The Chunking Paradigm: Recursive Semantic for RAG Optimization

Seemab Latif [*], **Huma Ameer**[†], **Muhammad Hannan Akram**[†] **and Mehwish Fatima**[†]
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST), Islamabad, Pakistan
{seemab.latif, hameer.msds20seecs,makram.bsds23seecs,
mehwish.fatima}@seecs.edu.pk

Abstract

Retrieval Augmented Generation (RAG) has risen to prominence for boosting the capabilities of Large Language Models (LLMs) through the integration of external knowledge. Notably, the document chunking process plays a central role in the performance of RAG pipelines. Nevertheless, incoherent document splits and inappropriate chunk sizes hinder retrieval efficiency and contextual accuracy. To address this, we propose Recursive Semantic Chunking (RSC), a dynamic and adaptive chunking framework that ensures semantic coherence. It maintains coherence by recursively splitting large chunks and merging smaller ones. Unlike conventional methods, RSC preserves contextual integrity while optimizing retrieval efficiency. The evaluation across 4 distinct datasets outperformed traditional semantic chunking techniques on evaluation metrics; contextual relevancy, contextual precision, contextual recall, retrieval time, faithfulness and answer relevancy. Results demonstrate that RSC consistently outperforms traditional chunking techniques, achieving higher contextual relevancy and total score while maintaining efficient retrieval times. These findings highlight the potential to optimize RAG systems and to improve the document chunking steps in the systems.

1 Introduction

Large Language Models (LLMs) are widely adopted across various domains in the form of chatbots, AI assistants, and other applications (Siddharth and Luo, 2024; Sahlman et al., 2023). The performance of LLMs is enhanced via the integration of external knowledge sources, specifically for custom applications. In addition, we can leverage the capabilities of LLMs without training them. The aforementioned enhancement can be made via Retreival-Augmented Generation (RAG) (Lewis et al., 2020).

The RAG process begins with a user's query being sent to the LLM, which generates a retrieval request based on that query. This request is forwarded to the retriever system, which searches the vector database. Embeddings of documents chunk i.e. context is stored in vector database. The relevant context is then retrieved and combined with the user's query before being sent to the LLM for a final response, as shown in Figure 1. Researchers have developed various RAG-based solutions across different domains, such as finance and healthcare (Alkhalaf et al., 2024; He et al., 2024; Feng et al., 2024; Mathur et al., 2024).

The critical aspect of the RAG pipeline is the chunking of documents. Chunking in RAG systems is a technique that breaks down large documents into smaller, manageable segments known as "chunks" (LangChain, 2024). This process is crucial as it enhances the efficiency and accuracy of information retrieval, which leads to better outcomes for the system. The nature of context retrieved from the vector database is based on the segmentation of these documents, therefore, the choice of chunking techniques is a significant step in the pipeline (Setty et al., 2024). The chunking techniques directly affect the quality of retrieved-context and retrieval time. It eventually affects the quality of the product that is utilizing RAG-based applications. The choice of chunking is quite challenging i.e. larger chunks can lead to slower retrieval, or retrieve irrelevant chunks and small chunks may not adhere to a coherent information unit. Recently, there has been a shift in research focus towards optimal chunking techniques i.e. (Yepes et al., 2024). Although frameworks such as LangChain (AI, 2024) and LamaIndex (Liu, 2022) have various chunking strategies. Due to complexities of the document structure, and cus-

[*]Corresponding author.
[†]Equal contribution.

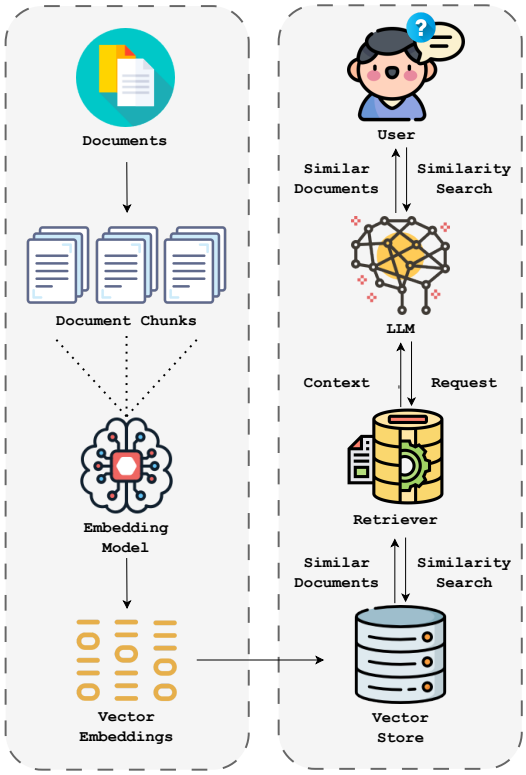


Figure 1: Information Flow in Retrieval Augmented Generation (RAG)

tom systems, it is still a challenging task at hand.

