$ (Algorithm 1, lines 11–15). If Budgeter is disabled, we fall back to a deterministic greedy packing. This corresponds to the middle panel "Budgeter module" and the packing block in the right panel "Overall process" in Figure 2.

Step 5: Generation. We build the generator prompt using the packed context and let an LLM produce $\hat{a}$ (Algorithm 1, lines 16–18).

Algorithm 1 query-adaptive multi-granularity RAG

Require: Question q; indices for `sent/para/doc`; Router (optional); Budgeter (optional); context budget B; retrieval sizes K/T.
Ensure: Predicted answer $\hat{a}$.

1: **if** Router enabled **then**
2: $(k_{\texttt{sent}}, k_{\texttt{para}}, k_{\texttt{doc}}) \leftarrow \mathsf{Router}(q)$
3: **else**
4: $(k_{\texttt{sent}}, k_{\texttt{para}}, k_{\texttt{doc}}) \leftarrow \mathsf{DefaultQuota}(K)$
5: **end if**
6: $\mathcal{P} \leftarrow \mathsf{Retrieve}(q, k_{\texttt{sent}}, k_{\texttt{para}}, k_{\texttt{doc}})$
7: $\mathcal{P} \leftarrow \mathsf{MergeSort}(\mathcal{P})$
8: **if** Reranker enabled **then**
9: $\mathcal{P} \leftarrow \mathsf{RerankTopM}(q, \mathcal{P})$
10: **end if**
11: **if** Budgeter enabled **then**
12: $\mathcal{S} \leftarrow \mathsf{BudgetSelect}(q, \mathcal{P}, B)$
13: **else**
14: $\mathcal{S} \leftarrow \mathsf{GreedyPack}(\mathcal{P}, B)$
15: **end if**
16: $x \leftarrow \mathsf{BuildPrompt}(\mathcal{S}, q)$
17: $\hat{a} \leftarrow \mathsf{LLMGenerate}(x)$
18: **return** $\hat{a}$

C. Multi-Grained Indexing

For each granularity $g \in \mathcal{G}$, we segment the corpus into chunks $\mathcal{C}_g = \{c_i^{(g)}\}_{i=1}^{N_g}$ and build a FAISS dense index $\mathcal{I}_g$. At inference time, we search for each index with $\mathbf{e}_q$:

$$\text{Search}(\mathcal{I}_g, \mathbf{e}_q, k_g) \rightarrow \mathcal{P}_g(q),$$

where $\mathcal{P}_g(q)$ are the top candidates k_g for granularity g.

D. Stagewise Retrieval

We use a two-stage template for controllability. Let K be the size of the stage-1 pool (sum of quota set) and T be the final retrieval size. The Router determines (k_g) with

$\sum_g k_g = K$. We retrieve $\cup_g \mathcal{P}_g(q)$, merge-sort them into $\mathcal{P}(q)$, and optionally apply a cross-encoder reranker to the top M items for efficiency.

E. Router: Dynamic Multi-Grained Top-k Quota Allocation

a) Role: The Router performs *query-level* control: it does not select evidence directly, but allocates retrieval capacity across granularities for stage-1 retrieval (Figure 2, left; Algorithm 1, lines 1–5).

b) Model: Given $\mathbf{e}_q$, we optionally concatenate low-cost descriptors (e.g., normalized budget signal and question length) and apply a lightweight MLP:

$$\mathbf{p}(q) = \mathrm{softmax}(\mathrm{MLP}([\mathbf{e}_q; \mathbf{f}_q])) \in \mathbb{R}^{|\mathcal{G}|}.$$

We map $\mathbf{p}(q)$ to an integer quota set by rounding under the constraint $\sum_g k_g = K$.

c) Stability safeguards: To ensure robust allocation in practice, we use: (i) **base quotas** to avoid starving any granularity; (ii) optional **prior mixing** with a user-specified prior share; (iii) a **cap** on over-allocating to the finest granularity. These safeguards make ablations comparable and avoid degenerate allocations.

