2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT) | 979-8-3315-2228-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/AINIT65432.2025.11036011

LateSplit: Lightweight Post-Retrieval Chunking for Query-Aligned Text Segmentation in RAG Systems

Zirong Peng
lSchool of Computer Science,
Zhongyuan University of Technology
Zhengzhou, China
pengzirong@zut.edu.cn

Xiaoming Liu*
School of Computer Science, Zhongyuan University of Technology
Zhengzhou, China
ming616@zut.edu.cn

Guan Yang
School of Computer Science, Zhongyuan University of Technology
Zhengzhou, China
yangguan@zut.edu.cn

***Abstract*—Retrieval-Augmented Generation (RAG) has become indispensable for knowledge-intensive applications, enhancing large language models (LLMs) by integrating external information. However, the efficacy of RAG systems critically depends on text chunking strategies, where conventional pre-retrieval methods often produce misaligned or fragmented content, impairing downstream tasks. This paper introduces LateSplit, a hybrid chunking framework that combines conventional pre-retrieval chunking with novel post-retrieval refinement to better align documents with query intent. Building on standard chunking approaches, LateSplit performs dynamic boundary adjustment after retrieval, preserving logical coherence while improving relevance. As a lightweight, plug-and-play solution, LateSplit reduces computational overhead by 1.9× compared to semantic chunking techniques while achieving precision improvements of up to 44.2% and IoU gains of 20.4% across diverse datasets. Experimental results demonstrate its flexibility and practicality, offering a cost-effective enhancement for RAG pipelines.**

Keywords—Retrieval-Augmented Generation (RAG), Text chunking, Query-aligned segmentation, Post-retrieval refinement, Computational efficiency

I. Introduction

Retrieval-Augmented Generation (RAG) has emerged as a transformative paradigm in the field of natural language processing (NLP), addressing critical limitations of large language models (LLMs) including the lack of domain-specific knowledge, the issue of hallucinations, and the need to work with up-to-date or sensitive information [1, 2, 3, 4, 5]. By integrating a retriever module with a generator, RAG systems enable LLMs to augment their responses with external, relevant information, improving the quality and fidelity of generated content [6]. This makes RAG particularly valuable in knowledge-intensive tasks such as open-domain question answering, recommendation systems, and specialized decision support systems [7, 8].

While the efficacy of RAG systems has been widely demonstrated [6], their performance is critically dependent on the relevance and accuracy of the retrieved content [9, 10]. However, a recurring challenge in existing retrieval systems lies in the introduction of excessive or irrelevant content during retrieval. Such redundancy or noise can overwhelm the generation model, leading to degraded response quality [11]. Addressing this bottleneck requires not only advancements in retrieval algorithms but also improvements in the preprocessing and postprocessing of retrieved documents.

Traditional pre-retrieval chunking methods, whether rule-based or semantic, face inherent limitations. Fixed-length splits often sever logical connections between sentences, while semantic chunking may group content without regard to contextual progression [12]. These issues manifest acutely in downstream tasks - misaligned chunks introduce irrelevant content that degrades generation quality, or omit critical context needed for coherent responses [11]. Recent attempts to address this through sophisticated pre-processing (e.g., LLM-powered boundary detection [13]) incur prohibitive computational costs while still operating in the suboptimal preretrieval paradigm.

In this paper, we propose LateSplit, a novel post-retrieval chunking method designed to refine retrieved documents by aligning chunks more closely with the query. Unlike conventional approaches that rely on pre-retrieval chunking strategies, LateSplit first employs standard chunking methods, then operates after the retrieval step, dynamically adjusting the segmentation of text based on query relevance, as shown in Fig. 1. This approach addresses the mismatch often observed between pre-chunked documents and retrieval queries, enabling more focused and relevant retrieval. The initial chunking ensures scalable document processing, while the subsequent refinement focuses computational resources only on potentially relevant content.

To demonstrate the efficacy of LateSplit, we conducted extensive experiments across multiple datasets and evaluation benchmarks. Our results show that LateSplit achieves substantial improvements in retrieval precision and recall without compromising computational efficiency. Additionally, LateSplit enhances the balance between retrieval recall and precision, achieving query-aligned chunk refinement with minimal resource requirements. These results underscore the practical advantages of LateSplit as a cost-effective and scalable solution for improving RAG systems.