In this paper, we propose Recursive Semantic Chunking that focuses on optimizing the semantic chunking of the documents. The following are the key contributions of this work:

- We propose a Recursive Semantic Chunking method, designed to split textual documents into coherent semantic chunks of an appropriate size.
- We introduce a dynamic method for adjusting the chunk size. The recursive nature of this method ensures that larger text segments are progressively broken down while maintaining semantic integrity. Smaller segments are merged strategically, keeping the chunk size balanced.
- We demonstrate that the proposed method improves retrieval time compared to traditional chunking techniques.
- As part of this work, we introduce NewsMatrix-71, a large-scale, multi-domain news dataset.

2 Related Work

Retrieval Augmented Generation systems rely on the context returned from the retrieval algorithms, making chunking a key factor in the pipeline (Yepes et al., 2024). Therefore, the choice of chunking strategies is a critical step. Ineffective techniques can result in either incomplete chunks leading to losing context or large chunks with irrelevant information negatively impacting the accuracy of the retrieval (Setty et al., 2024).

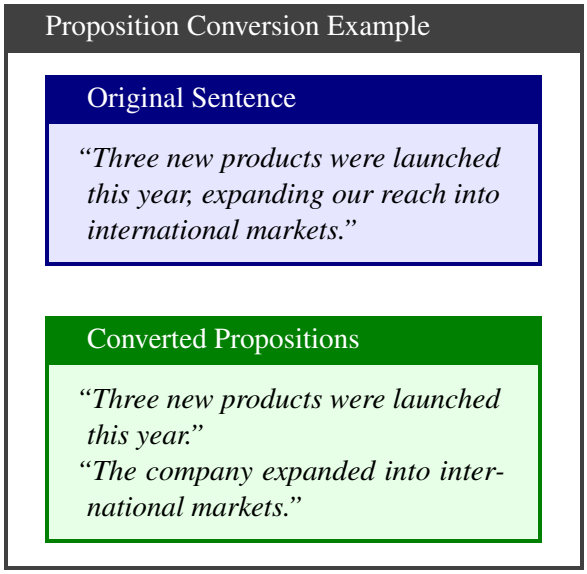
One of the common approaches is to split the document based on fixed numbers of chunks. However, it has a potential loss of context in both cases larger or smaller chunk size (Teja, 2023). To address this, the researchers introduce recursive split by character technique (LangChain, 2023). It recursively splits keeping the longest text chunks together with a need to define and constant adjustment of chunk size overlapping making it computationally expensive.

Although the recursive text split tends to keep the chunks semantically closed together, it does not directly account for semantic meaning. Conversely, semantic chunking (LangChain, 2024) groups the text that is semantically similar together. It first splits the text into sentences and groups them into three sentences, then merges similar groups in the embedding space. However, this technique does not ensure optimal chunk sizes. Since its mechanism is dependent on the similarity of the embedding vectors, it may lead to larger chunks and cause hallucinations.

Agentic chunking (FullStackRetrieval, 2024) pushed this idea further by leveraging Large Language Models. It converts text into propositions via LLMs (Chen et al., 2024). Propositions are defined as standalone statements that convey a single fact clearly without needing extra context. It can be referred to as the smallest unit of meaning within a text, each expressing one distinct idea. Propositions retain the semantic meaning in individual statements as shown in the following example:

Once the propositions are created, these are passed to an LLM, which is then prompted to group these chunks. This approach offers flexibility and higher accuracy. Nevertheless, it requires well-crafted prompts and dependency on the capability of the acquired LLM.

Working on efficient chunking techniques is an open research area as not much has been explored in this regard.



3 Dataset

We introduce a new large-scale news dataset, named NewsMatrix-71[1]. It covers a diverse set of news categories over multiple years.