F. Budgeter: Budget-Aware Evidence Scoring and Packing

a) Role: The Budgeter performs *within-budget evidence selection* given the candidate pool $\mathcal{P}(q)$ and a token budget B (Figure 2, middle/right; Algorithm 1, lines 11–15). Unlike predicting a budget bucket, our final implementation learns a *chunk utility scorer*: it assigns each candidate chunk $c \in \mathcal{P}(q)$ a utility score conditioned on the query and then constructs the final evidence set $\mathcal{S}(q)$ by packing high-utility chunks under budget B.

b) Chunk scoring model: For each candidate chunk c, we form a lightweight feature vector from signals already available at inference time, including the query embedding $\mathbf{e}_q$ and inexpensive chunk-side attributes (e.g., granularity id, dense retrieval score, optional rerank score, and token length). A small MLP outputs a scalar utility:

$$s(c, q) = \mathrm{MLP}_b\big([\mathbf{e}_q; \mathbf{f}(c)]\big),$$

where $\mathbf{f}(c)$ denotes the chunk-side features. This design keeps Budgeter lightweight and decouples learning from discrete set selection.

c) Budget-constrained packing: Given utilities $\{s(c, q)\}$, we deterministically build $\mathcal{S}(q)$ using a budget-aware packing procedure: candidates are prioritized by their utilities (with optional redundancy-aware penalties) and are added if the total token length remains within B. When Budgeter is disabled, we return to the same deterministic greedy packing baseline (Algorithm 1, line 14), ensuring fair and reproducible comparisons.

G. Oracle Supervision for Router and Budgeter

We train Router and Budgeter using weak supervision derived from an *oracle policy search* on the 2Wiki training split. For each training question, we enumerate a small grid of candidate **granularity allocation policies** (used as priors for quota allocation) and, when applicable, a set of **budget/selection options**. For each candidate configuration, we run the same stagewise retrieval and packing procedure as inference, then score the packed context using automatic signals that correlate with answerability (e.g., coverage of supporting evidence and/or answer string recall after normalization). We select the best configuration by maximizing the oracle score, with tie-breaking that prefers lower cost (smaller context when scores are equal).

a) Router labels: The selected oracle *granularity policy* induces a target distribution on $\{\texttt{sent}, \texttt{para}, \texttt{doc}\}$, which is used as a soft label $\mathbf{y}_r(q)$ for Router training.

b) Budgeter labels: For Budgeter training, the oracle search additionally yields a *budget-aware supervision signal* for candidate chunks under the same budget B used in inference. Concretely, after the oracle selects an evidence set $\mathcal{S}^*(q) \subseteq \mathcal{P}(q)$ under budget B, we derive per-chunk targets for training the utility scorer. A simple and effective choice is a binary label $y_b(c, q) = 1[c \in \mathcal{S}^*(q)]$ indicating whether the oracle selects a candidate chunk. This transforms Budgeter learning into supervised utility prediction, while keeping inference deterministic via packing.

H. Training Objectives

a) Router objective: We train Router with soft-label cross-entropy (KL divergence):

$$\mathcal{L}_{\text{router}} = \mathrm{KL}\big(\mathbf{y}_r(q) \,\|\, \mathbf{p}(q)\big).$$

b) Budgeter objective: We train the Budgeter as a per-chunk utility predictor using oracle-derived labels. With binary targets $y_b(c, q) \in \{0, 1\}$, we use a standard logistic loss:

$$\mathcal{L}_{\text{budget}} = \sum_{c \in \mathcal{P}(q)} \mathrm{BCE}\big(\sigma(s(c, q)),\, y_b(c, q)\big),$$

where $\sigma(\cdot)$ is the sigmoid function. This objective is simple, stable, and aligns with our inference-time packing, which selects high-utility chunks under budget.

c) Protocol: Oracle labels are built once on 2Wiki training data, and Router/Budgeter are trained with mini-batch optimization for a small number of iterations. At evaluation time, we reuse the trained Router/Budgeter on dataset-specific indices and report EM/F1 on the target benchmark split.

IV. Experiments

A. Setup

a) Datasets and evaluation: We evaluate on three standard Multi-hop QA benchmarks (Table I): HotpotQA [3], 2WikiMultiHopQA [4], and MuSiQue [5], using their official train/dev/test splits and building retrieval indices over the corresponding Wikipedia-based corpora for retrieval-based

TABLE I: Dataset statistics (number of questions).

