# Hierarchical Retrieval-Augmented Generation Framework with Multi-Level Granularity

Ran Arino [†], Nipun Ravindu Fernando Warnakulasuriya [†], Umer Aftab [†]

[†] School of Software Design & Data Science, Seneca Polytechnic, Toronto, Ontario, Canada

**Abstract**

Large Language Models (LLMs), while continuously advanced, face challenges with context limitations, factual inaccuracies, and knowledge staleness. Retrieval-Augmented Generation (RAG) mitigates these challenges by incorporating vector search mechanisms to external knowledge. However, traditional flat retrieval methods struggle with efficiency and effectiveness, especially across large or complex document sets. This paper introduces a hierarchical RAG framework leveraging multi-level semantic granularity to address these shortcomings. Our framework constructs a four-layer retrieval phase, comparison document clusters, document summaries, chunk clusters, and chunks. This coarse-to-fine retrieval strategy navigates semantic hierarchy to prune the search space and locate relevant information. Our experiment results reveal mixed performance, but our framework demonstrated improved generation accuracy, especially for complex multi-document searching tasks. This work suggests that while hierarchical structures may offer benefits for both retrieval and generation performance in specific complex RAG scenarios, careful consideration of the observed trade-offs with retrieval breadth and the potential risks of losing contextual information due to clustering or dimensionality reduction are necessary.

**Keywords:** Retrieval-Augmented Generation, Information Retrieval, Vector Searching, Sentence Transformer, Natural Language Processing

## 1. Introduction

Large Language Models (LLMs) have showcased remarkable capabilities in various Natural Language Processing (NLP) tasks. The recent advancement in Artificial Intelligence (AI)-driven applications and services has huge potential to boost optimization and productivity. However, LLMs are notorious for context window limitation [1] and their tendency to produce hallucinations [2]. Furthermore, the model's fine-tuning to maintain the up-to-date knowledge is costly [3], requiring sufficient computational resources, training time, and budgets. To address those challenges stemming from knowledge-intensive features, Retrieval-Augmented Generation (RAG), initiated by Lewis et al. [4], has emerged as a promising solution by incorporating external information during LLM's inference phase.

The FAISS index [5], introduced by the Facebook Research team, has brought a significant contribution as an optimized vector similarity search framework, particularly by Inverted File (IVF) and Hierarchical Navigable Small Word (HNSW). The IVF approach clusters vectors and prunes irrelevant ones, achieving the efficient retrieval processing for a large amount of vector search. On the other hand, HNSW utilizes hierarchical graphs by constructing a multi-layer structure where connections jump to different layers. Search initiates at the top layer and progressively moves down to lower layers for neighbor discovery, enabling efficient traversal [6].

Although the RAG systems have contributed to enhancing reliability and truthfulness of LLM generation, the traditional approach of employing fixed-length chunks (i.e. specific number of words or tokens) for text segmentation has several limitations. Recent studies have explored different granularity levels for retrieval units, including propositions [7] and logic units [8]. Those novel approaches have demonstrated that the choice of retrieval granularity significantly affects the retrieval performance in the RAG system.

However, challenges are identified in the traditional flat or linear retrieval approaches, which are searching relevant text chunks through directly applying a given query. For example, a study highlights the ineffectiveness and inefficiency of a single retrieval process across a vast amount of context chunks [9]. Specifically, a few shot retrieval restricts the ability to capture various sections across documents for handling thematic questions [1].

To address these limitations, recent research has focused on hierarchical structures for retrieval, particularly organizing information in tree-based [1] or graph-based structures [10]. These approaches primarily aim to capture semantic relationships and logical structure across documents, simultaneously optimizing the retrieval processes[1]. This paper explores the semantic index-based hierarchical searching architecture to enhance retrieval effectiveness and efficiency of the RAG systems.

## 2. Related Works

### 2.1 Vector Search

Vector similarity search is a key component to achieve efficient retrieval of relevant textual information based on high-dimensional vector embeddings. FAISS [5] is a prominent library offering various indexing strategies. For instance, IVF conducts efficient pruning of irrelevant vectors during search, according to vector cluster centroids. In this case, those cluster centroids are calculated as the simple arithmetic means of all vectors assigned to each cluster. However, this method does not guarantee that those centroids capture the semantic details within textual data clusters [11]. Moreover, it is questionable if those centroids act as a coarse information of finer-grained vectors within clusters. This raises concerns over their ability to act as cluster representatives, compared with more contextually-rich document summaries [12].

Alternatively, HNSW is based on a hierarchical graph-based approach, navigating vector nodes at different layers and aiming for a more efficient retrieval system. Despite its advantages in retrieval processing speed, each top layer node is selected randomly, without considering granularity-based filtering or pruning. This random selection can enhance the efficiency, at its cost, the retrieval effectiveness could be undermined, particularly in multi-document searching. Furthermore, HNSW has potentially been evaluated by the simple datasets. Given the increasing complexity of the emerging real-world datasets [13], this approach may hinder the performance of downstream tasks of RAG, especially in scenarios requiring multi-hop reasonings at the retrieval phase.

### 2.2 Text Segmentation and Retrieval Granularity

Text segmentation, or document chinking, is a crucial preprocessing of RAG systems. The traditional RAG employs the fixed-length chunking strategy, such as a specific number of tokens [14] or words [3]. However, various studies have explored different chunk sizes, or retrieval granularities, to optimize the performance of the retrieval and generation in RAG systems.