3.1 Scraped News Dataset (NewsMatrix-71)

We compile this dataset by scraping English news articles from Dawn[2], Tribune[3], and Daily Times[4]. This dataset covers the span of three years (2021–2023) and has up to 96,859 news articles categorized into 71 unique topics, including Business, Fashion, Health, World, and more. It offers a diverse, time-spanning, and category-rich corpus suitable for various NLP tasks. It captures a broad spectrum of global and regional news, making it a valuable resource for research. Given the size and scope of this dataset, we will selectively release a publicly available subset to facilitate reproducibility and further research.

4 Recursive Semantic Chunking

This section presents the Recursive Semantic Chunking framework in detail. The primary objective is to ensure the splitting of chunks is semantically coherent and maintains the integrity of the content. In addition, the size of the chunks should be optimal. The standard semantic chunking technique tends to generate large chunks, which negatively impact the performance of retrieval-augmented generation systems. Furthermore, in custom RAG projects, documents often belong to specific topics, and larger chunks reduce system efficiency.

Algorithm 1 provides a detailed outline of the proposed chunking process. All predefined values are determined after extensive experimentation. The following steps describe the pipeline.

Segmentation of Textual Data from Files

The data store consists of files f_i containing textual data stored as strings T_i. Since LLMs have token limits, each T_i undergoes a length check. If it exceeds the threshold T_{max}, the file is split into smaller segments $\{t_1, t_2, \ldots, t_n\}$, ensuring that $|t_j| \leq T_{\text{max}}$. The splitting occurs at the nearest sentence boundary (e.g., full stop, question mark) to preserve linguistic coherence.

Initial Semantic Chunking

Each segment t_j undergoes an initial semantic chunking process (LangChain, 2024). In this step, the semantically similar texts are grouped in the embedding space, forming $C_0 = \{c_1, c_2, \ldots, c_m\}$, where c_k represents an initial chunk.

Recursive Semantic Chunking

For each chunk $c_k \in C_0$, the semantic chunker is recursively applied if its length exceeds the threshold T_{chunk} (1,500 characters). With each recursive iteration, the breakpoint threshold parameter is gradually reduced, ensuring that large chunks are broken into smaller, semantically meaningful segments. The recursive function $R(c, T)$ operates as follows:

$$R(c,T) = \begin{cases} c & \text{if } |c| \leq T \\ R(\text{split}(c, T-\delta), T-\delta) & \text{if } |c| > T \end{cases}$$

where δ represents a small reduction factor to progressively decrease chunk size in each iteration. The reduction factor δ is heuristically set to 3 after initial experimentation. Although not tuned through systematic search, this value is selected to ensure a gradual and controlled recursive breakdown of large chunks. This value is kept fixed across all datasets to maintain consistency and reproducibility.

[1]This data will be published publicly and free for research purposes after the paper's acceptance. It will be shared under the **Creative Commons Attribution 4.0 International License (CC BY 4.0)**

[2]Dawn

[3]Tribune

[4]Daily Times

Merging Short Chunks

Following recursive chunking, some chunks may become too short (i.e., less than T_{merge}, set to 350 characters). Extremely small chunks may lack semantic coherence, leading to information loss. To address this, the similarity score of smaller chunks is calculated with both preceding and subsequent chunks. It is merged with the chunk that has the highest similarity score. This ensures semantic integrity while preventing the loss of meaningful text. The merging process is defined as follows:

$$\text{For } i = 1 \text{ to } n: \begin{cases} \text{If } |c_i| < T_{\text{merge}}, \text{ compute:} \\ \quad S_{\text{prev}} = \text{sim}(c_i, c_{i-1}) \\ \quad S_{\text{next}} = \text{sim}(c_i, c_{i+1}) \\ \text{Merge with highest similarity chunk} \\ \quad \text{If } S_{\text{prev}} \geq S_{\text{next}}, \text{ then:} \\ \quad\quad c_{i-1} \leftarrow c_{i-1} + c_i \\ \quad \text{Else:} \\ \quad\quad c_{i+1} \leftarrow c_i + c_{i+1} \end{cases}$$

Here, S_{prev} and S_{next} represent the similarity scores between the small chunk c_i and its neighboring chunks c_{i-1} and c_{i+1}, respectively. The chunk c_i is merged with the chunk that has the highest similarity score, ensuring that the resulting merged chunk maintains semantic coherence.