Dataset	Train	Dev	Test
HotpotQA	90K	7.4K	7.4K
2WikiMultiHopQA	113K	13K	13K
MuSiQue (Ans)	20K	2.4K	2.4K

methods. Performance is measured by Exact Match (EM) and token-level F1, computed with the standard normalization used in the official evaluation scripts.

b) Model and retrieval setup: All generation-based methods use `Qwen2.5-14B-Instruct` [7]. For retrieval-based systems, we use `bge-large-en-v1.5` for dense retrieval and `bge-reranker-large` for reranking [28], while GenGround uses `ColBERTv2.0` following its original setting [29]. Naive RAG constructs context by greedy packing of retrieved chunks, and our method additionally applies query-adaptive routing and budget-aware selection.

c) Implementation details and final training setup: Across all experiments, we keep the pipeline deterministic for reproducibility and use greedy decoding (no sampling), so temperature/top-p do not affect results. We fix the evaluation protocol to a subset of 1,500 questions with a fixed random seed, and keep the retrieval budget and stagewise sizes consistent across ablations. For retrieval, we build FAISS indices and use `bge-large-en-v1.5` for dense embeddings and `bge-reranker-large` for reranking. For training, both Router and Budgeter are trained with oracle supervision in 2Wiki using the final configuration of $N = 20{,}000$ training instances for 3 epochs (i.e., $E = 3$), with a learning rate 2×10^{-4} and a batch size of 128 unless otherwise stated.

d) Baselines: We compare our method against four representative baselines that cover "no retrieval", "standard RAG", "granularity routing", and "iterative grounding".

- Direct answers questions using `Qwen2.5-14B-Instruct` without external retrieval, serving as a no-RAG reference.
- Naive RAG implements the standard retrieve-then-generate pipeline [1], retrieving top-k text chunks and assembling them into a single context for generation.
- MoG (Mix-of-Granularity) introduces a query-conditioned router to select retrieval granularity for RAG [6].
- GenGround (Generate-then-Ground) alternates between generating intermediate single-hop information needs and grounding them with retrieval, tailored for multi-hop QA [27].

B. Main Results and Analysis

Table II and Table III report EM and F1 on three multi-hop QA benchmarks. Across all methods, we use `Qwen2.5-14B-Instruct` as the generator to isolate the impact of retrieval and evidence construction. DIRECT answers without retrieval, while RAG-based baselines differ in how they chunk, retrieve, and select evidence. In addition, NAIVE RAG includes greedy packing refinement, and our approach further introduces query-adaptive *granularity allocation* (Router) and *budget-aware evidence selection/packing* (Budgeter) to better coordinate the retrieval pipeline end-to-end.

a) HotpotQA: In HotpotQA, DIRECT is substantially weaker than retrieval-based methods, indicating that multi-hop questions often benefit from external evidence rather than relying purely on the parametric knowledge of the LLM. NAIVE RAG provides a strong baseline, showing that retrieving and greedily packing high-ranked chunks is already effective on this benchmark. Our method further improves over NAIVE RAG, consistent with the intended roles of our components: (i) the Router adapts the retrieval mix across sentence/paragraph/document views to match question needs, and (ii) the Budgeter promotes a more supportive *set* of chunks (less repetitive and more complementary) than purely score-sorted packing. In contrast, MOG and GENGROUND are less effective than NAIVE RAG in our controlled setting, suggesting that routing or iterative grounding alone does not guarantee gains unless evidence selection is aligned with the candidate pool and the final packing decision.

b) 2WikiMultiHopQA: On 2WikiMultiHopQA, NAIVE RAG drops noticeably compared to HotpotQA, reflecting the increased difficulty of assembling cross-document evidence chains under a fixed retrieval-and-pack recipe. Our method yields a much larger improvement on this dataset, which matches the motivation of our design: the Router can shift stage-1 retrieval toward granularities that better preserve multi-hop context when needed, and the Budgeter explicitly favors evidence combinations that are jointly useful (e.g., covering different sources/entities) rather than individually high-scoring but redundant chunks. In general, the averaged EM/F1 over HotpotQA and 2WikiMultiHopQA also improves, indicating that the gains are not restricted to a single benchmark.