This work's main contributions are: (1) We propose LateSplit, the first post-retrieval chunking method that dynamically optimizes text segmentation using query context. By deferring boundary decisions until after retrieval, our approach better preserves logical flow and intent alignment compared to conventional pre-retrieval strategies. (2) LateSplit operates as a lightweight, plug-and-play solution, reducing computational overhead by 1.9× compared to semantic chunking techniques while achieving significant precision improvements of up to 29.8% and IoU gains of 13.1% across diverse datasets. (3) Experiments demonstrate LateSplit enhances retrieval quality without requiring costly retraining or architectural modifications, offering a cost-effective upgrade for existing RAG pipelines.

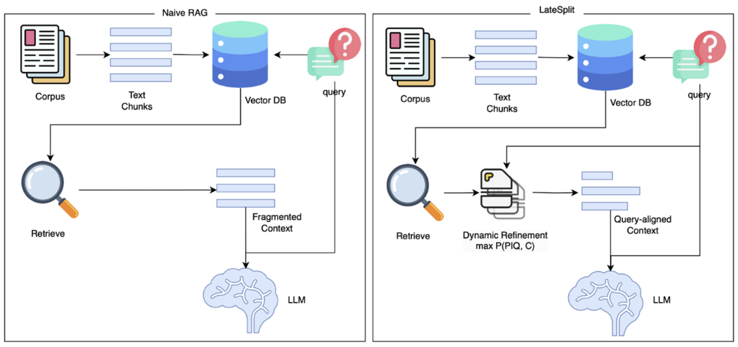

Fig. 1. Contrast between naive chunking (left) and LateSplit (right). Naive chunking splits documents into fixed segments before retrieval, risking fragmented or irrelevant context. LateSplit refines chunks after retrieval, dynamically adjusting boundaries to align with the query, ensuring logical coherence and relevance.

II. Realted Works

A. Text Segmentation Techniques

Text segmentation is a fundamental task in Natural Language Processing (NLP), involving the division of text into meaningful units such as words, sentences, or topics. This process is crucial for various downstream applications, including information retrieval and text summarization. Early methods for text segmentation relied on rule-based systems and statistical models. Techniques such as Latent Dirichlet Allocation (LDA) and Probabilistic Latent Semantic Analysis (PLSA) were employed for topic modeling to identify thematic boundaries within documents. However, these methods often struggled with capturing semantic relationships and contextual nuances. With the advent of deep learning, neural network architectures have been applied to text segmentation tasks. For instance, a model was introduced, introduced a model utilizing cross-segment attention mechanisms to improve segmentation accuracy by capturing dependencies across segments [14]. Similarly, another work framed text segmentation as a sentence-level sequence labeling task, employing BERT to encode multiple sentences simultaneously and predict segmentation points [15]. Recent advancements have focused on semantic-based segmentation, where embeddings are used to group semantically similar text chunks. This approach identifies segmentation points by monitoring significant changes in embedding distances, allowing for a more contextually aware division of text.

B. Chunking Strategies in RAG

In RAG systems, effective text chunking is vital for enhancing retrieval accuracy and, consequently, the quality of generated responses. Inefficient chunking can lead to incomplete contexts or the inclusion of irrelevant information, adversely affecting system performance. Traditional chunking methods often involve dividing text into fixed-length segments or utilizing syntactic cues such as punctuation and sentence boundaries. While straightforward, these approaches may not adequately capture the semantic coherence necessary for effective retrieval in RAG systems. Advanced chunking strategies leverage semantic information to create more meaningful text segments. For example, the author in [16] explored various chunking techniques, including semantic and agentic methods, to optimize the performance of language models in RAG applications. By identifying meaningful chunk boundaries based on content understanding, these methods aim to improve retrieval relevance and efficiency. In domain-specific contexts, such as financial reporting, chunking based on structural elements of documents has been proposed. [17] introduced an approach that dissects documents into constituent elements, facilitating the retrieval of contextually rich information and enhancing the accuracy of RAG-assisted question-answering tasks. Recognizing that a single chunking strategy may not be optimal for all scenarios, [18] proposed a Mix-of-Granularity (MoG) method. This approach dynamically determines the optimal granularity of knowledge databases based on input queries, allowing for flexible adjustment of chunk sizes to improve retrieval performance in RAG systems.

