The Impact of Chunking Strategies on Domain-Specific Information Retrieval in RAG Systems

1st Maximilian Stäbler
Institute for AI Safety & Security
German Aerospace Center (DLR)
Ulm, Germany
maximilian.staebler@dlr.de

2nd Steffen Turnbull
Institute for AI Safety & Security
German Aerospace Center (DLR)
Sankt Augustin, Germany
steffen.turnbull@dlr.de

3rd Tobias Müller
Industry-University Collaboration
SAP SE
München, Germany
tobias.mueller15@sap.com

4th Chris Langdon
Drucker School of Business
Claremont Graduate University
Claremont, USA
chris.langdon@cgu.edu

5th Jorge Marx-Goméz
Deptarment of Business Informatics
University of Oldenburg
Oldeburg, Germany
jorge.marx.gomez@uol.de

6th Frank Köster
Institute for AI Safety & Security
German Aerospace Center (DLR)
Sankt Augustin, Germany
frank.koester@dlr.de

***Abstract*—We benchmark 90 chunker–model configurations across seven arXiv domains (2 520 retrieval runs) and show that a sentence-based splitter with a `512`-token window and `200`-token overlap reaches the highest token-level Intersection-over-Union (IoU ≈ 0.099) while remaining compute-efficient. Our study systematically pairs seven open-source embedding models with semantic and fixed-size chunking strategies, measuring their impact on retrieval quality and latency in Retrieval-Augmented Generation (RAG) pipelines. Results reveal that (i) sentence splitting consistently outperforms alternative heuristics, (ii) smaller embeddings deliver more stable cross-domain performance than larger ones, and (iii) finance texts benefit most, whereas astrophysics lags. The accompanying code provides practitioners with empirically grounded guidelines for selecting chunking–embedding combinations that balance accuracy and efficiency.**

***Index Terms*—RAG, Information Retrieval, Chunking.**

I. Introduction

Despite the centrality of retrieval-augmented generation (RAG) pipelines in modern language-centric applications, the *interaction* between document–chunking heuristics and embedding models remains under-explored, especially when systems are deployed across heterogeneous knowledge domains. Existing benchmarks either evaluate full-document ranking or rely on coarse Top-k metrics, offering little guidance on how chunk size, overlap, and semantic cohesion influence token-level relevance, compute cost, and, ultimately, answer quality [1], [2].

RAG addresses the context-length and freshness limits of pretrained language models by retrieving grounded snippets before generation [3], [4]. Sophisticated variants now add overlapping windows, recurrence, or agentic controllers [5]. However, practical design still hinges on two levers: *how* a document is partitioned and *which* embedding space best encodes those partitions.

979-8-3315-2037-3/25/$31.00

Enterprises are increasingly leveraging large language models to mine corpora in legal, finance, healthcare, and technical domains [6]. In these settings, passing entire documents through a model is often infeasible—context windows are limited, and irrelevant tokens dilute the signal, raising latency and degrading response quality. Document chunking, therefore, functions as a critical pre-processing step [7], [8]. Yet no systematic, domain-level comparison of chunker–embedding combinations exists.

Contributions

1) We benchmark **90** chunker–model configurations across **seven** scientific domains, executing **2 520** retrieval runs with a token-level Intersection-over-Union (IoU) metric.
2) We demonstrate that sentence-based splitting with a 512-token window and 200-token overlap achieves the top IoU (≈ 0.099) while remaining computationally efficient.
3) Results reveal that (i) sentence splitting consistently outperforms alternative heuristics, (ii) smaller embeddings deliver more stable cross-domain performance than larger ones, and (iii) finance texts benefit most, whereas astrophysics lags.
4) We release reproducible code and empirical guidelines to help practitioners select chunking–embedding pairs tailored to domain characteristics and resource budgets.

II. Related Work

1) Chunking Techniques: Early RAG deployments relied on fixed-size windows for their simplicity, yet these ignore sentence or structural boundaries and may split atomic facts [3]. Semantic and hybrid strategies attempt to preserve discourse