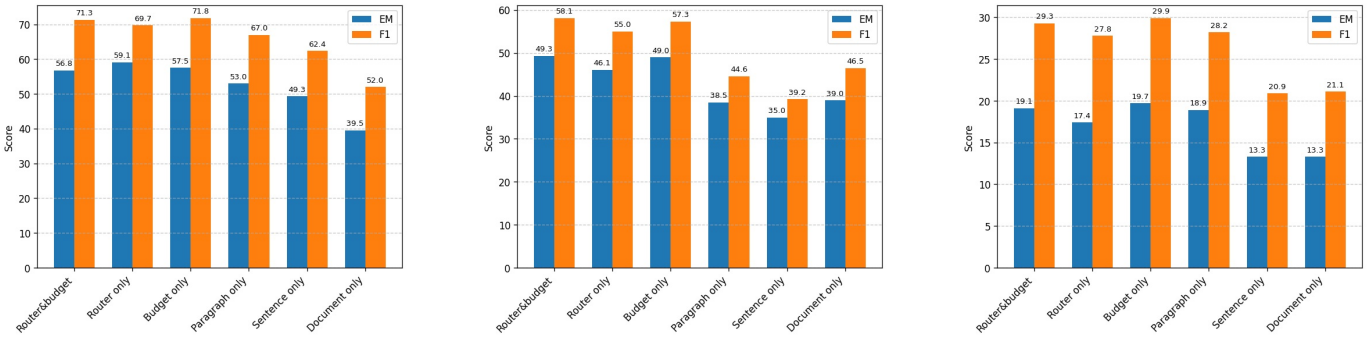
c) MuSiQue: On MuSiQue, our method consistently improves over NAIVE RAG, though the margin is smaller than on 2WikiMultiHopQA. A plausible explanation is that the compositional construction of MuSiQue can place stronger emphasis on the initial recall quality: when key evidence is missing from the candidate pool, selection-level improvements become less pronounced. This suggests that future improvements may come from strengthening the recall mechanism (e.g., retriever adaptation or query expansion) while keeping the same Router+Budgeter coordination mechanism.

C. Sensitivity to Training Samples and Training Step

a) Training sample size: We analyze how the number of training samples influences the two learned modules while keeping the rest of the pipeline unchanged. Figure 4 reports EM/F1 (%) for the Router and the Budgeter trained with sample sizes in $\{1\text{k}, 5\text{k}, 10\text{k}, 15\text{k}, 20\text{k}\}$. For the Router (Fig. 4a), performance increases steadily with more data: F1 rises from 53.7 to 55.1 and EM from 44.8 to 46.3, indicating that the router benefits from additional supervision but remains relatively stable even at smaller scales. For the

Methods	HotpotQA		2WikiMultiHopQA		Average	
	EM	F1	EM	F1	EM	F1
Direct(qwen2.5-14B) [7]	20.6	27.4	23.2	25.49	21.9	26.5
Naive RAG [1]	53.1	67.1	39	46.5	46.05	56.8
MoGRAG [6]	40.4	52.37	19	23.61	29.7	37.99
GenGroundRAG [27]	36.59	45.91	21.2	26.42	28.9	36.17
Ours	**56.8**	**71.3**	**49.3**	**58.1**	**53.05**	**64.7**

TABLE II: Answer EM and F1 (%) on HotpotQA [3] and 2WikiMultiHopQA [4]. We compare DIRECT (LLM-only, no retrieval) with RAG baselines and our query-adaptive multi-granularity retrieval with budget-aware evidence packing. **Average** reports the mean score over HotpotQA and 2WikiMultiHopQA (higher is better). Underlined values indicate the best baseline performance.