III. Methodology

The proposed LateSplit method integrates two sequential stages: conventional pre-retrieval chunking followed by query-aware post-retrieval refinement. This two-step process forms a unified solution that leverages the efficiency of conventional chunking while mitigating its inherent misalignment issues through context-aware post-retrieval optimization.

A. Pre-Retrieval Chunking

In the first stage, documents undergo conventional chunking using standard approaches (e.g., fixed-length or recursive splitting) to enable efficient document indexing. Given a document corpus $\mathcal{D}$, we split each document $d \in \mathcal{D}$ into chunks $\{c_1, \ldots, c_n\}$ using standard techniques:

$$\mathcal{C}(d) = \text{Chunker}(d; \theta) \qquad (1)$$

Where θ denotes configuration parameters (e.g., fixed chunk size, semantic thresholds). This design intentionally maintains backward compatibility with existing retrieval systems while providing a foundation for subsequent refinement. The framework accommodates arbitrary chunking strategies from simple fixed-length splitting to sophisticated semantic segmentation, ensuring compatibility with existing retrieval frameworks.

B. Post-retrieval Chunk Refinement

The second stage activates after retrieving top-k candidate chunks $C' \subseteq C$ relevant to query Q. Our method reprocesses this subset through its dynamic boundary adjustment mechanism analyzing both the query context and retrieved content to optimize passage alignment.

TABLE I. PERFORMANCE COMPARISON OF VARIOUS CHUNKING STRATEGIES WITH AND WITHOUT THE LATESPLIT APPROACH.

Chunker	recall（%）	precision(%)	precision_Ω(%)	IoU(%)	chunking time(s)	post processor time(s)
FixedToken	89.1	5.2	21.0	5.2	0.20	0.00
+ LateSplit w/o extra	87.1 (-2.2%)	5.9 (+14.1%)	30.3 (+44.2%)	5.9 (+13.6%)	0.20	15.47
+ LateSplit w/ extra	88.6 (-0.6%)	5.6 (+8.8%)	26.3 (+25.2%)	5.6 (+8.6%)	0.19	15.45
RecursiveToken	86.6	6.7	29.9	6.7	0.59	0.00
+ LateSplit w/o extra	84.5 (-2.4%)	7.6 (+13.1%)	40.2 (+34.4%)	7.5 (+12.6%)	0.59	14.45
+ LateSplit w/ extra	86.3 (-0.3%)	7.1 (+6.0%)	35.1 (+17.3%)	7.1 (+5.9%)	0.58	14.48
SemanticGradient	87.6	3.7	18.8	3.7	85.65	0.00
+ LateSplit w/o extra	86.5 (-1.3%)	4.1 (+9.1%)	24.6 (+31.3%)	4.1 (+8.9%)	85.39	15.62
+ LateSplit w/ extra	87.5 (-0.0%)	3.9 (+5.0%)	22.3 (+18.6%)	3.9 (+5.0%)	85.42	15.73
SemanticIQR	87.6	3.7	18.8	3.7	85.76	0.00
+ LateSplit w/o extra	86.5 (-1.3%)	4.1 (+9.1%)	24.6 (+31.3%)	4.1 (+8.9%)	85.45	15.59
+ LateSplit w/ extra	87.5 (-0.0%)	3.9 (+5.0%)	22.3 (+18.6%)	3.9 (+5.0%)	85.48	15.50
SemanticPercentile	87.6	3.7	18.8	3.7	85.67	0.00
+ LateSplit w/o extra	86.5 (-1.3%)	4.1 (+9.1%)	24.6 (+31.3%)	4.1 (+8.9%)	85.39	15.74
+ LateSplit w/ extra	87.5 (-0.0%)	3.9 (+5.0%)	22.3 (+18.6%)	3.9 (+5.0%)	85.68	15.55
SemanticPercentile	89.4	2.0	10.7	2.0	86.17	0.00
+ LateSplit w/o extra	89.1 (-0.3%)	2.1 (+4.2%)	13.4 (+25.5%)	2.1 (+4.1%)	86.28	11.24
+ LateSplit w/ extra	89.4 (+0.0%)	2.1 (+2.5%)	12.4 (+16.6%)	2.1 (+2.5%)	86.50	11.09
SemanticStdDev	87.6	3.7	18.8	3.7	85.73	0.00
+ LateSplit w/o extra	86.5 (-1.3%)	4.1 (+9.1%)	24.6 (+31.3%)	4.1 (+8.9%)	85.35	15.62
+ LateSplit w/ extra	87.5 (-0.0%)	3.9 (+5.0%)	22.3 (+18.6%)	3.9 (+5.0%)	85.52	15.67
ClusterSemantic	81.2	8.4	37.1	8.3	29.84	0.00
+ LateSplit w/o extra	78.5 (-3.3%)	9.7 (+15.4%)	49.3 (+32.8%)	9.4 (+14.2%)	29.90	13.93
+ LateSplit w/ extra	80.9 (-0.4%)	9.0 (+7.0%)	43.7 (+17.8%)	8.8 (+6.8%)	29.89	13.88

