Optimal Chunking Strategies for Document Embeddings: A Focused Evaluation [E]

Sagar Abhyankar, Aditya Kulkarni, Adish Jain, TKS
{sra9,aak14,adishj2,at71}@illinois.edu
University of Illinois Urbana-Champaign, Urbana, Illinois, USA

Abstract

Chunking strategies are fundamental for preparing documents for embedding-based retrieval in modern RAG (retrieval-augmented generation) systems. Despite their widespread use, there is limited systematic evaluation of how different chunking strategies affect embedding quality and downstream retrieval performance across datasets and models. In this paper, we compare fixed-size, semantic, and sentence-based chunking methods, evaluating their impact on retrieval accuracy and efficiency across leading embedding models and benchmark datasets with varying question-context difficulty. Our results provide actionable guidance for practitioners seeking optimal chunking strategies for RAG workflows.

1 Introduction

Vector embeddings are foundational to modern information retrieval and retrieval-augmented generation (RAG) systems [3], enabling semantic search and context-aware generation. However, most real-world documents far exceed the context window of leading embedding models [4], necessitating that they be split into smaller *chunks* before embedding and indexing. The choice of chunking strategy-how a document is divided-can significantly impact both the quality of the embeddings and the downstream performance of RAG systems. Common chunking strategies include fixed-size chunking (with or without overlap), semantic chunking (using embedding similarity or large language models to find breakpoints), and sentence-based chunking. Despite their widespread adoption in industry and academia, there is still no comprehensive, systematic evaluation comparing these chunking strategies across models and tasks [5]. As a result, practitioners lack clear guidance on optimal chunking for retrieval and RAG performance. Recent work by Pinecone [1] and Chroma [2] has begun to explore these strategies, but robust, multi-model, multi-metric evaluations remain rare. The intuition behind why chunking strategy would matter is that splitting a document inevitably risks losing some of the semantic coherence that binds each chunk to their broader context. Poorly chosen chunking strategies can lead to information fragmentation, reducing retrieval accuracy and degrading the quality of generated answers in RAG workflows. Thus, it is crucial to evaluate which strategies best preserve context while remaining computationally efficient. In this work, we systematically evaluate three major chunking strategies-fixed-size (with various overlaps), semantic, and sentence-based-across multiple top-performing embedding models (as ranked on the MTEB leaderboard [9]) and benchmark datasets (SQuAD [20], NQ [22], QuALITY [21]). All datasets used consist of multiple-choice questions with a single correct answer, allowing us to precisely gauge RAG performance in a way that closely mirrors real-world applications, rather than relying solely on statistical metrics like semantic chunk coherence. We measure the impact of each chunking strategy on retrieval accuracy and efficiency, and analyze how these effects vary by dataset and model. Our findings provide actionable insights for practitioners seeking to optimize chunking in RAG pipelines.

1.1 Chunking Strategies Overview

We briefly introduce the chunking strategies considered in this study and illustrate their operation in Figure 1.

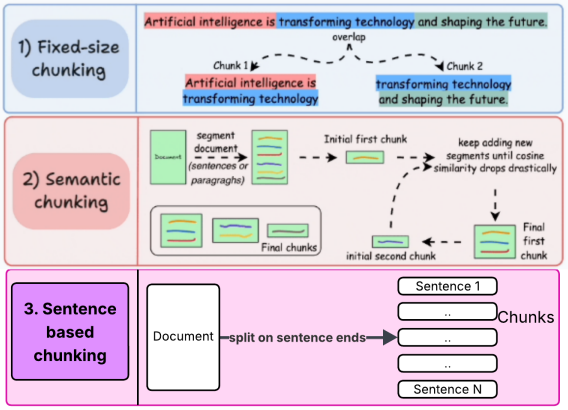

Figure 1: Overview of the chunking workflow for each strategy. Part taken from [27]

1.1.1 Fixed-Size Chunking. The document is divided into consecutive chunks of a fixed length (e.g., by tokens or characters). Overlap between chunks can be introduced to ensure that information near chunk boundaries is not lost. The percentage of overlap can be varied. This method is simple and efficient, but does not account for the underlying semantic structure of the text.

1.1.2 Semantic Chunking. The document is first segmented into sentences or paragraphs. Embeddings are computed for each segment, and these are grouped together into chunks such that each chunk maintains high semantic similarity. When the similarity between neighboring segments drops below a threshold, a new chunk is started. This approach aims to preserve topic coherence within each chunk and adaptively finds breakpoints where the meaning shifts.