Uniform Chunk Size Adjustment

Finally, the algorithm checks whether any chunk exceeds the threshold T_{final} (2,500 characters). If a chunk surpasses this limit, it undergoes a recursive character-based text split (LangChain, 2023). The final adjustment process is defined as:

$$\text{For } i = 1 \text{ to } m: \begin{cases} \text{If } |c_i| > T_{\text{final}}: \\ \quad \text{Apply Recursive Split Function:} \\ \quad c_i \leftarrow \text{RecursiveSplit}(c_i, T_{\text{final}}) \end{cases}$$

This step ensures that the final chunk set, $C_{\text{final}} = \{c_1, c_2, \ldots, c_m\}$, meets size constraints while maintaining semantic coherence. The processed chunks are then stored in vector databases for RAG tasks.

Distinction from Baseline Chunkers

While our method incorporates components from existing LangChain utilities, i.e. semantic chunking for initial grouping and character-based recursive splitting for final chunk size enforcement. These steps function as structural helpers rather than the core of our approach. The key innovation of RSC lies in its intermediate refinement logic: recursive breakdown with dynamic thresholds, similarity-based merging of smaller chunks, and controlled preservation of semantic coherence. These operations are not present in the baseline LangChain chunkers and are designed to address the limitations of fixed-size or purely embedding-based segmentation. Therefore, while we leverage LangChain for low-level chunk initialization and splitting, the significant performance improvements observed in contextual and answer-level metrics stem from our recursive and adaptive chunking strategy.

Algorithm 1: Recursive Semantic Chunking

Input: Dataset $D = \{f_1, f_2, \ldots, f_N\}$;
Maximum chunk size $T_{\text{max}} = 15{,}000$;
Recursive chunking threshold $T_{\text{chunk}} = 1{,}500$;
Final chunk size threshold $T_{\text{final}} = 2{,}500$;
Minimum chunk size for merging $T_{\text{merge}} = 350$
Output: Final set of chunks C_{final}

Initialization:
$C_{\text{final}} \leftarrow \emptyset$
Initial Semantic Chunking:
Apply chunking to each segment t_j:
$C_0 \leftarrow \{c_1, c_2, \ldots, c_m\}$
foreach *chunk* $c_k \in C_0$ **do**
  if $|c_k| > T_{chunk}$ **then**
    Recursive Semantic Chunking:
    $R(c_k, T_{\text{chunk}}) = R(\text{split}(c_k, T_{\text{chunk}} - \delta), T_{\text{chunk}} - \delta)$
    $c_k \leftarrow R(c_k, T_{\text{chunk}})$
foreach *chunk* $c_k \in C_0$ **do**
  if $|c_k| \leq T_{merge}$ **then**
    Compute similarity with previous chunk:
    $S_{\text{prev}} \leftarrow \text{similarity}(c_{k-1}, c_k)$
    Compute similarity with next chunk:
    $S_{\text{next}} \leftarrow \text{similarity}(c_k, c_{k+1})$
    if $S_{prev} \geq S_{next}$ **then**
      Merge with previous chunk:
      $c_{k-1} \leftarrow c_{k-1} + c_k$
    else
      Merge with next chunk:
      $c_{k+1} \leftarrow c_k + c_{k+1}$
Add merged chunks to C_{final}
foreach *chunk* $c_k \in C_{final}$ **do**
  if $|c_k| > T_{final}$ **then**
    Split chunk:
    $c_k \leftarrow \text{split}(c_k, T_{\text{final}})$
Return: Final set of chunks C_{final}

5 Experimental Design

Our evaluation framework is designed to rigorously assess the impact of our proposed technique: Recursive Semantic Chunking (RSC). Incorporating RSC in the RAG pipeline for question-answering tasks, we demonstrate its capabilities in preserving contextual coherence and improving retrieval precision. This section details our evaluation methodology, covering dataset selection, synthetic question

Table 1: Summary of Datasets used for Evaluating the Proposed Chunking Technique, including Open-source Corpora and the Custom Dataset NewsMatrix-71.