Chen et al. [7] proposed the propositional-level retrieval as a finer-grained approach. Researchers define the proposition as atomic expressions in text, characterized by unique meaning, conciseness, and self-containment. They are motivated to mitigate potential performance bottlenecks caused by coarser-grained approaches, such as irrelevant information in

[1] Our GitHub Repository here

a passage and compounded context in a sentence. The study validates a performance gain in question-answer tasks, highlighting the importance of tuning retrieval granularity to harmonize completeness and compactness within each chunk.

Although the proposition-based chunks are well-performed when requiring factual answers pertaining to specific details, their finest-grained units hinder the generation of sequential explanations. For instance, each proposition can articulate atomic pieces of factual information, but they should be processed through an accurately ordered workflow. A new level of granularity, termed “logic units”, is presented, underscoring a more structured and interconnected representation by prioritizing logical and coherent expressions [8]. The study demonstrates the enhanced performance on downstream tasks, particularly in how-to questions that necessitate the linear task sequences and multi-stage decision processes. However, each logic unit is generated by the prompt-based LLM, requiring huge computational resources. Moreover, the efficacy of the proposed approach has been verified exclusively on how-to question style.

Considering coherence and dynamic sizing, several researches have explored semantically-related chunks as another retrieval unit. Qu et al. [15] proposed two approaches; one for identifying breakpoints from semantic distance, and the other for forming topic-centric chunk clusters from semantic similarity. However, both approaches have shown no significant advantages compared to the fixed-size chunking in the cost of computational complexities. LumberChunker [16] instructs the LLM model to optimally select segmentation points, particularly designed for narrative texts. Despite achieving top QA accuracy among baselines, the reliance on the LLM models for the retrieval units remains significant computational overhead. On the other hand, Meta-Chunking [17] utilizes the inherent capability of the LLM to calculate perplexity scores, without relying on the prompt-based LLM generation. This approach achieves less computational overload than LumberChunker, simultaneously enhancing the retrieval efficiency by considering logical relationships between sentences. For example, the study highlights the improved performance compared to fixed-length and similarity-based chunking methods, requiring only 45.8% of the processing time of the prompt-based LLM approach.

## 2.3 Mixed of Different Granularity Levels

The information retrieval techniques have further evolved to incorporate multiple levels of granularity, including passages, sentences, and propositions. Although Meta-Chunking [17] dynamically adjusts the granularity between passages and sentences, novel retrieval-focused architectures explore the proposition-level units, query-driven dynamic granularity, or progressive coarse-to-fine retrieval.

MixGR [18] is proposed to enhance the generalization of dense retrievers in a scientific domain by employing complementary granularity in both queries and documents. The method integrates four levels of granularity (original queries, subqueries, full documents, and propositions) and applies Reciprocal Rank Fusion for score aggregation. The study highlights the improved retrieval accuracy in knowledge intensive tasks requiring complex queries and factual information.

Mix-of-Granularity (MoG) [19] dynamically determines the optimal granularity levels based on the input query. Specifically, a trained router predicts weights for pre-defined granularity levels, and those weights are integrated with sentence similarity during retrieval. Although MoG demonstrates significant performance and well-grounded responses in the scientific domain, MoG is not designed for multi-document or cross-database search. Moreover, the retrieval overload would exponentially increase due to searching for all the snippets generated for each predefined granularity level.

FunnelRAG [9] notably implements the progressive paradigm with a coarse-to-fine granularity approach. The proposed architecture specifies three levels of granularity in a retrieval pipeline; clustered documents (4K tokens), document-level units (1K tokens), and passage-level units (around 100 tokens). The pipeline progressively shifts a sparse retriever (for document-clusters), cross-encoder (for documents), and list-wise model (for passage). Integrating these two approaches achieves both retrieval efficiency and effectiveness, albeit requiring manual hyperparameter tuning and careful data flow design for each dataset or task.

## 2.4 Hierarchical Retrieval

Since the traditional retrieval methods, categorized to a flat or linear retrieval, highly rely on the sentence similarity, those methods generally ignore the hierarchical relationships between the retrieval units, including documents, passages, and propositions. To capture the inherent structure or handle the multi-hop reasoning, the hierarchical retrieval approaches have recently evolved.

Joint optimization of TRee-based index and query encoding (JTR) [20] departs from the traditional dense retrieval to the hierarchical approach. To construct tree-based structure, the k-mean algorithm recursively clusters document embeddings into semantic groups. Therefore, individual documents are represented as leaf nodes, and the cluster centroids serve as tree nodes, which are the initial targets for similarity search. This approach prunes irrelevant semantic groups during the retrieval phase, leading to the significant searching optimization. Although JTR simultaneously achieves a better retrieval performance, the flexibility could be a bottleneck. For example, the study highlights the limitations of conducting large-scale web search and handling index update by adding or removing documents.

Recursive Abstractive Processing for Tree-Organized Retrieval (RAPTOR) [1] constructs a hierarchy through the repetitive cycle of embedding, clustering, and summarizing. The indexing algorithm proceeds bottom-up, starting with the finest-grained units - initially chunked text with approximately 100 tokens - representing the tree's leaf nodes. Embeddings of these chunks are clustered semantically. Summaries of each cluster by the prompted LLM undergo chunking and embedding until further clustering is no longer feasible. Two retrieval options are provided; the tree traversal selects the most similar top-k nodes until reaching the deepest nodes; the collapsed tree extracts relevant contexts from a flattened tree node. While RAPTOR demonstrates superiority over traditional retrieval methods in complex reasoning, the recursive summary generation by the LLM could pose a significant challenge in computational efficiency.