1.1.3 Sentence-Based Chunking. The document is split strictly at sentence boundaries, with each sentence (or group of sentences) forming a chunk. This method is straightforward and ensures that

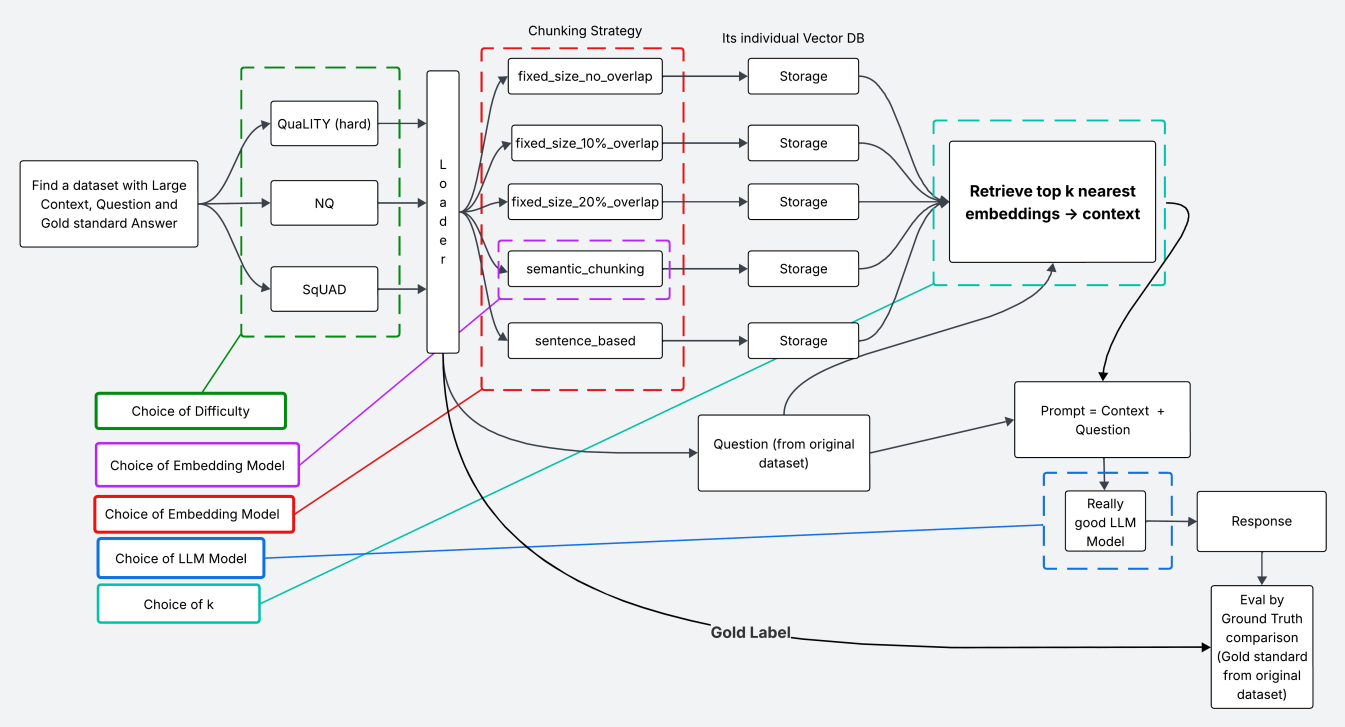

Figure 2: Evaluation Approach.

chunks do not cut across sentences, but may result in many small chunks and can miss broader context that spans multiple sentences.