The goal of chunk refinement is to identify and extract a passage $P \subseteq C'$ that maximally satisfies the user query. The process begins by jointly encoding the concatenation of Q and C', yielding a sequence of input tokens $\{t_1, t_2, \dots, t_N\}$. Offset mappings $\{(s_1, e_1), (s_2, e_2), \dots, (s_N, e_N)\}$ are then obtained to indicate the character spans of each token. Subsequently, a pre-trained language model processes the tokenized input to produce logits $\boldsymbol{L}_{start}$ and $\boldsymbol{L}_{end}$ for potential start and end positions of the relevant passage:

$$L_{start}, L_{end} = \text{Model}(t_1, t_2, \dots, t_N) \tag{2}$$

A softmax function is applied to these logits to generate probability distributions over all possible start and end indices:

$$\text{P}_{start}(\text{i}) = \frac{\text{e}^{\text{L}_{start}[\text{i}]}}{\sum_\text{j} \text{e}^{\text{L}_{start}[\text{j}]}}, \quad \text{P}_{end}(\text{i}) = \frac{\text{e}^{\text{L}_{end}[\text{i}]}}{\sum_\text{j} \text{e}^{\text{L}_{end}[\text{j}]}} \tag{3}$$

To accommodate the diverse chunking requirements of users, a tunable parameter λ is introduced, enabling adjustments to cater to varying needs. Specifically, a dynamic threshold $\tau = \lambda/N$ is employed to determine the top k start and end positions, satisfying $P_{\text{start}}(i) > \tau$ and $P_{\text{end}}(i) > \tau$, where $k = \lceil 0.01/\tau \rceil$. This mechanism enables precision-recall tradeoff control: higher λ values produce shorter, more focused chunks while lower values preserve broader context. The final refined passage P becomes:

$$P = arg\max_{P \subseteq C'} \left[\max_{i \in \mathcal{B}_{start}} P_{\text{start}}(i) \times \max_{j \in \mathcal{B}_{end}} P_{\text{end}}(j) \right] \tag{4}$$

where $\mathcal{B}_{start}$ and $\mathcal{B}_{end}$ denote boundary sets for start/end positions. To ensure coherence and contextual completeness, the passage boundaries are further refined by expanding to the nearest sentence boundaries, typically defined by punctuation or newline characters.

C. Theoretical Analysis

LateSplit leverages the probabilistic outputs of language models to identify the most relevant segments within retrieved documents. By focusing on high-probability start and end positions, LateSplit effectively narrows down the context to passages that are semantically aligned with the query. This approach reduces the inclusion of irrelevant or noisy information, thereby enhancing the precision of the RAG system.

Formally, let $P(P|Q,C)$ denote the probability of passage P given query Q and context C. LateSplit aims to maximize this probability by selecting P such that:

$$P(P|Q,C) \propto \exp\left(\sum_{i\in\mathcal{B}_{start}} \alpha_i + \sum_{j\in\mathcal{B}_{start}} \beta_j\right) \qquad (5)$$

where α_i, β_j represent start/end position logits. This formulation ensures that the selected passage P is the most probable segment that satisfies the query's informational needs.

IV. Experiments

In this section, we detail the experimental setup employed to evaluate the performance of the proposed LateSplit method. We describe the datasets utilized, the baseline chunking strategies for comparison, the evaluation metrics, and the implementation specifics.