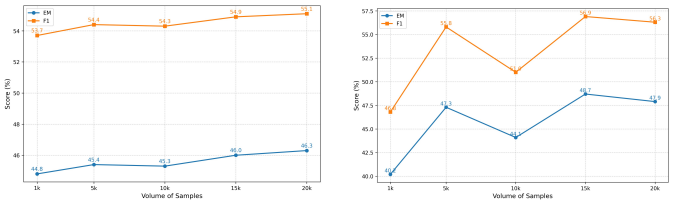


(a) Hotpot (b) 2WikiMultiHopQA (c) MuSiQue

Fig. 3: Ablation results across three multi-hop QA benchmarks: (a) HotpotQA, (b) 2WikiMultiHopQA, and (c) MuSiQue. We compare the full system (Router+Budgeter) with component variants (Router-only, Budgeter-only) and single-granularity retrieval (sentence/paragraph/document only). Bars report answer EM and F1 (%).

Methods	MuSiQue	
	EM	F1
Direct [7]	5	13.6
Naive RAG [1]	18.7	28.3
MoGRAG [6]	6.6	12.29
GenGroundRAG [27]	10.87	15.1
Ours	**19.1**	**29.3**

TABLE III: Answer EM and F1 (%) on MuSiQue [5]. We compare DIRECT (LLM-only) and several RAG baselines against our method. Underlined values indicate the best baseline performance.

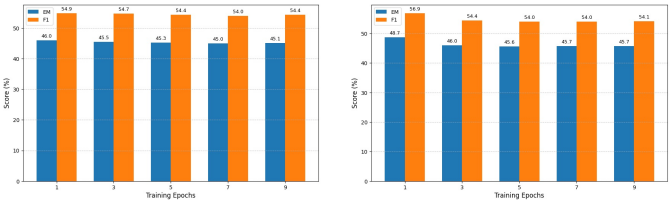


(a) Router (b) Budgeter

Fig. 4: Sensitivity to training set size (number of training questions) for the learned components. We report EM/F1 of (a) Router and (b) Budgeter when varying the training sample size from 1k to 20k (other settings fixed).

Budgeter (Fig. 4b), the effect is stronger and less monotonic: performance improves dramatically from 1 to 5k, dips around 10k, and peaks at 15k (F1=56.9, EM=48.7) before slightly decreasing at 20k. Overall, these results suggest that the Budgeter is more data-sensitive, and moderate-to-large training sets are important for robust evidence selection behavior.

b) Training iterations: We study the impact of training iterations while fixing the training set size to 15k. Figure 5 reports EM/F1 (%) for iteration steps in $\{1, 3, 5, 7, 9\}$. For the Router (Fig. 5a), performance is stable across iterations: EM varies within 45.0–46.0 and F1 within 54.0–54.9, with no consistent gain from longer training (e.g., the best at 1 training iteration achieves EM=46.0 and F1=54.9). For the Budgeter (Fig. 5b), the best performance is achieved in one iteration step (EM=48.7, F1=56.9), while additional iterations lead to a noticeable drop and then plateau (EM $\approx$ 45.6–46.0; F1 $\approx$ 54.0–54.4). Therefore, unless otherwise stated, we use one training iteration as the default configuration for stability and efficiency.



(a) Router (b) Budgeter

Fig. 5: iter step sensitivity of the learned components (training set size = 15K). (a) Router and (b) Budgeter are trained with the same pipeline and supervision protocol and evaluated by EM/F1 (%). Performance typically peaks at one iter step, while additional iter steps tend to bring diminishing returns; the Budgeter is notably more sensitive, showing a clearer drop from iter step=1 to later iter steps.

D. Ablation Study

a) Component ablation and granularity study: Figure 3 reports an ablation study on HotpotQA, 2WikiMultiHopQA, and MuSiQue. Overall, combining Router+Budgeter yields the strongest or near-strongest performance across datasets, indicating that coordinating (i) query-adaptive granularity allocation and (ii) evidence set selection is generally beneficial for multi-hop QA where evidence may be distributed across documents.

b) Router vs. Budgeter: Across datasets, Budgeter-only is typically more competitive than Router-only, supporting the intuition that multi-hop QA benefits from selecting a complementary *set* of evidence rather than relying on point-wise relevance alone. The Router provides complementary gains by shaping a better candidate pool, and combining both components yields the most consistent improvements.

c) Multi-Grained vs. single granularity: Single-granularity baselines show clear gaps: sentence-only retrieval tends to underperform due to fragmented evidence chains, while document-only retrieval can improve coverage but may introduce more noise. Paragraph-only lies in between but remains behind the coordinated approach, motivating our use of multiple simple granularities plus learned selection.