Dataset	Words	Characters	Paragraphs	Source
BBC	854,490	5,039,982	2,225	BBC Dataset
SQuAD	152,394	966,345	1,204	SQuAD
QuaC	440,971	2,664,801	1,000	QuaC
NewsMatrix-71	677,258	4,227,679	1,500	Dawn, Tribune Daily Times

generation, chunking techniques, implementation setup, and performance metrics

5.1 Datasets

We evaluate our proposed chunking technique using four datasets, including three open-source corpora—BBC (Greene and Cunningham, 2006), SQuAD (Rajpurkar et al., 2016), and QuaC (Choi et al., 2018)—along with a custom-scraped news dataset, NewsMatrix-71. The NewsMatrix-71 dataset, created by scraping English news articles, is stored in .txt format. For experimentation, we use a 1,500-article subset containing 677,258 words and 4,227,679 characters. A summary of all datasets is provided in Table 1.

5.2 Synthetic Question Generation

These evaluations of the chunking techniques are based on the response from the question-answering system. Therefore, we utilized LLM to create synthetic questions from each dataset. For each dataset, we randomly generate 50 synthetic questions per dataset to balance computational feasibility with evaluation diversity. This quantity is consistent with recent study Merola and Singh, 2025. This quantity is consistent with recent study Merola and Singh, 2025. To generate synthetic questions, we randomly selected passages from each dataset. To ensure reasonable topic coverage, we manually examined multiple random subsets and selected one for question generation. While this approach does not guarantee perfect topic stratification, it provides a practical balance between topic diversity and simplicity in sampling. We employ Gemini Flash 1.5 to generate corresponding questions. The ChatPromptTemplate module from LangChain is used to structure the input prompt, guiding the model to generate relevant and context-aware questions for each passage. Once generated, the synthetic questions are stored and later used to assess the retrieval and response quality of different chunking techniques. By introducing synthetic queries, we create an additional layer of evaluation that allows us to measure how well-chunked text segments support question-answering tasks beyond the scope of existing datasets.

5.3 Chunking Techniques

To establish a baseline, we implement three widely used chunking techniques. Recursive Character Text Splitter segments (LangChain, 2023), and Semantic Chunking (LangChain, 2024). Next, we employ our proposed technique; Recursive Semantic Chunking framework for comparison.

5.4 Implementation Details

For downstream question-answering tasks, we store the chunks in the RAG pipeline using LangChain[5]. All the techniques use "*all-MiniLM-L6-v2*" [6]embedding. The resulting chunks are stored in the Facebook AI Similarity Search (FAISS) vector database (Douze et al., 2024). The "*ChatPromptTemplate module*" is used with "*Gemini Flash 1.5*" [7], a state-of-the-art Large Language Model optimized for contextual reasoning.

5.5 Evaluation metrics

We assess chunking techniques by integrating them into the RAG pipeline for a question-answering task. For evaluation, we use DeepEval by Confident AI [8], an open-source framework designed for LLM evaluation. DeepEval leverages LLMs and other NLP models to measure performance. In our study, GPT-3.5-turbo generates answers, with evaluation metrics focusing on contextual accuracy and relevance in both retrieval and generation stages. The following formulas are taken from DeepEval for evaluation. Additionally, we compare retrieval time across different strategies.

Contextual Precision

It measures how well relevant nodes are ranked higher in the retrieval context.

$$\text{CP} = \frac{1}{\text{Rel. Nodes}} \sum_{k=1}^{n} \left(\frac{\text{Rel. Nodes to } k}{k} \times r_k \right)$$

where r_k is 1 for relevant nodes, 0 otherwise.

[5]LangChain
[6]Sentence Embedding: all-MiniLM-L6-v2
[7]Gemini Flash 1.5
[8]https://www.confident-ai.com

Contextual Recall

The metric evaluates the ability of the system to capture relevant information:

$$\text{CR} = \frac{\text{Attributable Statements}}{\text{Total Statements}}$$

Contextual Relevancy

It measures the overall relevance of the retrieval context with respect to the query:

$$\text{CRel} = \frac{\text{Relevant Statements}}{\text{Total Statements}}$$

Answer Relevancy

Answer Relevancy evaluates the relevance of the generated output:

$$\text{AR} = \frac{\text{Relevant Statements}}{\text{Total Statements}}$$

Faithfulness

Faithfulness measures how factually accurate the output is:

$$\text{Faithfulness} = \frac{\text{Truthful Claims}}{\text{Total Claims}}$$

Retrieval Time

The Retrieval Time RT is defined as the total time taken to retrieve the context and generate the final answer for a query:

$$\text{RT} = t_{\text{end}} - t_{\text{start}}$$

These evaluation metrics allow us to compare the trade-offs between semantic integrity, retrieval effectiveness, and computational efficiency across different chunking approaches.