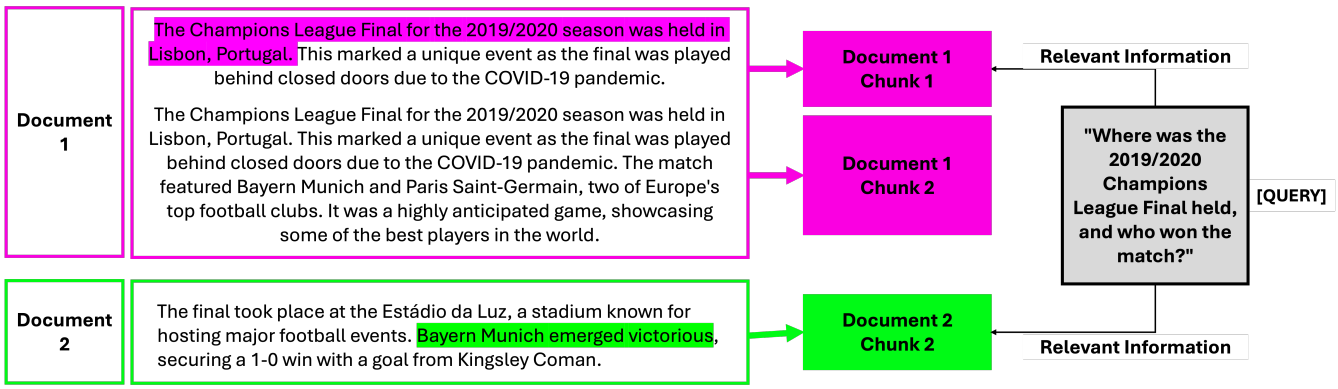

Fig. 1. The challenges of text chunking in retrieval systems: an example query about the 2019/2020 Champions League final shows how relevant cues (location and winner) are distributed across chunks and documents. Pink regions stem from *Document 1*, while green regions stem from *Document 2*, underscoring how critical facts can be fragmented irrespective of the chunking policy.

coherence at the expense of additional computation [9]–[11]. Empirical work suggests that 250–512-token chunks maximise retrieval precision, whereas 1 000–2 000 tokens favour context retention for summarisation [6]. More recent agentic pipelines such as Meta's JESTR-RAG integrate dynamic overlap and recurrence but still hinge on the underlying chunk boundaries. The debate over cost–benefit trade-offs, therefore, remains open [10], [12]. Traditional benchmarks emphasize document ranking, whereas LLMs are indifferent to the position of relevant information within their context window. This necessitates a more sophisticated approach to evaluating chunking strategies, as illustrated in Figure 1.

2) Token-level Evaluation: Traditional IR benchmarks (e.g., MTEB) report document-level Top-k scores that overlook how LLMs treat all tokens in their context window uniformly [1]. Smith et al. introduced Intersection-over-Union (IoU) to measure token-wise overlap between retrieved context and gold spans, capturing both redundancy and noise [2].

TABLE I
RANKING VS. TOKEN-LEVEL METRICS.

Metric	Granularity	Penalizes Noise?
Top-k Accuracy	Chunk / Document	No
IoU (ours)	Token	Yes

3) Domain-aware Retrieval: Chunk effectiveness is modulated by genre and structure: structural-element splitting outperforms paragraphs in financial filings [13], while medical vision–language models benefit from anatomy-guided segmentation [14]. Cross-domain studies further report that the interaction between chunkers and embedding models varies widely, affecting both accuracy and latency [10], [15]. Nevertheless, systematic comparisons spanning *multiple* domains—and evaluated with token-level metrics—are scarce.

Our work addresses these gaps by jointly analysing chunker–embedding combinations across seven scientific fields with IoU, providing the first large-scale, domain-sensitive benchmark to inform real-world RAG design.

III. Methodology

This section details the chunking approaches, metrics, and query–answer (Q&A) generation pipeline used to benchmark RAG systems.

A. Chunking Strategies

We compare three families of splitters—*semantic*, *sentence*, and *token*. All are implemented via `LlamaIndex` node parsers to ensure a uniform API.

SemanticSplitterNodeParser: Semantic boundaries are detected by computing cosine similarity between adjacent sentence embeddings (OpenAI `gpt-4o-mini`). Split points are chosen when similarity falls below the {95, 85, 75}-percentile of the document-level distance distribution. Across the seven-domain corpus, these thresholds correspond to progressively finer granularity: the 95 % cut produces the fewest but longest chunks (median length $>$ 500 tokens), 85 % roughly halves that length, and 75 % yields the smallest, highly local segments. Figure 2 in the appendix visualises the resulting length histograms.

SentenceSplitter: Text is broken at sentence boundaries and then packed into windows of {128, 256, 512} tokens with optional overlaps of {0, 75, 200} tokens. Triple new-lines (`\n\n\n`) mark paragraph breaks to preserve structure.