2 Related Work

Chunking strategies have been explored in both industry implementations and academic research. [6]. Industry guides from Pinecone [1] and Chroma [2] emphasize the importance of chunking for embedding quality, proposing semantic and overlap-based methods. However, these resources primarily offer heuristic recommendations without rigorous empirical validation or cross-dataset comparisons. Recent academic work has begun quantitatively evaluating chunking strategies [7]. The LangChain Semantic Chunking Arena study [8] compared percentile-, gradient-, and deviation-based methods across five domains (ML, healthcare, history, law, and e-commerce), finding that optimal strategies vary significantly by domain (F1-score differences of 12–18%). Chroma Research [2] introduced token recall metrics for RAG-oriented evaluation, demonstrating that semantic chunking improves relevant token retrieval by 22% over fixed-size approaches on legal documents, though at 3× higher computational cost. For embedding models, the MTEB leaderboard [9] provides standardized benchmarks across 56 datasets, but excludes chunking strategy as a variable. IBM's Granite tutorial [10] showed that 300–500 token chunks with 10% overlap optimize RAG performance for clinical texts when using domain-specific embeddings. However, their findings were limited to healthcare data and a single embedding family. Our work extends these foundations with three key innovations: 1) systematic comparison of chunking strategies across multiple MTEB-top embedding models (BGE, E5, MiniLM), 2) evaluation on diverse benchmark datasets (SQuAD, NQ, QuALITY) with controlled query types, and 3) quantitative analysis of the accuracy/efficiency tradeoffs inherent to semantic vs. fixed-size chunking.

3 Proposed Method

As seen in the Figure 2 we setup a standard RAG pipeline for the evaluation. We selected pre-existing benchmark datasets with varying difficulty. We standardized all datasets into a 4-option single-correct-answer multiple-choice format.

- **QuALITY [21]**: This is one of the hardest and modern benchmark dataset which needs deep contextual reasoning to answer the question accurately.
- **SQuAD [20]**: This dataset is an easier dataset to answer, mostly contains factoid based/straightforward questions which are answered in the context.
- **NQ [22]**: This dataset again contains Wikipedia context and questions which don't have a direct word to word answer in the context and requires a basic understanding of the context. One unique aspect about this dataset is that the context text isn't clean and contains a lot of tags from the html scraping which confuses some of the chunking strategies as explained ahead.

We implement a common dataset loader template to eliminate any biases from the way the questions and gold labels are tracked, this also makes the datasets hotswappable and experimenting much faster. For the fixed size chunking strategy we tested three variants,

with 0% overlap, 10% overlap and 20% overlap. All strategies processed identical context-question-answer pairs, with embeddings stored in separate FAISS [23] vector databases per strategy. When chunking and embedding for each strategy, we record different metrics such as the processing time, average chunk size, average number of chunks per document, variance of chunk size to compare the behaviour of different chunking strategies.During retrieval, each of strategy could access only its own vector embeddings. We retrieved top k (k=5 across all the strategies) chunks from the vector db and supplied those as the context to the LLM along with the question. The LLM was prompt-engineered to output only the letter corresponding to the correct answer (e.g., 'C'). We then match if the LLM's output matches the gold label option for that question, accuracy is 1 if matches else 0. The LLM used was LLAMA 8B instruct available on huggingface [19]. We also include a no _context baseline to prove that RAG is actually enhancing the performance using chunking strategy. The source code for this project is open and available here [25]. Now as seen in the Figure 2 we also discuss different choices which we think might impact the overall accuracy of the strategies.

3.0.1 ***Choice of Difficulty.*** As pointed out by Pinecone [1], a strategy may change its behaviour based on the quality of the text in the context. To avoid any bias we choose different types of datasets as discussed above.

3.0.2 ***Choice of Semantic Embedding Model.*** This choice will affect only the accuracy of semantic chunking. To give semantic chunking the best shot at the evaluation we test different embedding models such as all-MiniLM-L6-v2 [12], Alibaba-NLP/gte-large-en-v1.5 [13] and Snowflake/snowflake-arctic-embed-l-v2.0 [14]. We also implement chunking using both LangChain and LlamaIndex [24], allowing us to compare their semantic chunkers.

3.0.3 ***Choice of Embedding Model.*** We tested the following embedding models multilingual-e5-large-instruct [15], all-MiniLM-L6-v2 [12], bge-m3 [16], bge-large-en-v1.5 [17], snowflake-arctic-embed-l v2.0 [14], intfloat/e5-large [18].

3.0.4 ***Choice of LLM Model.*** We only test meta-llama/Llama-3.1-8B-Instruct [19] as we think a better model than this will only increase the accuracy across all the model, but our evaluation focuses on the comparison between strategies.

3.0.5 ***Choice of k.*** The number of chunks we retreive will essentially affect the context size we load, we use the value of 5 which should be enough for all strategies to gather the required context to answer the question. But we believe this parameter can be individually optimized for each strategy individually to give it its best shot. For embedding, we use top models from the MTEB leaderboard, including BGE-large-en-v1.5, MiniLM-L6-v2, and intfloat/e5-large-v2 [18].