6 Results and Analysis

6.1 Results

Table 2 shows the chunk counts for different techniques. RSC achieves the best balance between granularity and coherence. In contrast, the Recursive Character Text Splitter generates the highest number of chunks due to its character-based splitting, while Semantic Chunking produces the fewest, resulting in larger segments. This balance reflects an important trade-off in RAG system design. Excessive chunking can inflate the retrieval space, leading to fragmented context. While larger chunks provide broader context, they increase the risk of irrelevant retrieval, hallucinations, and longer retrieval times. RSC finds a middle ground, ensuring semantic integrity while maintaining meaningful chunk sizes. By keeping the chunk count within an optimal range, RSC improves contextual relevancy, as further supported by the downstream performance metrics in Table 3.

Table 2: Number of Chunks Formed by Each Chunking Method Across Datasets.

Dataset	**Recursive Character**	**Semantic**	**RSC (Proposed)**
BBC News	12,674	3,844	8,115
SQuAD	1,258	2,327	2,343
QuAC	2,464	2,307	4,121
NewsMatrix-71	3,793	2,961	5,474

Table 3 presents the comparative performance of chunking techniques on the question-answering task across multiple datasets. The proposed Recursive Semantic Chunking consistently outperforms other techniques, particularly in Contextual Relevancy and Total Score, while maintaining an optimal balance between chunk size and retrieval efficiency.

The performance of chunking techniques across the datasets reveals interesting trends as shown in Figure 2. The best results are observed in SQuAD and NewsMatrix-71. SQuAD, achieving the highest Total Score under RSC, highlights the advantage of semantically coherent segmentation in structured question-answering datasets. NewsMatrix-71 achieves the highest Contextual Relevancy with RSC, demonstrating its effectiveness in handling diverse and large-scale articles.

In contrast, QuAC performs the worst, particularly under Semantic Chunking and Recursive Semantic Chunking. This is likely due to its conversational nature, which demands deeper contextual understanding.

While RSC does not lead in Answer Relevancy across all datasets, it is an important metric for evaluating end-to-end RAG performance. It consistently achieves top performance in Total Score and Contextual Relevancy. It is important to note that Answer Relevancy may be influenced by factors beyond chunking quality, such as the formulation of user queries (Sclar et al., 2024) or reasoning behavior of the language model during generation (Jiang et al., 2025). In contrast, Contextual Relevancy more directly reflects the quality and alignment of retrieved content with the query, making it a

Table 3: Performance Metrics for Different Chunking Techniques Across Datasets. Scores are out of 50, except Total Score (out of 250). Retrieval time is measured in seconds.
Abbreviations: RC = Recursive Character, RSC = Recursive Semantic Chunking,Avg Retv Time = Average Retrieval Time
Bold values indicate the highest performance for each metric.

Metric	RC	Sem	RSC (Proposed)
BBC News			
Answer Relevancy	**45.89**	43.89	41.97
Answer Faithfulness	38.51	35.18	**42.55**
Contextual Recall	43	45.5	**46.33**
Contextual Precision	47.02	44.82	**48.98**
Contextual Relevancy	11.40	8.78	**11.56**
Total Score ↑	185.83	178.17	**191.39**
Avg Retv Time(s) ↓	0.721	0.799	**0.716**
NewsMatrix-71			
Answer Relevancy	**47.92**	47.81	47.08
Answer Faithfulness	**43.96**	43.93	40.11
Contextual Recall	46.33	**46.5**	45.67
Contextual Precision	48.5	47.26	**48.83**
Contextual Relevancy	13.94	14.71	**19.83**
Total Score ↑	200.65	200.21	**201.52**
Avg Retv Time(s) ↓	0.72	**0.71**	**0.71**
SQuAD			
Answer Relevancy	47.28	46.67	**48.59**
Answer Faithfulness	44.98	43.71	**46.5**
Contextual Recall	**50**	49	**50**
Contextual Precision	47.09	47.99	**47.99**
Contextual Relevancy	17.7	20.09	**20.12**
Total Score ↑	207.05	207.46	**213.2**
Avg Retv Time(s) ↓	0.97	0.97	**0.96**
QuAC			
Answer Relevancy	**45.4**	44.69	43.67
Answer Faithfulness	41.675	**44.25**	43.63
Contextual Recall	47.08	45.33	**48.58**
Contextual Precision	43.67	45.16	**45.76**
Contextual Relevancy	**12.47**	9.64	9.38
Total Score ↑	190.29	189.07	**191.01**
Avg Retv Time(s) ↓	**0.62**	0.65	0.64

stronger indicator of chunking effectiveness.