SiReRAG [21] constructs two trees to capture both similarity and relatedness to address the limitations of methods that focus on either one of these aspects. RAPTOR's indexing strategy is directly utilized as a similarity-centric tree, and the relatedness-based tree is built by extracting propositions and shared entities. Experiments on multihop reasoning tasks, SiReRAG consistently outperforms RAPTOR, highlighting the importance of synthesizing context similarity and relatedness. However, the trade-off for this improved performance is a longer inference time than RAPTOR since the retrieval pool size is expanded to a flattened node from two trees.

Hierarchical Indexing for Retrieval-Augmented Opinion Summarization (HIRO) [22] is a hierarchical indexing particularly optimized to opinion summarization, combining the strengths of extractive (attributable, scalable) and abstractive approaches (coherent, fluent). Unlike RAPTOR's repetitive LLM-based summarization for each tree, HIRO constructs a hierarchical index by capturing the prevalence of opinions at different granularity levels (i.e., the higher levels represent broader topics). While HIRO is limited to review summarization tasks, experiments demonstrate its capacity to accurately capture overall sentiment expressed across numerous opinions.

## 3. Methodologies

In this section, we will introduce our proposed Hierarchical RAG framework driven by Semantic Granularity. Our approach employs semantic clustering within a four-layer hierarchical structure, including document clusters, documents, chunk clusters, and chunks. Table 1 provides a list of all mathematical symbols and their corresponding definitions.

| **Symbol** | **Definition** |
|---|---|
| $q$ | User's input query (question) |
| $D = \{d_1, ..., d_N\}$ | Set of $N$ source documents. |
| $d_i$ | The $i$-th document in the set $D$ |
| $s_i$ | Summary generated for document $d_i$ |
| $C = \{c_{i,1}, ..., c_{i,M_i}\}$ | Set of $M_i$ text chunks within document $d_i$ |
| $c_{i,j}$ | The $j$-th chunk of the $i$-th document |
| $e_{\cdot} = E(\cdot)$ | Embedding function (sentence transformer) mapping text to vectors |
| $R(\cdot)$ | Optional dimensionality reduction function (Principal Component Analysis) |
| $C(\cdot)$ | Clustering function (K-Means or GMM) applied to a set of embeddings |
| $k_D \in K_D$ | A specific cluster identifier for document summary embedding |
| $k_{C_i} \in K_{C_i}$ | A specific cluster identifier for chunks within document $d_i$ |
| $centroid(k)$ | Centroid vector representation cluster $k$ |
| $VS_{doc\_cc}$ | Vector store containing the document summary cluster centroids |
| $VS_{doc}(k_D)$ | Vector store containing the document embeddings belong to cluster $k_D$ |
| $VS_{chunk\_cc}(d_i)$ | Vector store containing the chunk cluster centroids for document $d_i$ |
| $VS_{chunk}(k_{C_i})$ | Vector store containing the chunk embeddings belong to cluster $k_{C_i}$ |
| $sim(\cdot, \cdot)$ | Similarity function (L2 distance) between vectors |
| $top_k(\cdot)$ | Function selecting the top $k$ most similar items based on $sim(\cdot, \cdot)$ |

*Table1: Mathematical Symbols and Definitions*

Our framework comprises a four-layer hierarchical framework with semantic clustering and optional dimensionality reduction to facilitate contextually refined and navigable retrieval. Figure 1 shows the overview of this four-layer structure.

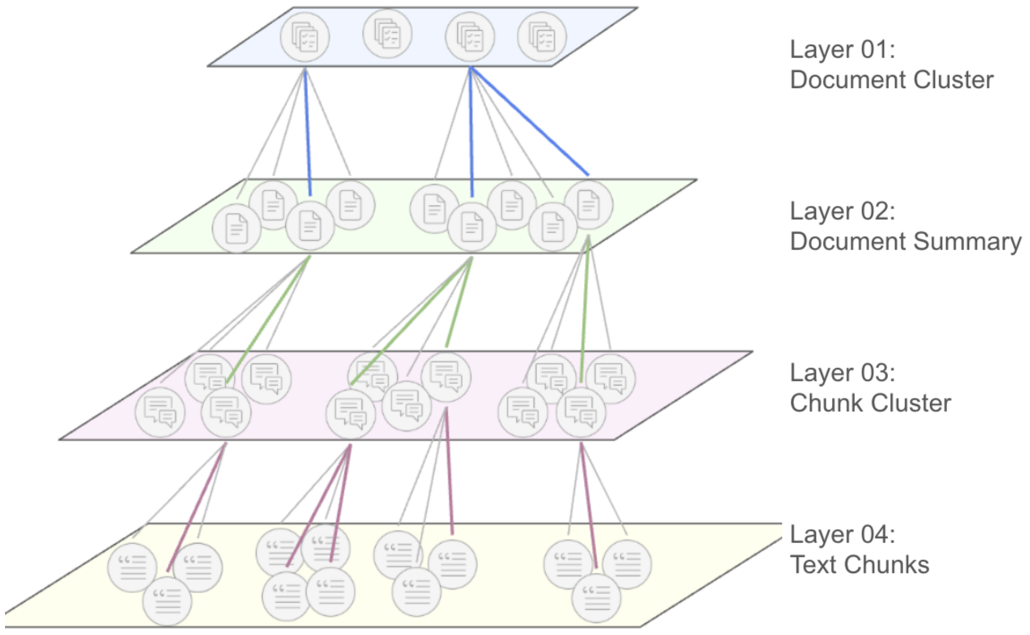

*Figure1: Overview of the Four-Layer Structure*

## 3.1 Architecture

The core architecture comprises four primary layers:

- Document Summary Centroids Layer ($VS_{doc\_cc}$): storing centroids vectors ($centroid(k_D)$) based on clustering document summary embeddings ($e_{s_i}$ or $R(e_{s_i})$). This layer offers a coarse-grained semantic map for the document corpus.
- Document Summary Layer ($VS_{doc}$): containing the actual document summary ($s_i$) and its vectors ($e_{s_i}$). Summaries are grouped based on the clustering result above. Thus, each group ($k_d$) stores a separate index ($VS_{doc}(k_D)$).
- Chunk Centroids Layer ($VS_{chunk\_cc}$): For each document($d_i$), this layer stores centroid vectors ($centroid(k_{C_i})$) derived from clustering $e_{c_{i,j}}$ or $R(e_{c_{i,j}})$ of its constituent chunks ($c_{i,j}$). This provides a medium-grained view of topics within individual documents, stored per document.
- Chunk Layer ($VS_{chunk}$): storing the individual text chunk ($c_{i,j}$) and its embedding ($e_{c_{i,j}}$). Chunks within a document ($d_i$) are grouped based on clustering results. So, each group ($k_c$) stores a separate index ($VS_{chunk}(k_{C_i})$)

## 3.2 Indexing Process

On the indexing phase, the given documents ($D$) are processed as following steps:

1. Document Summarization and Embedding: As shown in Figure 2, generate an abstractive summary ($s_i$) for each document and get its embedding $e_{s_i} = E(s_i)$.

2. Document Chunking and Embedding: As shown in Figure 3, splitting each document into chunks $C_i = \{c_{i,1}, ..., c_{i,M_i}\}$ and get its embedding $e_{c_{i,j}} = E(c_{i,j})$. Each chunk is set by approximately 100 tokens and sentence-based segmentation. Furthermore, to preserve contextual continuity between chunks, an overlap mechanism is implemented. For example, approximately 20 tokens are exactly the same contexts between the end of $c_{i,j}$ and the beginning of $c_{i+1,j}$.
3. Optional Dimensionality Reduction: Apply function $R(\cdot)$ to $e_{s_i}$ and $e_{c_{i,j}}$. The default reduction size is set as the maximum of 15 dimensions.
4. Hierarchical Clustering and Centroid Indexing:
    a. Document Level: Cluster the set of $e_{s_i}$ or $R(e_{s_i})$ using $C(\cdot)$. Then, get $centroid(k_D)$ and store them in $VS_{doc\_cc}$.
    b. Chunk Level: Cluster the set of $e_{c_{i,j}}$ or $R(c_{i,j})$. Then, compute and store $centroid(k_{C_i})$ in $VS_{chunk\_cc}(d_i)$.
5. Clustered Data Indexing:
    a. Store document summary embedding ($e_{s_i}$) in $VS_{doc}(k_D)$ corresponding to its assigned document cluster $k_D$.
    b. Store chunk embedding ($e_{c_{i,j}}$) in $VS_{chunk}(k_{C_i})$ corresponding to its assigned document cluster $k_{C_i}$.

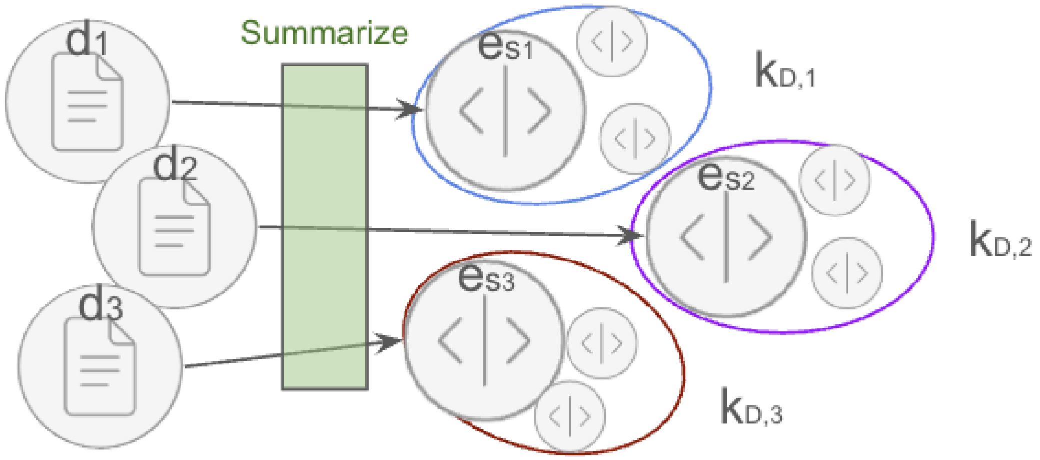

*Figure2: Document Summarization and Clustering Process*

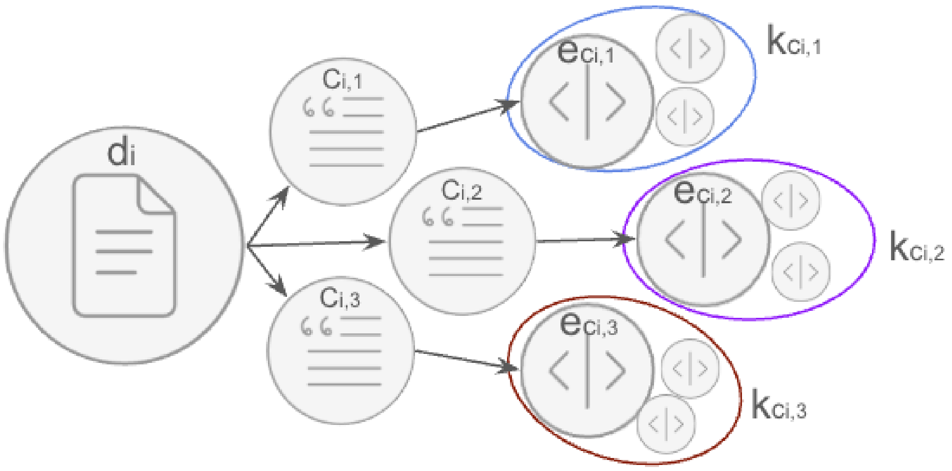

*Figure3: Document Chunking and Chunk Clustering Process*

### 3.3 Retrieval Process

The retrieval phase follows a coarse-to-fine processing:

1. Query Embedding and Reduction: Obtain query embedding ($e_q$); if dimensionality reduction is employed, computing $R(e_q)$.
2. Document Cluster Retrieval: Search $VS_{doc\_cc}$ using query embedding to find top-k relevant document cluster identifiers $\{k_{D,rel}\}$.
3. Document Summary Retrieval: Search within the vector stores of $VS_{doc}(k_{D,rel})$ using query embedding to find top-k document summaries and their corresponding document IDs $\{d_{rel}\}$
4. Chunk Cluster Retrieval: For each $d_{rel}$:
   a. Apply document-specific reduction $R(\cdot)$ to $e_q$ if applicable.
   b. Search corresponding $VS_{chunk\_cc}(d_{rel})$ using query embedding to find top-k chunk cluster identifiers $\{k_{C_{rel}}\}$ within that document.
5. Chunk Retrieval: Search within the relevant stores $VS_{chunk}(k_{C_{rel}})$ using query embedding to find top-k chunks across these clusters.
6. Final Aggregation: Aggregate and rank all retrieved chunks based on similarity to $e_q$ to select the final top-k context.

## 4. Experiment

### 4.1 Dataset

We evaluated our proposed hierarchical RAG framework using four publicly available datasets, each chosen to reflect different retrieval scenarios and reasoning demands. The datasets span both intra-document (searching within a single document) and inter-document (searching across multiple documents) settings, allowing for a comprehensive assessment of retrieval and generation capabilities.

### 4.1.1 Intra-Document Search

For the intra-document retrieval experiment, we mainly used Qasper [23], a long-form question answering dataset built on NLP research papers. It contains 5,049 QA pairs linked to 1,585 documents. This dataset requires retrieving and synthesizing information from various sections of a single document, emphasizing the need for semantic chunking and hierarchical indexing. It is ideal for evaluating intra-document search performance, particularly the chunk-level and cluster-level reasoning abilities of the framework.

For the alternative experiment, we tested NarrativeQA [24], consisting of story-based comprehension tasks, with abstractive questions and answers constructed from summaries of books and movie scripts. This dataset requires deeper narrative understanding and multi-sentence reasoning across extended storylines. It is used to test the system's ability to retain semantic flow and capture global coherence within a single long document.

### 4.1.2 Inter-Document Search

For the inter-document search, we primarily used FRAMES [25], which is a recently released benchmark containing 824 complex, human-annotated multi-hop questions derived from 2–15 Wikipedia articles. It evaluates retrieval accuracy, factual grounding, and multi-constraint reasoning in a unified setting. FRAMES was particularly used to test the robustness of our system in high-complexity retrieval scenarios and to analyze trade-offs between lexical recall and semantic depth.

The other dataset that we used for experimental testing is MultiHop-RAG [26], a benchmark specifically designed for retrieval-augmented generation systems involving multi-hop questions. It evaluates the ability to retrieve relevant content from different documents and synthesize an answer. The dataset highlights the importance of modeling semantic relationships across documents and was used to assess the framework's effectiveness in cross-document reasoning and coarse-to-fine retrieval.

## 4.2 Metrics

To evaluate the effectiveness of our proposed four-layer framework, we adopted several metrics. These were designed to assess both the retrieval accuracy and the quality of generated responses across multiple dimensions—semantic relevance, factuality, and lexical similarity. We organized the evaluation into two main categories, retrieval and generation.

1. **ROUGE Scores** (ROUGE-1, ROUGE-2, ROUGE-L) [27]: ROUGE is a standard for summarization and QA tasks, capturing fluency and content preservation. These scores provide an estimate of how much content from the ground truth is reproduced in the generation within unigram overlap (ROUGE-1), bigram overlap (ROUGE-2), and longest common subsequence (ROUGE-L).
2. **Semantic Similarity with Ground Truth**: Using the Sentence-BERT [11], the similarity scores are computed by passing the ground truth and the generated answer. This metric assesses content similarity even when lexical overlap is low—important for abstractive or paraphrased answers.
3. **LLM-based Self-Check**: This metric is inspired from the LLM-as-a-Judge [28]. An LLM is used to evaluate the output for factual accuracy by comparing between ground truth and generated answer based on the retrieved contexts. ROUGE scores do not account for

semantic correctness or factual hallucinations, also Semantic Similarity does not work if the ground truth answer is True/False or significantly short. This subjective metric by the LLM is applied to both the retrieval and generation performance.

### 4.3 Pretrained Model

Our RAG experiment utilizes several pre-trained models at several stages, including embedding generation, document summarization, and answer generation.

For generating vector embeddings from textual information, such as document chunks, summaries, and user queries, we employed the multi-qa-mpnet-base-cos-v1 model, which is a BERT-based encoder [11]. This model is specifically optimized for semantic similarity tasks, facilitating the effective retrieval based on vector proximity.

During the indexing phase, specifically for generating abstract document summaries, we utilized a smaller language model, llama3.2 (1B) [29, 30]. Its compact parameter size makes it suitable for faster preprocessing tasks, especially summarization.