4 Results

We ran the above described evaluation pipeline across variety of models, and while its impractical to show all the graphs for each run here, we have kept all the results available here [26]. We discuss the common trends we see across all the experiments instead of talking for each model and dataset.

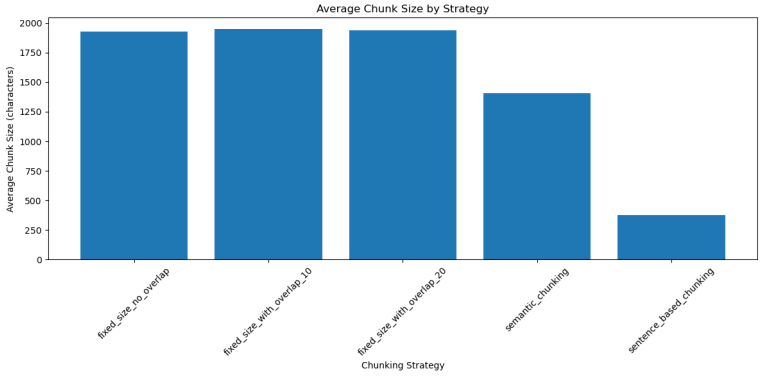

Figure 3: Average Chunk Sizes.

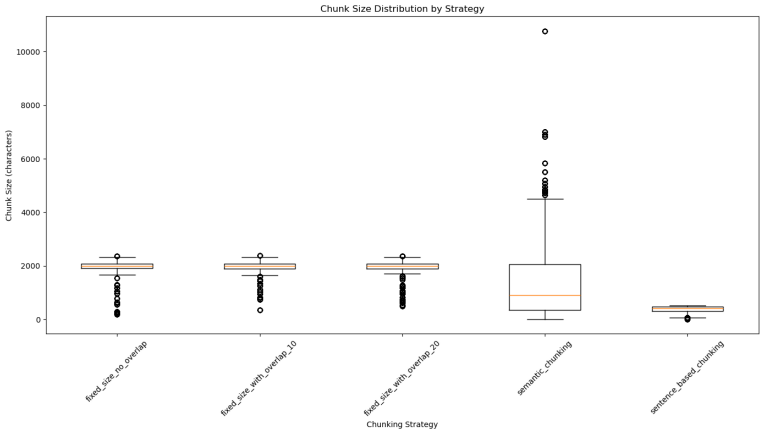

Figure 4: Chunk Size Distribution.

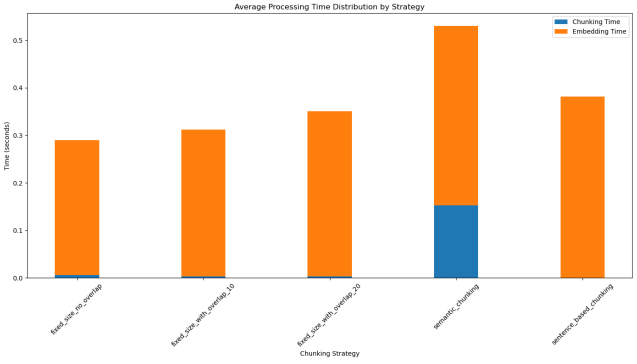

Figure 5: Average Processing Time.

4.1 Chunking and Embedding Metrics

- **Average chunk size:** As seen from Figure 3 and Figure 4, Fixed-size chunking yields consistent sizes; semantic chunking produces slightly smaller but higher variance chunks; sentence-based chunking results in the smallest and most numerous chunks.

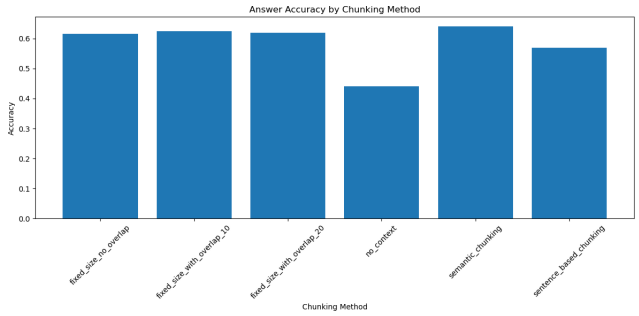

Figure 6: Accuracy on QuALITY.