E. Case Study

Table IV presents two representative MuSiQue questions. For Q1, only our method outputs the exact gold answer, while DIRECT, MOG, and GENGROUNDRAG predict "NO" and NAIVERAG returns an incomplete entity name "Pasifika". This suggests that fixed retrieval/selection strategies (or generation without retrieval) may miss the precise supporting mention needed for an exact match when multi-hop evidence must be connected under limited context. In contrast, our Router shapes a more query-aligned candidate pool across sentence/paragraph/document indices, and the Budgeter selects a compact, complementary evidence set via deterministic packing, reducing distraction from irrelevant chunks. For Q2, most methods produce a nearly correct answer, with the differences mainly due to whether the acronym "ACM" is included, illustrating the sensitivity of EM to surface-form variations even when token-level F1 remains high. Overall, these cases support decoupling query-level granularity routing from within-context evidence composition to improve robustness and reduce catastrophic failures on multi-hop questions.

Item	Content
Q1	What is the name of the festival in New Zealand and the festival it is based on?
Gold	*Pasifika Festival and the Pasifika Festival*
Direct	*NO*
NaiveRAG	*Pasifika*
MoG	*NO*
GenGroundRAG	*NO*
Ours	*Pasifika Festival and the Pasifika Festival*
Q2	What is the association for computing machinery?
Gold	*Association for Computing Machinery*
Direct	*Association for Computing Machinery (ACM)*
NaiveRAG	*Association for Computing Machinery (ACM)*
MoG	*Association for Computing Machinery*
GenGroundRAG	*NO*
Ours	*Association for Computing Machinery (ACM)*

TABLE IV: Case study on two questions from MuSiQue, comparing Direct generation, retrieval-based baselines, and our Router+Budgeter pipeline. Our method succeeds on Q1 where several baselines output "NO" or a partial entity, and remains competitive on Q2 where answers differ mainly in whether the acronym is included.

V. CONCLUSION AND FUTURE WORK

We propose a query-adaptive multi-grained RAG framework for multi-hop question answering that coordinates evidence *units* and evidence *composition* under a fixed context budget. By maintaining aligned sentence/paragraph/document indices, a lightweight Router allocates stage-1 retrieval quota set across granularities, while a Budgeter performs budget-aware scoring and deterministic greedy packing to construct a compact, complementary context for generation. Across HotpotQA, 2WikiMultiHopQA, and MuSiQue, our approach consistently improves over DIRECT and strong RAG baselines, with the largest gains on harder multi-hop cases where both candidate formation and set-level selection matter. Future work will refine supervision and training for routing/packing, strengthen retrieval recall, and extend evaluation to broader datasets and dynamic corpora.

VI. ACKNOWLEDGMENT

We sincerely thank our collaborators for their valuable support and the anonymous reviewers for their insightful comments on this work. This work was supported by the National Natural Science Foundation of China (Grant No.62406316), and the Xinjiang 'Tianchi Talent' Recruitment and Introduction Program (awarded to Mieradilijiang Maimaiti).

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[3] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2369–2380.

[4] X. Ho, A.-K. Duong Nguyen, S. Sugawara, and A. Aizawa, "Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 6609–6625.

[5] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Musique: Multihop questions via single-hop question composition," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 539–554, 2022.

[6] Z. Zhong, H. Liu, X. Cui, X. Zhang, and Z. Qin, "Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation," in *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*, 2025, pp. 5756–5774.

[7] Qwen Team, "Qwen2.5 technical report," 2024.

[8] X. Wang, Z. Wang, X. Gao, F. Zhang, Y. Wu, Z. Xu, T. Shi, Z. Wang, S. Li, Q. Qian, R. Yin, C. Lv, X. Zheng, and X. Huang, "Searching for best practices in retrieval-augmented generation," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2024, pp. 17 716–17 736.