For the core answer generation task given by the retrieved contexts and the original query as input, our default model is llama3.1 (8B) [29, 31]. This model is a primary generator during our experiment phases. To compare the impact of model size and architecture, we also used two additional LLMs for the generation steps:

- deepseek-r1 (8B) [32]: Used to evaluate the downstream task in RAG with a reasoning model architecture with the same parameter size of our default.
- phi4 (14B) [33]: Employed to compare performance against llama3.1 (8B) and assess the impact of the model size on the generation performance on the RAG system.

At the self-check phase, Gemini 2.0 Flash [34] was used as evaluator LLM to assess the output for factual accuracy by comparing the generated answer against the ground truth.

## 5. Results

We evaluated our proposed four-layer framework using both K-means and Gaussian Mixture Models (GMMs) for clustering and compared a Baseline model on two distinct operations, Intra-Document Search and Inter-Document Search. The performance was measured using ROUGE scores (Rouge-1, Rouge-2, Rouge-L), Cosine Similarity using sentence transformer model, and Self Check Metris for both Generation and Retrieval@5 (the top 5 retrieved contexts).

### 5.1 Intra-Document Search

Figure 4 shows the performance of Intra-Document Search between Baseline and our four-layer proposed frameworks:

- **ROUGE Scores**: Proposed-GMM demonstrated slightly higher performance compared to Proposed-Kmeans. Focusing on ROUGE-2 score, Proposed-GMM achieved the highest result, showcasing better bigram overlap between the generated answer and the ground truth. Although Proposed-GMM is well-performed in ROUGE-1 and ROUGE-L compared to Proposed-Kmean, Baseline stands on the higher performance.
- **Cosine Similarity**: Proposed-GMM achieved the highest cosine similarity score, slightly outperforming Baseline. This result suggests better semantic relevance of its generated output compared to Baseline and Proposed-Kmeans.

- **Self Check Metrics**: The Baseline model outperformed both proposed frameworks. While Proposed-Kmeans performed better than Proposed-GMM, both had lower scores than the Baseline.

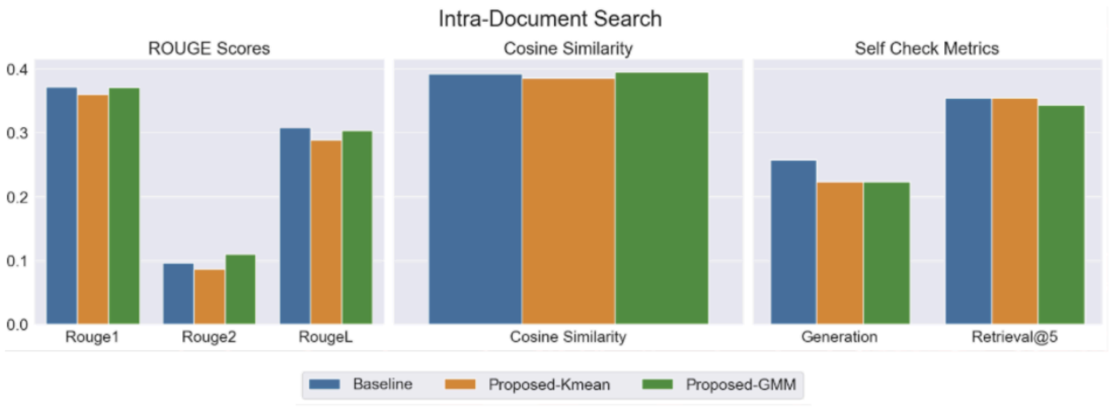

*Figure 4: Performance of Intra-Document Search among Baseline and Proposed Four-Layer Frameworks*

## 5.2 Inter-Document Search

Figure 5 shows the performance of Intra-Document Search between Baseline and our four-layer proposed frameworks:

- **ROUGE Score**: the Baseline model consistently achieved the highest scores across all ROUGE metrics. Both Proposed-Kmeans and -GMM underperformed the Balinese. The lower ROUGE-L scores for the proposed methods indicate a slightly reduced alignment with the ground truth compared to the Baseline.
- **Cosine Similarity**: Proposed-GMM showed the higher score compared to Proposed-Kmeans, but both proposed frameworks edge out the Baseline. Although all experiments indicated comparable semantic performance, the Baseline model outperformed both proposed frameworks.
- **Self Check Metrics**: A key finding is the performance on the Generation metric, where both Proposed-Kmeans and -GMM demonstrated superior performance over the Baseline. Proposed-GMM achieved the highest accuracy overall, under a larger LLM evaluation. However, on the retrieval performance at the top 5 contexts (Retrieval@5), the Baseline model maintained the highest result, with both proposed four-layer frameworks.

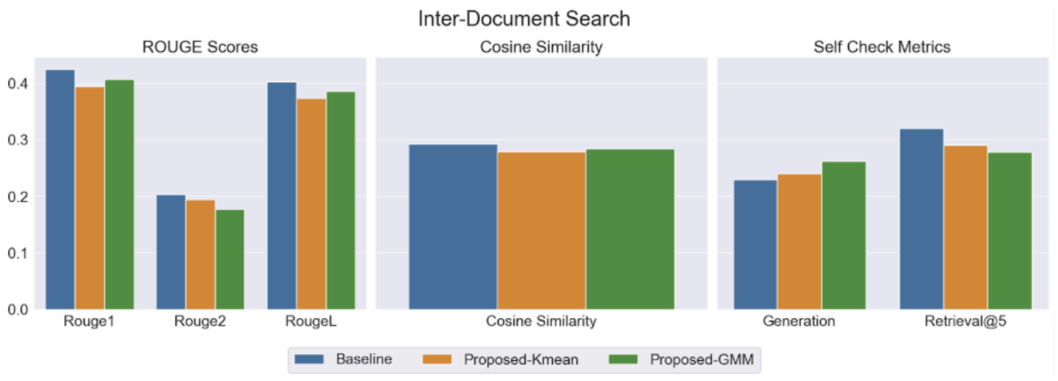

*Figure 5: Performance of Inter-Document Search among Baseline and Proposed Four-Layer Frameworks*

## 6. Discussion

Our proposed framework implements a hierarchical structure utilizing clustering at multiple granularity levels from document clusters to original text chunks. We evaluated both its generation and retrieval performance using K-Means and Gaussian Mixture Models (GMMs) for clustering against a baseline flat retrieval system primarily on intra-document (Qasper) and inter-document (Frames) search tasks. As we presented, the results present a nuanced performance of the framework's capabilities and trade-offs.