TokenTextSplitter: A deterministic tokenizer-level splitter creates chunks of 1,024 tokens (default) with overlaps {0, 25, 50, 75}. This ensures compatibility with any embedding model's context window, while providing direct control over retrieval costs.

B. Evaluation Metrics

Retrieval quality is judged at **token level**. Let t_e be the set of gold tokens and t_r the tokens returned by the retriever; we compute

$$\text{IoU} = \frac{|t_e \cap t_r|}{|t_e| + |t_r| - |t_e \cap t_r|} \tag{1}$$

A high Intersection-over-Union balances completeness and noise by simultaneously rewarding recall and penalising irrelevant or redundant tokens. Precision $P = |t_e \cap t_r|/|t_r|$ and recall $R = |t_e \cap t_r|/|t_e|$ are reported inline with IoU rather than as separate equations.

C. Retrieval-Evaluation Pipeline

For each of the 28 source papers, we generate ten query–answer pairs (total 260) using `gpt-4o-mini`:

> *"You are a domain expert. Craft one fact-based question that can be answered exclusively from the passage below. Return JSON with fields 'question', 'answer_start', 'answer_end'."*

The model was chosen because it is architecturally aligned with the embedding family and offers a favourable cost-to-quality ratio—roughly 3× cheaper than `gpt-4o-large` at identical temperature settings, enabling 2 520 retrieval runs within a single-GPU budget.

Generated answer spans are validated by enforcing exact string matches in the source text and de-duplication against the previous ten questions to guarantee coverage diversity. This automated yet traceable procedure keeps the evaluation set disjoint from any embedding-model training data and can be optionally complemented by human raters for critical applications.

IV. Experiments and Results

This section details and analyzes our experiments.

A. Baseline Choices

To evaluate the performance of semantic interoperability across different embedding approaches, we carefully selected a diverse range of models and established rigorous experimental parameters.

a) Embedding Models: We selected seven embedding models, each representing a different architecture and parameter size. As benchmark models, we included OpenAI's *text-embedding-3-small* (1.5 billion parameters) and *text-embedding-3-large* (3 billion parameters). For open-source alternatives, we selected five leading models from the Ollama platform: *mxbai-embed-large* (335M parameters), *all-minilm* (22M parameters), *bge-m3* (567M parameters), *nomic-embed-text* (1.5B parameters), and *all-minilm-33m* (33M parameters). This selection offers a comprehensive comparison across various model sizes and architectures, enabling us to examine the trade-offs between model complexity and embedding quality.

b) Input Data Selection: To ensure consistent quality and structural comparability across domains, we utilized research papers from arXiv[1] as our primary data source. We selected papers from seven distinct domains: statistics, astrophysics, computer science, economics, finance, engineering, and biology. For each domain, we analyzed five peer-reviewed papers, ensuring high-quality content with standardized academic formatting. This approach minimizes potential bias from varying text qualities or formatting inconsistencies that might affect embedding performance. For PDF to text conversion, we utilized *PyMuPDF*[2], a high-performance Python library that enables efficient document processing and data extraction.

c) Text Chunking Approaches: For text segmentation, we implemented three established chunking methods described in Section III-A.

The configuration parameters for each chunker were selected based on established best practices and technical documentation. The *SentenceSplitter* was configured with standard sentence detection rules, the *SemanticSplitterNodeParser* with semantic coherence thresholds, and the *TokenTextSplitter* with chunk sizes optimized for the embedding models' context windows.

B. Experimental and Implementation Details

Our experimental setup encompassed 90 distinct parameter configurations, combining 7 embedding models with 15 different chunking configurations (3 SemanticSplitterNodeParser, 6 SentenceSplitter, and 6 TokenTextSplitter variants). Each configuration was tested on 4 representative documents from 7 academic domains, resulting in $2,520$ individual experiments. For evaluation purposes, we generated a question-answer dataset comprising 10 questions per document, specifically designed to reference key text passages [2]. After rigorous filtering to ensure quality and relevance, we established a benchmark set of 260 questions with verified correct answers, serving as our primary performance metric baseline. The implementation framework was developed in Python, utilizing the Ollama platform for embedding model deployment. All experiments were accelerated using an NVIDIA RTX4090 GPU. To ensure reproducibility and scalability, our implementation supports distributed computing through remote GPU integration, specifically enabling connection to cloud-based GPU services such as RunPod.io[3] through remote Ollama instances. The full code implementation with all related input data will be made available on GitHub [4].

a) Throughput and cost analysis: To quantify the computational footprint behind the "good-and-fast" quadrant in Fig. 3, we measured the time each model needs to embed 1 000 chunks on a single RTX 4090 (FP16, batch size 32). Table II lists those wall-clock seconds together with the average token payload of the chunking configuration that yielded the model's peak IoU. The figures reveal a clear three-tier landscape: lightweight embedders such as *all-minilm* finish in under 10s, medium commercial models stay below 15s, while heavyweight open-source models roughly double the cost. These numbers provide a concrete speed-to-quality trade-off for practitioners planning large-scale deployments.