A. Datasets and Metrics

To assess the effectiveness of LateSplit, we employed the Chunk Evaluation dataset, a comprehensive benchmark designed for evaluating chunking strategies in retrieval-based AI applications [19]. This dataset encompasses a diverse range of corpora, including: State of the Union Address, Wikitext, Chatlogs, Finance and Pubmed. Evaluation metrics were precision, recall, and intersection-over-union (IoU), providing a comprehensive assessment of retrieval performance at the token level.

B. Baselines

We compared LateSplit against several baseline chunking methods, including:

Fixed Token chunker [20]: A rule-based chunking method commonly used in many RAG systems as the default. It divides text into chunks with a fixed number of tokens.

Recursive Token Chunker [20]: A widely used method that splits text based on a specified chunk size using a predefined set of characters. In this paper, we employ the separators ["\n\n", "\n", ".", "?", "!", " ", ""] as the default.

Semantic Chunker [21]: An advanced chunking method based on semantic similarity. It detect discontinuities by computing embedding differences between consecutive sentences. When the difference exceeds a specified threshold, a split is made. Supported split types include percentile, standard deviation, interquartile, and gradient methods.

Cluster Semantic Chunker [19]: An embedding-model aware chunking method, which produces globally optimized chunks by maximizing the total cosine similarity within chunks using a dynamic programming method, constrained by a user-defined maximum length. By preserving as much semantic consistency as possible across the document, it compacts relevant information into each chunk.

C. Experiment Settings

For the LateSplit method, we utilized the RoBERTa-base model fine-tuned on the SQuAD2 dataset [22, 23], with no additional fine-tuning applied. The system performed dense retrieval from a vector database using the BGE-M3 embedding model, consistently applied for both semantic chunking, with top-k set to 5. All experiments were conducted using an NVIDIA RTX A5000 GPU.

V. Result and Analysis

A. Overall Performance

Tab. I presents a comprehensive comparison of various chunking strategies, both with and without the integration of the proposed LateSplit method. LateSplit demonstrated consistent improvements across multiple chunking strategies, particularly enhancing precision and IoU metrics. When applied to the FixedToken baseline, LateSplit without extra processing increased IoU by 13.6%(from 5.2 to 5.9) and precision by 44.2% (from 21.0 to 30.3), with only a marginal recall decline of 2.2%. Similar gains were observed for RecursiveToken, where IoU improved by 16.2% (6.7 to 7.4) and **precision_Ω** by 34.4% (29.9 to 40.2). The ClusterSemantic method, already the strongest baseline, saw further gains: IoU increased by 14.2% (8.3 to 9.4) and **precision_Ω** by 38.8% (37.1 to 49.3) for chunk size 200. These results validate LateSplit's ability to refine chunk boundaries post-retrieval, aligning them more closely with query intent.

Effect of Chunk Size: Analyzing the results across different chunk sizes reveals a consistent pattern in the performance metrics, as shown in Tab. II. As the chunk size increases from 200 to 400 tokens, the baseline FixedToken chunker exhibits a decrease in IoU from 5.2 to 2.8, indicating less overlap with the ground truth segments. This reduction suggests that larger chunk sizes may encompass more irrelevant information, thereby diluting the relevance of each chunk. However, the integration of LateSplit mitigates this decline by enhancing IoU across all chunk sizes. For instance, with a chunk size of 200 tokens and no overlap, LateSplit improves IoU by 14.1% (5.2 to 5.9). In terms of recall, larger chunk sizes inherently capture more content per chunk, which can lead to higher recall as more relevant information is included within each chunk. However, our results indicate that while the baseline recall remains relatively high across all chunk sizes (ranging from 89.1 to 93.7), the application of LateSplit generally results in slight decreases in recall. For example, with a chunk size of 400 tokens and no overlap, recall decreases from 91.5 to 91.2. Despite these minor reductions, the overall recall remains substantial, ensuring that the majority of relevant information is still captured.