### 6.1 Interpretation of Findings

For the intra-document search performance using Qasper dataset, the proposed framework showed mixed results. Proposed-GMM achieved the highest ROUGE-2 score, while maintaining the semantic-based similarity. This behavior suggests capturing bigram overlap and semantic relevance effectively. Since the GMM offers a softer clustering, allowing each vector to be assigned to multiple clusters, it could facilitate more nuanced grouping of chunk embeddings compared to K-means hard assignments. However, both proposed methods on ROUGE-1 and ROUGE-L, and the accuracy provided from LLM-based self-checking underperformed the baseline framework. While the hierarchical approach aims to prune an irrelevant bunch of passages (chunk clusters, in our case) at the first retrieval step, the chunks containing essential words and phrases for high lexical overlap might be lost during dimensionality reduction or inefficient centroid representation.

Regarding the inter-document search result, the proposed framework revealed both strength and weakness. The baseline consistently achieved higher ROUGE scores, indicating superior lexical recall and ability to reconstruct longer answer sequences from the ground truth. However, focusing on self-checking metrics for the LLM generation with the retrieved contexts, both proposed frameworks demonstrated higher accuracy compared to Baseline. This suggests that a hierarchical structure can effectively prune irrelevant information and enhance consistency within the retrieved contexts under multi-document search; even if contexts are semantically relevant, they might provide opposites or inconsistent information due to being retrieved from a significant amount of documents. Conversely, the baseline excelled in Retrieval@5, implying its retrieved context was deemed more sufficient by the evaluator LLM. This contradictory result might indicate that the baseline's brute-force search approach, while retrieving less focused context overall, is more likely to include at least some sentences directly supporting the answer within the top-5 contexts, which satisfies the sufficient check. In contrast, our proposed method might sometimes miss a specific supporting fact due to pruning, even if the overall retrieved context leads to an accurate generation via general inference.

### 6.2 Comparison with Related Works

Although we did not compare the generation and retrieval performance numerically due to the difficulty of implementing a precise architecture without source codes, our framework incorporated various principles.

Our hierarchical structure, LLM summarization, and clustering implementation are based on RAPTOR [1] and JTR [20]. We apply LLM summarization for each document at once, since RAPTOR's repetitive use of LLM generation increases computational resources. However, replacing extensive use of summarization with cluster centroids (referring from FAISS IVF method [5]) might be a root cause of less informative proxies.

Regarding the coarse-to-fine retrieval approach, exemplified by FunnelRAG [9], we aimed to implement this paradigm by using document summaries and cluster centroids as the coarse semantic layer, guiding the search towards the finer-grained layer composed of the original text chunks. While we employed the hierarchical structure and the step-by-step approach from coarse to finer layers, we could incorporate the different levels of granularity at each step of retrieval, exemplified by MixGR [18] and MoG [19].

### 6.3 Limitation and Future Works

Throughout the experimentation phases, we identified several limitations in our proposed approaches:

- **Evaluation Scope**: We relied on ROUGE, cosine similarity, and LLM-based self-consistency checks. However, since the ground truth answer has various formats, including Yes/No, a short answer, and a passage, we should have applied appropriate metrics for each type of answer or create an aggregated evaluation metric.
- **Centroid Representation**: We used PCA for dimensionality reduction model, but this might not adequately capture the semantic richness of the original contexts and lead to retrieval errors. On the other hand, the clustering for higher dimensional vectors are likely to cause local minima errors. So, we could utilize the nearest-neighbor approaches that the Facebook Research team explored in FAISS Library [5].
- **Insufficient Query**: We conducted a similarity calculation through L2 distance between two vectors; one is the original query, and the other is each indexed vector in the database. The original query is often insufficient, so we could integrate the generated document summary and implement the query expansions. Furthermore, we could employ an agent system; for instance, letting the LLM rephrase or regenerate the original query, which generally have richer contexts.
- **Dense Retrieval**: Our proposed frameworks implemented dense retrieval only, relying on dense vector embeddings to capture semantically relevant vectors. However, the integration of keyword-based sparse retrieval, (TF-IDF, BM25, etc.) could enhance the retrieval performance, especially within a single document search. Thus, our proposed frameworks can achieve further improvements through the use of the aggregated similarity scores.

## 7. Conclusion

Our proposed four-layer hierarchical RAG framework demonstrates potential in enhancing generated answer correctness for complex queries by leveraging semantic structure, despite trade-offs in lexical recall. Further refinement and evaluation are necessary to fully understand its capabilities and optimize its performance across diverse RAG applications. For future work, we could expand the capability of dense and sparse retrieval to the information organization system beyond the emerging searching and retrieval usage.

## References

[1] Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C. D. (2024). Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059v1.*

[2] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys, 55*(12), Article 3571730. https://doi.org/10.1145/3571730

[3] Li, S., Stenzel, L., Eickhoff, C., & Bahrainian, S. A. (2025). Enhancing Retrieval-Augmented Generation: A Study of Best Pra. *arXiv preprint arXiv:2501.07391*.