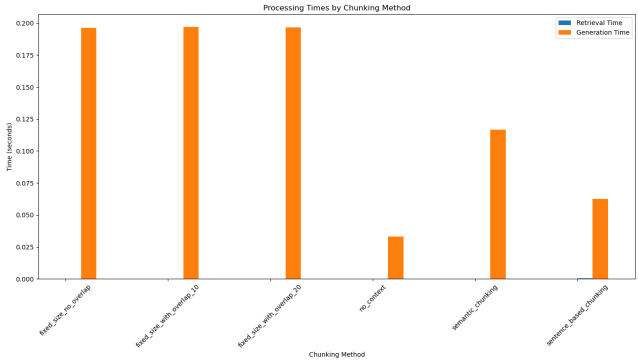

Figure 7: LLM Generation Time.

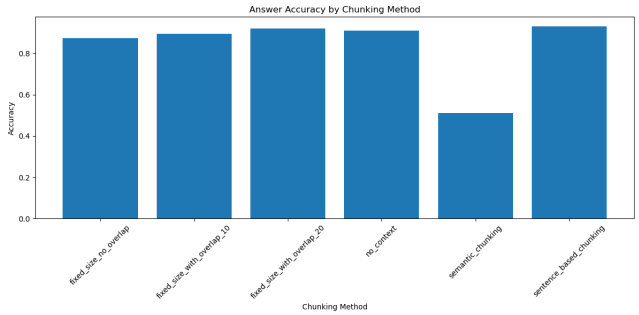

Figure 8: Accuracy on NQ.

- **Chunking time:** For the following items refer Figure 5. Semantic chunking is ∼20× slower than fixed-size; sentence-based is fastest.
- **Embedding time:** Scales with number of chunks and their size; generally highest for semantic chunking.
- **Total processing time:** Highest for semantic chunking, lowest for sentence-based.

4.2 Retrieval and RAG Performance

- **Accuracy:** As seen in Figure 6 Fixed-size with 20% overlap is consistently top-2, sometimes the best. Semantic chunking only outperforms when optimally tuned, otherwise it can vastly under perform compared to other strategies as seen in Figure 8. We observed that Semantic chunking underperforms when text contains noise like HTML tags or non-standard characters. Sentence-based chunking is generally 2–3% behind fixed sized strategies on the accuracy scale.
- **RAG improvement and Processing time:** RAG boosts accuracy by 10–20% over no-context baseline, especially on harder datasets. LLM answer generation time scales with context size, which itself depends on chunk size (Figure 7)
- **Dataset trends:** SQuAD is easy (all strategies/models perform well); NQ baseline is strong (87%) (the LLAMA model seems to have trained on the data used for the context); QuALITY is hardest (no-context baseline 44%).
- **Model effects:** The embedding model performance was exactly reflective of its rank on the MTEB leaderboard, where a better rank improved accruacy across all the strategies. Though for semantic chunking specifically, the model MiniLM-L6-v2 outperforms others on NQ/QuALITY; Alibaba GTE-large under performs.
- **Semantic Chunking Tuning:** Semantic chunking performance depends significantly on implementation and parameter tuning. LlamaIndex's semantic chunker [11] outperformed LangChain's due to superior breakpoint detection. Two critical parameters affected performance: buffer size (controlling context consideration) performed better at lower values for RAG, while breakpoint threshold required dataset-specific tuning-lower thresholds for high topic variation datasets like NQ, and higher thresholds (90+) for stable topical structure in SQuAD and QuALITY. This parameter sensitivity explains semantic chunking's inconsistent performance across evaluations, as it excelled only when carefully optimized for specific content types.

4.3 Conclusion

Our evaluation reveals that there is no one-size-fits-all chunking strategy for RAG systems. Fixed-size chunking with overlap consistently delivers robust performance across datasets and embedding models, making it a reliable default choice. While semantic chunking achieves superior results when carefully tuned for specific datasets and paired with compatible embedding models, this optimization demands substantial expertise and computational resources. LlamaIndex's semantic chunker proves particularly valuable for content with high topic variation, outperforming other implementations. While increasing overlap in fixed-size chunking reliably improves retrieval accuracy, this comes at the cost of increased processing time and storage requirements. Practitioners should select chunking strategies based on their specific performance requirements, computational constraints, and dataset characteristics.