Impact of Overlap: Overlap between chunks plays a critical role in balancing recall and precision, as shown in Tab. II. At a fixed chunk size of 200 tokens, introducing an overlap of 100 tokens leads to a baseline IoU of 5.2, which increases to 5.8 with LateSplit. The increased overlap allows for better coverage of relevant information, slightly offsetting the minor reductions in recall observed with LateSplit integration. Specifically, for a chunk size of 200 tokens with a 100-token overlap, recall decreases marginally from 89.5 to 89.4. Similarly, at a chunk size of 300 tokens with an overlap of 150 tokens, LateSplit enhances IoU by 10.7%. In this configuration, recall remains virtually unchanged with LateSplit (-0.0%), demonstrating that increased overlap can effectively maintain recall levels while still benefiting from the precision enhancements offered by LateSplit.

TABLE II. Impact of chunk size and overlap on LateSplit performance with FixedToken baseline.

Chunker	Size	Overlap	recall(%)	precision(%)	precision$_\Omega$(%)	IoU(%)
FixedToken	200	0	89.1	5.2	21.0	5.2
+ LateSplit w/o extra	200	0	87.1 (-2.2%)	5.9 (+14.1%)	30.3 (+44.2%)	5.9 (+13.6%)
+ LateSplit w/ extra	200	0	88.6 (-0.6%)	5.6 (+8.8%)	26.3 (+25.2%)	5.6 (+8.6%)
FixedToken	200	100	89.5	5.2	14.6	5.2
+ LateSplit w/o extra	200	100	89.0 (-0.6%)	6.3 (+20.6%)	21.0 (+43.2%)	6.2 (+20.4%)
+ LateSplit w/ extra	200	100	89.4 (-0.1%)	5.8 (+11.9%)	18.5 (+26.4%)	5.8 (+11.8%)
FixedToken	300	0	91.1	3.6	15.5	3.6
+ LateSplit w/o extra	300	0	90.0 (-1.3%)	4.1 (+13.1%)	23.4 (+51.4%)	4.1 (+13.0%)
+ LateSplit w/ extra	300	0	90.8 (-0.3%)	4.0 (+9.2%)	20.4 (+31.8%)	4.0 (+9.1%)
FixedToken	300	150	90.7	3.6	10.6	3.6
+ LateSplit w/o extra	300	150	90.2 (-0.5%)	4.2 (+16.1%)	14.7 (+38.1%)	4.2 (+16.1%)
+ LateSplit w/ extra	300	150	90.7 (-0.0%)	4.0 (+10.7%)	13.5 (+26.9%)	4.0 (+10.7%)
FixedToken	400	0	91.5	2.8	12.5	2.8
+ LateSplit w/o extra	400	0	90.2 (-1.4%)	3.0 (+9.4%)	17.2 (+37.9%)	3.0 (+9.3%)
+ LateSplit w/ extra	400	0	91.2 (-0.3%)	3.0 (+6.9%)	15.6 (+24.9%)	3.0 (+6.8%)
FixedToken	400	200	93.7	2.8	8.4	2.8
+ LateSplit w/o extra	400	200	93.6 (-0.1%)	3.2 (+13.5%)	10.9 (+28.9%)	3.2 (+13.5%)
+ LateSplit w/ extra	400	200	93.7 (-0.0%)	3.1 (+9.3%)	10.2 (+21.0%)	3.1 (+9.3%)

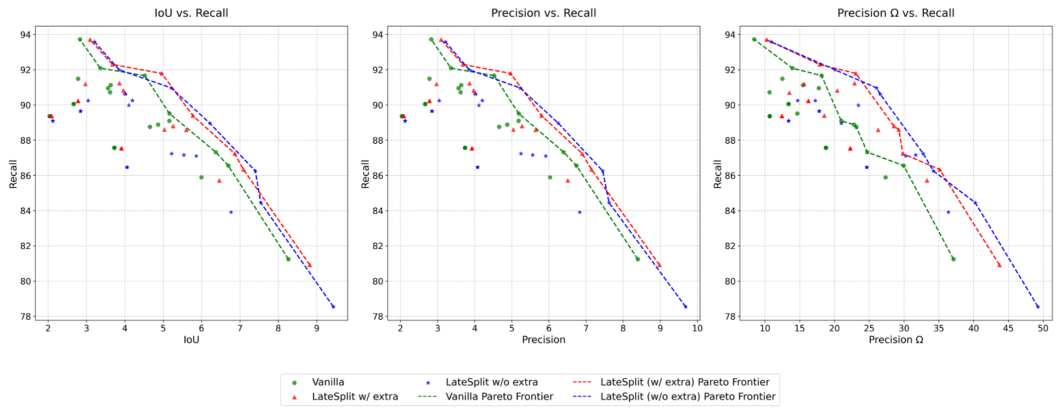

Fig. 2. Pareto Frontiers comparing chunking methods with and without LateSplit. The LateSplit-enhanced settings uniformly dominate the baseline settings across all three metrics, indicating improved efficiency and effectiveness when balancing recall with IoU, precision, and precision$_\Omega$.