[9] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," 2024.

[10] A. Jain, P. Aggarwal, and A. Saladi, "Autochunker: Structured text chunking and its evaluation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*. Association for Computational Linguistics, 2025, pp. 983–995.

[11] W. Zhang, Y.-H. Jiang, and Y. Wu, "Freechunker: A cross-granularity chunking framework," 2025.

[12] H. Sun, H. Cai, Y. Li, X. Fan, X. Wei, S. Wang, Y. Zhang, and D. Yin, "Enhancing retrieval-augmented generation via evidence tree search," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 24 116–24 127.

[13] A. Abdallah, J. Mozafari, B. Piryani, and A. Jatowt, "Asrank: Zero-shot re-ranking with answer scent for document retrieval," in *Findings of the Association for Computational Linguistics: NAACL 2025*. Albuquerque, New Mexico: Association for Computational Linguistics, Apr. 2025, pp. 2950–2970.

[14] X. Li and J. Ouyang, "How does knowledge selection help retrieval augmented generation?" in *Findings of the Association for Computational Linguistics: EMNLP 2025*. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 4104–4121.

[15] A. Abdallah, B. Piryani, J. Mozafari, M. Ali, and A. Jatowt, "How good are LLM-based rerankers? an empirical analysis of state-of-the-art reranking models," in *Findings of the Association for Computational Linguistics: EMNLP 2025*. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 5693–5709.

[16] J. Na, J. Kwon, E. Choi, and J. Lee, "Multi-view-guided passage reranking with large language models," in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 28 693–28 706.

[17] Y. Zhuang, A. Fiore, R. van der Goot, T. Nguyen, A. Faber, K. Kersting, and I. Gurevych, "Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 336–347.

[18] R. Gangi Reddy, J. Doo, Y. Xu, M. A. Sultan, D. Swain, A. Sil, and H. Ji, "FIRST: Faster improved listwise reranking with single token decoding," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 8642–8652.

[19] T. Hwang, S. Cho, S. Jeong, H. Song, S. Han, and J. C. Park, "EXIT: Context-aware extractive compression for enhancing retrieval-augmented generation," 2024.

[20] C. Peng, B. Wang, Z. Long, and J. Sheng, "Adagres: Adaptive greedy context selection via redundancy-aware scoring for token-budgeted rag," 2025.

[21] W. Su, Y. Tang, Q. Ai, Z. Wu, and Y. Liu, "DRAGIN: Dynamic retrieval augmented generation based on the real-time information needs of large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 12 991–13 013.

[22] Z. Yao, W. Qi, L. Pan, S. Cao, L. Hu, L. Weichuan, L. Hou, and J. Li, "SeaKR: Self-aware knowledge retrieval for adaptive retrieval augmented generation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 27 022–27 043.

[23] Q. Zhao, R. Wang, Y. Cen, D. Zha, S. Tan, Y. Dong, and J. Tang, "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*. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 22 600–22 632.

[24] J. Jin, X. Li, G. Dong, Y. Zhang, Y. Zhu, Y. Wu, Z. Li, Y. Qi, and Z. Dou, "Hierarchical document refinement for long-context retrieval-augmented generation," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 3502–3520.

[25] K. K. Maurya, K. A. Srivatsa, and E. Kochmar, "SelectLLM: Query-aware efficient selection algorithm for large language models," in *Findings of the Association for Computational Linguistics: ACL 2025*. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 20 847–20 863.

[26] R. Zhu, X. Liu, Z. Sun, Y. Wang, and W. Hu, "Mitigating lost-in-retrieval problems in retrieval augmented multi-hop question answering," 2025.

[27] Z. Shi, W. Sun, S. Gao, P. Ren, Z. Chen, and Z. Ren, "Generate-then-ground in retrieval-augmented generation for multi-hop question answering," 2024.

[28] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J. Nie, "C-Pack: Packed resources for general chinese embeddings," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24)*. Association for Computing Machinery, 2024, pp. 641–649.

[29] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, and M. Zaharia, "Colbertv2: Effective and efficient retrieval via lightweight late interaction," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022.