[4] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.T., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems, 33*, 9459-9474.

[5] Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P. E., ... & Jégou, H. (2024). The faiss library. *arXiv preprint arXiv:2401.08281*.

[6] Malkov, Y. A. and Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

[7] Chen, T., Wang, H., Chen, S., Yu, W., Ma, K., Zhao, X., Zhang, H. and Yu, D. (2024). Dense x retrieval: What retrieval granularity should we use?. *arXiv preprint arXiv:2312.06648v3*.

[8] An, K., Yang, F., Li, L., Lu, J., Cheng, S., Si, S., Wang, L., Zhao, P., Cao, L., Lin, Q., Rajmohan, S., Zhang, D., Zhang, Q., & Chang, B. (2024). Thread: A Logic-Based Data Organization Paradigm for How-To Question Answering with Retrieval Augmented Generation. *arXiv preprint arXiv:2406.13372.*

[9] Zhao, X., Zhong, Y., Sun, Z., Hu, X., Liu, Z., Li, D., Hu, B. and Zhang, M. (2024). FunnelRAG: A Coarse-to-Fine Progressive Retrieval Paradigm for RAG. *arXiv preprint arXiv:2410.10293v1.*

[10] Wang, X., Xiang, Y., Gui, L., & He, Y. (2024). GARLIC: LLM-Guided Dynamic Progress Control with Hierarchical Weighted Graph for Long Document QA. *arXiv preprint arXiv:2410.04790.*

[11] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

[12] Diaz-Rodriguez, J. (2025). k-LLMmeans: Summaries as Centroids for Interpretable and Scalable LLM-Based Text Clustering. *arXiv preprint arXiv:2502.09667.*

[13] Elliott, O. P., & Clark, J. (2024, August). The Impacts of Data, Ordering, and Intrinsic Dimensionality on Recall in Hierarchical Navigable Small Worlds. In *Proceedings of the 2024 ACM SIGIR International Conference on Theory of Information Retrieval* (pp. 25-33).

[14] Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M. (2020, November). Retrieval augmented language model pre-training. In *International conference on machine learning* (pp. 3929-3938). PMLR.

[15] Qu, R., Tu, R., & Bao, F. (2024). Is Semantic Chunking Worth the Computational Cost?. *arXiv preprint arXiv:2410.13070v1*.

[16] Duarte, A. V., Marques, J., Graça, M., Freire, M., Li, L., & Oliveira, A. L. (2024). Lumberchunker: Long-form narrative document segmentation. *arXiv preprint arXiv:2406.17526v1*.

[17] Zhao, J., Ji, Z., Qi, P., Niu, S., Tang, B., Xiong, F., & Li, Z. (2024b). Meta-Chunking: Learning Efficient Text Segmentation via Logical Perception. *arXiv preprint arXiv:2410.12788v2*.

[18] Cai, F., Zhao, X., Chen, T., Chen, S., Zhang, H., Gurevych, I., & Koeppl, H. (2024, November). MixGR: Enhancing Retriever Generalization for Scientific Domain through Complementary Granularity. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* (pp. 10369-10391).

[19] Zhong, Z., Liu, H., Cui, X., Zhang, X., & Qin, Z. (2024). Mix-of-Granularity: Optimize the Chunking Granularity for Retrieval-Augmented Generation. *arXiv preprint arXiv:2406.00456v1*.

[20] Li, H., Ai, Q., Zhan, J., Mao, J., Liu, Y., Liu, Z., & Cao, Z. (2023). Constructing Tree-based Index for Efficient and Effective Dense Retrieval. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*, 131-140. https://doi.org/10.1145/3539618.3591651

[21] Zhang, N., Choubey, P. K., Fabbri, A., Bernadett-Shapiro, G., Zhang, R., Mitra, P., Xiong, C., & Wu, C. S. (2024). SiReRAG: Indexing Similar and Related Information for Multihop Reasoning. *arXiv preprint arXiv:2412.06206v1*.

[22] Hosking, T., Tang, H., & Lapata, M. (2024). Hierarchical indexing for retrieval-augmented opinion summarization. *Transactions of the Association for Computational Linguistics*, *12*, 1533-1555.

[23] Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., & Gardner, M. (2021). A dataset of information-seeking questions and answers anchored in research papers. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, & Y. Zhou (Eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4599–4610). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.365

[24] Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., & Grefenstette, E. (2018). The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics, 6*, 317-328.

[25] Krishna, S., Krishna, K., Mohananey, A., Schwarcz, S., Stambler, A., Upadhyay, S., & Faruqui, M. (2024). Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. *arXiv preprint arXiv:2409.12941.*

[26] Tang, Y., & Yang, Y. (2024). Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391.*

[27] Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[28] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging LLM-as-a-judge with MT-bench and Chatbot Arena. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS '23) (Article 2020, pp. 46595–46623). Curran Associates Inc.

[29] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... & Vasic, P. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783.*

[30] Meta. (2024a, October 28). Llama 3.2: Powering new AI experiences on device and the edge. Meta AI Blog. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

[31] Meta. (2024b, July 23). Introducing Llama3.1; Our most capable models to date. Meta AI Blog. https://ai.meta.com/blog/meta-llama-3-1/

[32] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., ... & He, Y. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948.*

[33] Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., ... & Zhang, Y. (2024). Phi-4 technical report. *arXiv preprint arXiv:2412.08905.*

[34] Pichai, S., Hassabis, D., & Kavukcuoglu, K. (2024, December 11). Introducing Gemini 2.0: our new AI model for the agentic era. The Keyword. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/