B. *Comparison with Semantic Chunking Methods*

The performance characteristics of LateSplit and semantic chunking methods, such as ClusterSemanticChunker, reveal distinct trade-offs in precision, recall, and computational efficiency. While ClusterSemanticChunker leverages semantic coherence to optimize chunk boundaries, its reliance on pre-retrieval chunking can result in segments misaligned with query intent, particularly when semantic boundaries diverge from contextual relevance. In contrast, LateSplit dynamically refines chunk boundaries post-retrieval, prioritizing alignment with the query's informational needs. For instance, when applied to a non-semantic baseline (e.g., FixedToken, chunk size 200), LateSplit achieves a precision$_\Omega$ of 30.3 and an IoU of 5.9, surpassing SemanticGradient (precision$_\Omega$: 18.8, IoU: 3.7) by 61.2% and 59.5%, respectively. This demonstrates LateSplit's capacity to enhance precision and granularity even when operating on non-semantic initial chunks.

Notably, ClusterSemanticChunker, as a sophisticated semantic method, achieves higher absolute precision$_\Omega$ (37.1) and IoU (8.3) but incurs substantial computational costs (29.84s for chunking). LateSplit, when applied independently to non-semantic chunking, balances performance and efficiency: its total processing time (15.67s for chunking + post-processing) is 5.5× faster than SemanticGradient (85.65s) and 1.9× faster than ClusterSemanticChunker. While ClusterSemanticChunker excels in precision, it sacrifices recall (81.2 vs. LateSplit's 87.1), underscoring LateSplit's ability to retain broader contextual coverage while refining relevance.

In terms of time efficiency, LateSplit's post-processing overhead (11.1–24.7s) remains negligible compared to the resource demands of semantic methods. For example, ClusterSemanticChunker requires 29.84s for chunking alone, while SemanticGradient demands 85.65s-a prohibitive cost for real-time applications. LateSplit's lightweight design bridges this gap, offering precision improvements over basic semantic techniques (e.g., SemanticGradient) and competitive efficiency against advanced methods like ClusterSemanticChunker. These results position LateSplit as a pragmatic solution for scenarios requiring query-aligned chunking without the computational burden of full semantic processing.

C. *Pareto Frontier Analysis*

To holistically evaluate the trade-offs between key performance metrics, we analyze the Pareto frontiers of chunking methods with and without LateSplit across three critical dimensions: IoU versus Recall, Precision versus Recall, and precision_Ω versus Recall. The Pareto Frontier illustrates the best possible trade-offs between precision, recall, and IoU, where no improvement in one metric is possible without sacrificing another. As illustrated in Fig. 2, LateSplit-enhanced methods consistently dominate baseline strategies across all metrics, demonstrating superior efficiency in balancing competing objectives.

The uniform dominance of LateSplit across these frontiers underscores its effectiveness as a post-retrieval optimization layer. By dynamically adjusting chunk granularity based on query context, LateSplit resolves the inherent tension between recall (coverage) and precision (relevance), offering a more efficient operating point for real-world RAG systems. This contrasts with static chunking methods, which force a suboptimal compromise between these metrics due to fixed segmentation rules applied during preprocessing.

VI. Coclusion

LateSplit introduces dynamic post-retrieval chunking to align text segments with query intent, addressing misalignment issues in conventional pre-retrieval methods. By refining chunks after retrieval, it achieves precision improvements of up to 44.2% and IoU gains of 20.4% across datasets while reducing computational overhead by 1.9× compared to semantic chunking. This lightweight solution enhances RAG systems through contextually coherent retrieval, offering an optimal balance between relevance and efficiency

Acknowledgment

This work is supported by Key Scientific Research Project of Higher Education Institutions in Henan Province (24A520058). These contributions have been instrumental in facilitating our research endeavors and are deeply appreciated.