C. Results and Discussion

As illustrated in Figures 3 and 4, our experiments reveal several significant insights about the relationship between

[1]https://arxiv.org/

[2]https://github.com/pymupdf/PyMuPDF

[3]https://www.runpod.io/

[4]https://github.com/maxistaebler/COINS_2025

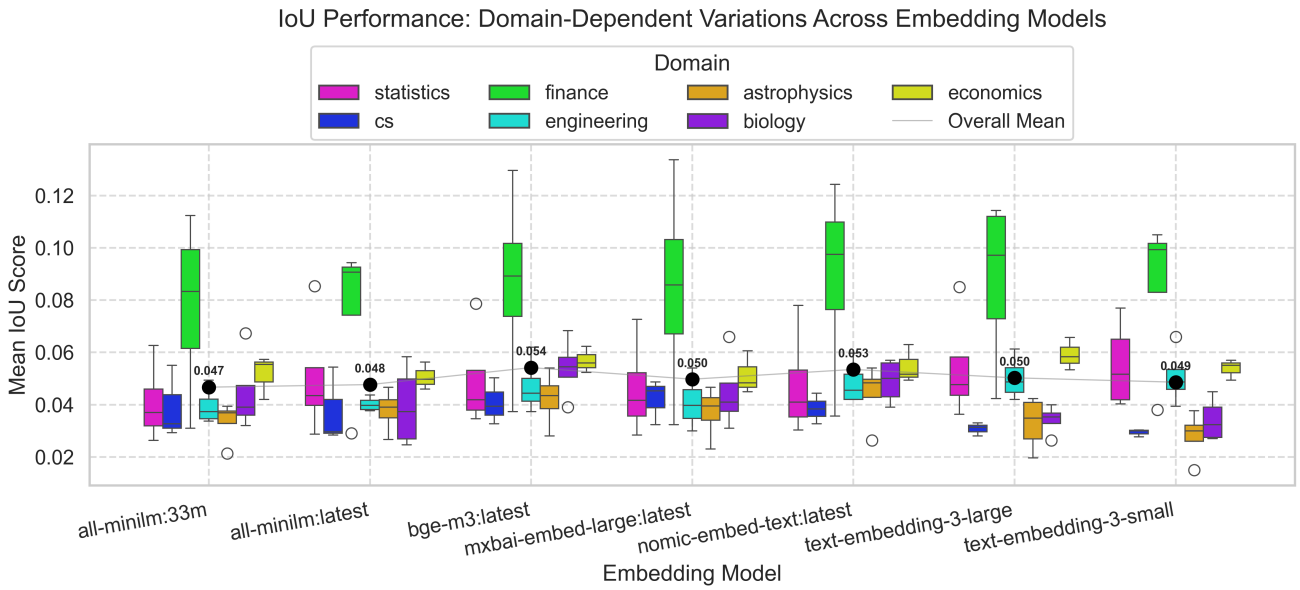

Fig. 2. Box plot distribution of IoU scores across embedding models and domains, highlighting significant domain-dependent performance variations. Finance consistently achieves the highest scores (0.06-0.11), while computer science and astrophysics show lower performance (0.03-0.04). *bge-m3* and *nomic-text-embed* models demonstrate best overall performance across domains.

TABLE II
EMBEDDING COST ON AN RTX 4090 (BATCH SIZE 32, FP16).

Embedding model	Avg. tokens / chunk	GPU sec / 1 000 chunks
all-minilm	485	**9.6**
text-embedding-3-small	486	12.3
all-minilm-33m	485	11.0
text-embedding-3-large	486	14.1
nomic-embed-text	512	16.8
mxbai-embed-large	957	18.5
bge-m3	512	21.7

embedding models, chunking strategies, and domain-specific performance. While the overall performance across embedding models and chunking strategies shows general consistency, we observed notable variations in specific domain-chunking strategy combinations. Notably, chunking parameters that perform well with a specific embedding model in one domain do not necessarily maintain this performance across other domains. A key finding is that larger embedding models with more parameters do not automatically translate to superior performance.