Overall, among the chunking techniques, RSC achieves the highest Total Score across all datasets. The recursive breakdown mechanism in RSC ensures that large chunks do not negatively impact RAG tasks. Additionally, Contextual Relevancy improves significantly with RSC, as evident in datasets like BBC News (11.56) and NewsMatrix-71 (19.83), demonstrating its capability to maintain semantic coherence.

These findings suggest the impact of the type and structure of data on the chunking techniques. However, in comparison, RSC is the most effective among the baseline chunking techniques.

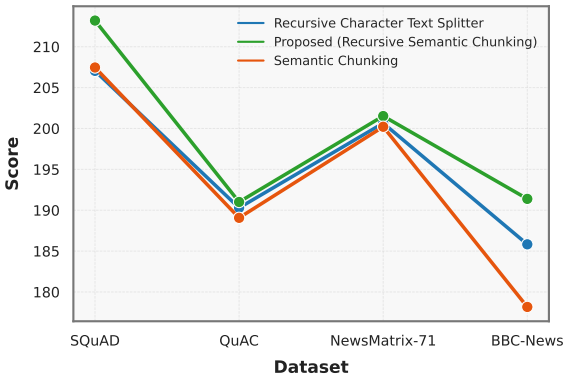


Figure 2: Performance Comparison of Chunking Techniques Across Datasets

6.2 Analysis

To evaluate the impact of Recursive Semantic Chunking on retrieval efficiency and chunk coherence, we conduct performance analysis across multiple datasets. The evaluation uses datasets of varying structures such as structured question-answering datasets (SQuAD, QuAC) and unstructured large-scale datasets (BBC News, NewsMatrix-71). It ensures that our findings are generalizable across multiple RAG tasks.

Study on Propositional Segmentation

We conduct a study to analyze the effect of propositional segmentation incorporated in our proposed chunking technique. The hypothesis is that propositional segmentation enhances Contextual Relevancy.

To validate our hypothesis, we experiment by including propositional segmentation in RSC and compare the results. For this case study, we employ the BBC News dataset. The comparison of results is presented in Table 4

The results confirm that propositional segmentation improves Contextual Relevancy (11.56 to 16.09). However, it is to be that improvement comes at the cost of increased retrieval time (from 0.716s to 0.8183s). In addition, it also has a computational overhead to convert all the sentences into propositions before they can be passed on for

Table 4: Comparison of RSC with and without Propositional Segmentation on BBC News Dataset.

Metric	RSC Without Propositions	RSC With Propositions
Answer Relevancy	**41.97**	42.95
Answer Faithfulness	**42.55**	39.51
Contextual Recall	**46.33**	45.14
Contextual Precision	**48.98**	47.99
Contextual Relevancy	11.56	**16.09**
Total Score ↑	191.39	**191.68**
Avg Retv Time(s) ↓	**0.716**	0.8183

chunking. However, it is an interesting area of study for the future.

Challenges of Agentic Chunking

Although not included as a formal baseline, we initially explored Agentic Chunking to assess the viability of LLM-based chunking pipelines. However, due to its high computational demand, it is excluded from comparative evaluation. Details of Agentic Chunking are mentioned in Section 2. Since the Agentic approach operates at the propositional level, so for this technique, on average, each proposition requires 6 to 7 calls to the LLM for chunk assignment and metadata updates. To start with, we use this technique on the BBC dataset. The dataset contained more than 75,000 propositions, but after 8 hours of processing, only 1,500 propositions were successfully assigned to chunks. Due to the high computational overhead, we discontinued the experimentation. Hence, high computational cost makes this approach impractical for large-scale datasets.

Despite its inefficiencies, Agentic Chunking may become viable in the future as LLMs improve in speed and affordability. However, for now, RSC provides a far more efficient and scalable solution.