References

[1] H. He, H. Zhang, and D. Roth, "Rethinking with retrieval: Faithful large language model inference," arXiv preprint arXiv:2301.00303, 2022.

[2] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Liu, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," arXiv preprint arXiv:2308.07107, 2023.

[3] Y. Chen, Q. Fu, Y. Yuan, Z. Wen, G. Fan, D. Liu, D. Zhang, Z. Li, and Y. Xiao, "Hallucination detection: Robustly discerning reliable answers in large language models," in Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 245–255.

[4] X. Shen, Z. Chen, M. Backes, and Y. Zhang, "In chatgpt we trust? measuring and characterizing the reliability of chatgpt," arXiv preprint arXiv:2304.08979, 2023.

[5] S. Zeng, J. Zhang, P. He, Y. Xing, Y. Liu, H. Xu, J. Ren, S. Wang, D. Yin, Y. Chang et al., "The good and the bad: Exploring privacy issues in retrieval-augmented generation (rag)," arXiv preprint arXiv:2402.16893, 2024.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. K¨ uttler, M. Lewis, W.-t. Yih, T. Rockt¨ aschel et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[7] K. J. Prabhod, "Ai-driven insights from large language models: Implementing retrieval-augmented generation for enhanced data analytics and decision support in business intelligence systems," Journal of Artificial Intelligence Research, vol. 3, no. 2, pp. 1–58, 2023.

[8] D. Di Palma, "Retrieval-augmented recommender system: Enhancing recommender systems with large language models," in Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 1369–1373.

[9] C.-H. Tan, J.-C. Gu, C. Tao, Z.-H. Ling, C. Xu, H. Hu, X. Geng, and D. Jiang, "Tegtok: Augmenting text generation via task-specific and open-world knowledge," in Findings of the Association for Computational Linguistics: ACL 2022, 2022, pp. 1597–1609.

[10] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Sch¨ arli, and D. Zhou, "Large language models can be easily distracted by irrelevant context," in International Conference on Machine Learning. PMLR, 2023, pp.31 210–31 227.

[11] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective retrieval augmented generation," arXiv preprint arXiv:2401.15884, 2024.

[12] J. Zhao, Z. Ji, P. Qi, S. Niu, B. Tang, F. Xiong, and Z. Li, "Meta-chunking: Learning efficient text segmentation via logical perception," arXiv preprint arXiv:2410.12788, 2024.

[13] A. Duarte, J. Marques, M. Grac¸a, M. Freire, L. Li, and A. Oliveira, "Lumberchunker: Long-form narrative document segmentation," in Findings of the Association for Computational Linguistics: EMNLP 2024, 2024, pp. 6473–6486.

[14] M. Lukasik, B. Dadachev, K. Papineni, and G. Sim˜ oes, "Text segmentation by cross segment attention," in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Nov. 2020, pp. 4707–4716.

[15] Q. Zhang, Q. Chen, Y. Li, J. Liu, and W. Wang, "Sequence model with self-adaptive sliding window for efficient spoken document segmentation," in 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE, 2021, pp. 411–418.

[16] M. Borck. (2024) Optimising Language Models with Advanced Text Chunking Strategies. A. J. Yepes, Y. You, J. Milczek, S. Laverde, and R. Li, "Financial report chunking for effective retrieval augmented generation," arXiv preprint arXiv:2402.05131, 2024.

[17] Z. Zhong, H. Liu, X. Cui, X. Zhang, and Z. Qin, "Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation," arXiv preprint arXiv:2406.00456, 2024.

[18] B. Smith and A. Troynikov, "Evaluating chunking strategies for retrieval," July 2024. [Online]. Available: https://research.trychroma.com/evaluating-chunking

[19] Langchain, "Langchain," 2023.

[20] G. Kamradt, "Semantic chunking," https://github.com/FullStackRetrieval-com/RetrievalTutorials, 2024.

[21] Y. Liu, "Roberta: A robustly optimized bert pretraining approach," arXiv preprint arXiv:1907.11692, vol. 364, 2019.

[22] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for squad," in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018, pp. 784–789.

[23] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," in Findings of the Association for Computational Linguistics: ACL 2024, Aug. 2024, pp. 2318–2335.