This is particularly evident in the comparison between commercial and open-source models. While OpenAI's models excel in average embedding time, they achieve lower mean IoU scores compared to smaller models, such as bge-m3 (567M parameters) and nomic-embed-text (1.5B parameters) (Figure 3). Our analysis reveals a slight negative correlation between model size and mean IoU scores (Figure 4). Domain-specific analysis shows that financial texts consistently achieve the highest performance metrics across all embedding models and chunking strategies, suggesting that financial documents may possess structural characteristics particularly well-suited to our semantic interoperability approach.

a) Embedding Models and Domains: Our analysis of IoU scores, as shown in Figure 3, reveals distinct patterns across domains and embedding models. Most notably, finance-related texts consistently achieve superior retrieval performance across all embedding models, with mean IoU scores significantly higher than those of other domains. However, this domain also exhibits the highest standard deviation ($\sigma = 0.032$), followed by statistics ($\sigma = 0.018$) and biology ($\sigma = 0.013$), indicating greater variability in retrieval quality. The *bge-m3* model demonstrates the best overall performance, surpassing both *nomic-embed-text* and OpenAI's *text-embedding-3-large* in mean IoU scores. Conversely, astrophysics consistently shows the lowest IoU scores across models, potentially due to limited representation in the models' training data. An observation emerges regarding model size and performance stability: smaller models exhibit more consistent performance across domains, as evidenced by their lower standard deviations (*all-minilm:latest* $\sigma = 0.021$, a*ll-minilm:33m* $\sigma = 0.021$) compared to larger models like *text-embedding-3-large* ($\sigma = 0.025$). This suggests that while larger models may offer superior performance in specific scenarios, they may be more sensitive to domain-specific variations.

b) Embedding Model and Chunking Strategy: The analysis shown in Figure 3 reveals optimal performance combinations between embedding models and chunking strategies. The *text-embedding-3-large* and *all-minilm* models achieve superior results when combined with *SentenceSplitter* (chunk_size = 512, overlap = 200). Similarly, *mxbai-embed-large* excels when using *TokenTextSplitter* (chunk_size = 1024, overlap

Fig. 3. Scatter plot of *mean* IoU score versus average embedding time for every chunker–embedder configuration. Marker **shapes** encode **SplitterType**, while colours identify the embedding model.The light-grey band shows the *interquartile range* (Q1 = 0.0450, median = 0.0495, Q3 = 0.0557; IQR = 0.0107) and the whiskers extend to the 95 % bootstrap confidence interval of the mean IoU. A paired t-test against the SentenceSplitter (512-75) baseline confirms that the three points inside the "good-and-fast" ellipse are significantly better at $\alpha = 0.05$. Smaller models such as *all-minilm* or the inference-optimised *text-embedding-3-small* therefore deliver competitive accuracy with much lower computational cost.

= 0), maintaining high IoU scores with efficient embedding times. The embedding time analysis reveals three distinct clusters corresponding to model sizes, which complements our earlier findings regarding model size and performance stability. While *bge-m3* and *mxbai-embed-large* demonstrate the highest average embedding times, these can be reduced by approximately 50% through NVIDIA H100 GPU acceleration on cloud platforms.

The top-performing chunking configurations achieved the following mean IoU scores:

$$\text{SentenceSplitter (512-200)} : IoU = 0.099$$
$$\text{TokenTextSplitter (1024-0)} : IoU = 0.093$$
$$\text{SentenceSplitter (512-200)} : IoU = 0.092$$

The experimental results presented highlight several key findings across our seven-domain evaluation. The combination of *text-embedding-3-small* with *SentenceSplitter* $(512 - 200)$ achieves the highest IoU score of 0.099, followed by *mxbai-embed-large* with *TokenTextSplitter* $(1024-0)$ at 0.093 and *all-minilm* with *SentenceSplitter* $(512-200)$ at 0.092. Notably, the latter two open-source models accomplish this performance with significantly smaller parameter counts of $335M$and $22M$ respectively.

The straightforward approach of sentence-based splitting consistently yields superior results across domains compared to more sophisticated methods, such as token-based or semantic chunking. This observation suggests that natural language boundaries provide more effective segmentation points for embedding-based retrieval tasks. Regarding chunk size configurations, our experiments indicate that moderate sizes $(256 - 512$ tokens) yield optimal results compared to smaller (128 tokens) or larger (1024 tokens) alternatives. This finding holds true across different model architectures and domains.