The results and analysis confirm that RSC enhances retrieval efficiency and semantic coherence. Additionally, our findings highlight a new direction with propositional segmentation, which improves Contextual Relevancy. Overall, RSC consistently outperforms both Recursive Character Text Splitter and Semantic Chunking in Total Score and Contextual Relevancy, making it the preferred approach for RAG generation pipeline. Moving forward, future work will focus on optimizing propositional segmentation to reduce retrieval time, ensuring that the benefits of enhanced semantic coherence do not come at the expense of computational overhead.

7 Conclusion

Our work offers a targeted contribution to optimizing the chunking process in RAG-based systems. The proposed technique, Recursive Semantic Chunking maintains a balance between retrieval efficiency and context relevancy. The novelty of RSC lies in the recursive nature of the proposed method dynamically adjusting the chunk size and going beyond the traditional approaches. The results, evaluated against the traditional techniques i.e. recursive character split, semantic and agentic techniques highlight the superiority of the proposed methodology. Additionally, its robustness is validated across structured question-answering datasets and unstructured large-scale datasets, with evaluation based on relevancy, retrieval quality, and time efficiency. The evaluation is based on relevancy, retrieval quality and time efficiency. These findings have significant implications for RAG-based applications such as medical, finance, legal, and education etc. Looking forward, the retrieval time will be further optimized with respect to Recursive Semantic Chunking on varied datasets.

Limitation

The scope of this study is limited to textual data, and it can be widened to more complex document types which may include tables, codes etc. In addition, Recursive Semantic which depends on propositions provides a new direction. However, its high computational cost, despite yielding improved results, highlights the need for a more efficient and scalable approach.

References

LangChain AI. 2024. Langchain. GitHub repository.

Mohammad Alkhalaf, Ping Yu, Mengyang Yin, and Chao Deng. 2024. Applying generative ai with retrieval augmented generation to summarize and extract key clinical information from electronic health records. *Journal of Biomedical Informatics*, 156:104662.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA. Association for Computational Linguistics.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *Preprint*, arXiv:2401.08281.

Ruitao Feng, Xudong Hong, Mayank Jobanputra, Mattes Warning, and Vera Demberg. 2024. Retrieval-augmented modular prompt tuning for low-resource data-to-text generation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14053–14062, Torino, Italia. ELRA and ICCL.

FullStackRetrieval. 2024. Agentic chunker. Accessed: 2024-09-14.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 377–384, New York, NY, USA. Association for Computing Machinery.

Shiming He, Yu Hong, Shuai Yang, Jianmin Yao, and Guodong Zhou. 2024. Demonstration retrieval-augmented generative event argument extraction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 4617–4625, Torino, Italia. ELRA and ICCL.

Yi Jiang, Sendong Zhao, Jianbo Li, Haochun Wang, and Bing Qin. 2025. GainRAG: Preference alignment in retrieval-augmented generation through gain signal synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10746–10757, Vienna, Austria. Association for Computational Linguistics.

LangChain. 2023. Recursively split by character. Accessed: 2024-09-14.

LangChain. 2024. Langchain documentation. Accessed: 2024-10-31.

LangChain. 2024. Semantic chunker. Accessed: 2024-09-14.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Jerry Liu. 2022. LlamaIndex.

Puneet Mathur, Zhe Liu, Ke Li, Yingyi Ma, Gil Karen, Zeeshan Ahmed, Dinesh Manocha, and Xuedong Zhang. 2024. DOC-RAG: ASR language model personalization with domain-distributed co-occurrence retrieval augmentation. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5132–5139, Torino, Italia. ELRA and ICCL.

Carlo Merola and Jaspinder Singh. 2025. Reconstructing context: Evaluating advanced chunking strategies for retrieval-augmented generation. *Preprint*, arXiv:2504.19754.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

WA Sahlman, AM Ciechanover, and E Grandjean. 2023. Khanmigo: Revolutionizing learning with genai. *Harvard Business School Case*, pages 824–059.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *Preprint*, arXiv:2310.11324.

Spurthi Setty, Katherine Jijo, Eden Chung, and Natan Vidra. 2024. Improving retrieval for rag based question answering models on financial documents.

L. Siddharth and Jianxi Luo. 2024. Retrieval augmented generation using engineering design knowledge. *Knowledge-Based Systems*, 303:112410.

R. Teja. 2023. Evaluating the ideal chunk size for a rag system using llamaindex. Accessed: 2024-09-14.

Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. 2024. Financial report chunking for effective retrieval augmented generation. *Preprint*, arXiv:2402.05131.