References

[1] Pinecone. *Chunking Strategies for LLM Applications*. 2025. [Online]. Available: https://www.pinecone.io/learn/chunking-strategies/

[2] Chroma Research. *Evaluating Chunking Strategies for Retrieval*. 2024. [Online]. Available: https://research.trychroma.com/evaluating-chunking

[3] Elastic. *What are Vector Embeddings?*. 2024. [Online]. Available: https://www.elastic.co/what-is/vector-embedding

[4] NVIDIA. *tutorial-chatgpt-over-your-data*. 2023. [Online]. Available: https://blog.langchain.dev/tutorial-chatgpt-over-your-data/

[5] Kopp Online Marketing. *Evaluation of Retrieval-Augmented Generation: A Survey*. 2024. [Online]. Available: https://www.kopp-online-marketing.com/patents-papers/evaluation-of-retrieval-augmented-generation-a-survey

[6] J. Doe et al. *S2 Chunking: A Hybrid Framework for Document Segmentation Through Integrated Spatial and Semantic Analysis*. arXiv, 2025. [Online]. Available: https://arxiv.org/html/2501.05485v1

[7] Superlinked. *An evaluation of RAG Retrieval Chunking Methods*. 2024. [Online]. Available: https://superlinked.com/vectorhub/articles/evaluation-rag-retrieval-chunking-methods

[8] M. Nguyen et al. *Benchmarking LangChain Semantic Chunking Methods*. GitHub, 2024. [Online]. Available: https://github.com/monami44/Langchain-Semantic-Chunking-Arena

[9] Hugging Face. *MTEB Leaderboard*. 2025. [Online]. Available: https://huggingface.co/spaces/mteb/leaderboard

[10] IBM. *Chunking Strategies for RAG with Granite*. 2025. [Online]. Available: https://www.ibm.com/think/tutorials/chunking-strategies-for-rag-with-langchain-watsonx-ai

[11] LlamaIndex. *Semantic Chunker Documentation*. 2024. [Online]. Available: https://docs.llamaindex.ai/en/stable/examples/node_parsers/semantic_chunking/

[12] Sentence-Transformers. *all-MiniLM-L6-v2 Model Card*. 2024. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[13] Alibaba-NLP. *gte-large-en-v1.5 Model Card*. 2024. [Online]. Available: https://www.aimodels.fyi/creators/huggingFace/Alibaba-NLP

[14] Snowflake. *snowflake-arctic-embed-l-v2.0 Model Card*. 2024. [Online]. Available: https://huggingface.co/Snowflake/snowflake-arctic-embed-l-v2.0

[15] IntFloat. *multilingual-e5-large-instruct Model Card*. 2025. [Online]. Available: https://huggingface.co/intfloat/multilingual-e5-large-instruct

[16] BAAI. *BGE-M3 Model Documentation*. 2024. [Online]. Available: https://bge-model.com/bge/bge_m3.html

[17] BAAI. *bge-large-en-v1.5 Model Card*. 2024. [Online]. Available: https://dataloop.ai/library/model/baai_bge-large-en-v15/

[18] IntFloat. *e5-large-v2 Model Card*. 2024. [Online]. Available: https://huggingface.co/intfloat/e5-large-v2

[19] Meta AI. *Meta-Llama-3-8B-Instruct Model Card*. 2024. [Online]. Available: https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[20] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016. [Online]. Available: https://aclanthology.org/D16-1264/

[21] A. Rogers, et al. *QuALITY: Question Answering with Long Input Texts, Yes!* 2022. [Online]. Available: https://arxiv.org/abs/2112.08608

[22] Natural Questions Dataset. *Papers With Code*. 2024. [Online]. Available: https://paperswithcode.com/dataset/natural-questions

[23] Meta AI. *FAISS: A library for efficient similarity search and clustering of dense vectors*. 2024. [Online]. Available: https://github.com/facebookresearch/faiss

[24] LlamaIndex. *Basic Chunking Strategies Documentation*. 2024. [Online]. Available: https://docs.llamaindex.ai/en/stable/optimizing/basic_strategies/basic_strategies/

[25] Implementation Source Code. *Source code for Pipeline Used in this Evaluation* [Online]. Available https://github.com/Ingenious-c0der/CS598-BYOMP

[26] Evaluation Results. *Evaluation Outputs for each of the models tested for this study*[Online]. Available https://drive.google.com/drive/folders/1nTULCFyU7Uzdtu1SklQ_o5M_N3oBD436?usp=sharing

[27] Blog on 5 chunking strategies. [Online]. Available https://blog.dailydoseofds.com/p/5-chunking-strategies-for-rag