V. Conclusion and Future Work

We conducted the first large-scale benchmark that jointly varies chunking strategies and embedding models across seven scientific domains, totalling 2 520 retrieval runs. Results confirm that retrieval quality is not uniform across fields: finance PDFs consistently top the IoU charts, whereas astrophysics lags. A plausible explanation is *schema conformity*: financial reports feature regular tables and numeric lists that survive chunk boundaries intact, while free-form astrophysics prose scatters relevant facts, increasing the chance that any single chunk omits key tokens.

The study also nuances the "larger≠better" narrative. Smaller embedders (e.g. *all-minilm*) match or surpass billion-parameter models when queries are short (< 300 tokens), but recent long-query benchmarks (> 300 tokens) indicate the advantage can flip, with larger models regaining ground; future work should revisit this regime once public datasets mature.

Limitations. (1) The corpus is restricted to scientific texts, excluding legal or clinical material; (2) we measure only retrieval fidelity, not the downstream faithfulness of generated answers.

Future Work

- Correlate IoU improvements with answer-faithfulness scores from GPT-4o evaluations to close the retrieval-to-generation gap.
- Investigate how document structure (tables, figures, headings) alters optimal chunk sizing and overlap.
- Design adaptive chunkers that tune window and overlap on-the-fly according to domain cues.
- Extend the benchmark to multilingual corpora and non-academic genres.

Practical takeaway. Organisations are advised to *run a 30-min pilot with our open-source scripts* before committing to commercial retrieval APIs; such a trial typically embeds 50 k chunks on a single RTX 4090 and reveals whether a lightweight model already meets accuracy and latency targets.

REFERENCES

[1] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "MTEB: Massive Text Embedding Benchmark," Mar. 2023.

[2] B. Smith and A. Troynikov, "Evaluating chunking strategies for retrieval," Chroma, Tech. Rep., Jul. 2024.

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Apr. 2021.

[4] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG," Jan. 2025.

[5] M. Arslan, S. Munawar, and C. Cruz, "Business insights using RAG–LLMs: A review and case study," *Journal of Decision Systems*, vol. 0, no. 0, pp. 1–30, 2024.

[6] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Barcelona Spain: ACM, Aug. 2024, pp. 6491–6501.

[7] Y. Lyu, Z. Li, S. Niu, F. Xiong, B. Tang, W. Wang, H. Wu, H. Liu, T. Xu, and E. Chen, "CRUD-RAG: A Comprehensive Chinese Benchmark for Retrieval-Augmented Generation of Large Language Models," Jul. 2024.

[8] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui, "Retrieval-Augmented Generation for AI-Generated Content: A Survey," Jun. 2024.

[9] R. Qu, R. Tu, and F. Bao, "Is Semantic Chunking Worth the Computational Cost?" Oct. 2024.

[10] D. Danter, H. Mühle, and A. Stöckl, "Advanced Chunking and Search Methods for Improved Retrieval-Augmented Generation (RAG) System Performance in E-Learning: AHFE International Conference on Human Factors in Design, Engineering, and Computing," Dec. 2024, pp. 1889–1898.

[11] W. Wang, S. Zhang, Y. Ren, Y. Duan, T. Li, S. Liu, M. Hu, Z. Chen, K. Zhang, L. Lu, X. Zhu, P. Luo, Y. Qiao, J. Dai, W. Shao, and W. Wang, "Needle In A Multimodal Haystack," Oct. 2024.

[12] R. Zhao, H. Chen, W. Wang, F. Jiao, X. L. Do, C. Qin, B. Ding, X. Guo, M. Li, X. Li, and S. Joty, "Retrieving Multimodal Information for Augmented Generation: A Survey," Dec. 2023.

[13] A. J. Yepes, Y. You, J. Milczek, S. Laverde, and R. Li, "Financial Report Chunking for Effective Retrieval Augmented Generation," Mar. 2024.

[14] I. Hartsock and G. Rasool, "Vision-Language Models for Medical Report Generation and Visual Question Answering: A Review," *Frontiers in Artificial Intelligence*, vol. 7, p. 1430984, Nov. 2024.

[15] K. Li, B. Yu, Q. Zheng, Y. Zhan, Y. Zhang, T. Zhang, Y. Yang, Y. Chen, L. Sun, Q. Cao, L. Shen, L. Li, D. Tao, and X. He, "MuEP: A Multimodal Benchmark for Embodied Planning with Foundation Models," in *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*. Jeju, South Korea: International Joint Conferences on Artificial Intelligence Organization, Aug. 2024, pp. 129–138.