Advanced Chunking Techniques: a Novel Approach for Semantic Splitters

Bernardo Ramos Toresan[1], **Viviane Pereira Moreira**[1], **Felipe Soares Fagundes Paula**[1], **Luciana Regina Bencke**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{bernardo.toresan, viviane, fsfpaula, luciana.bencke}@inf.ufrgs.br,

Abstract. *Chunking, the process of splitting large amounts of text into processable parts, is an essential but often overlooked step for multiple Information Retrieval and Vector Databases tasks. Traditional chunking techniques rely on fixed-length or syntactic structures, creating opportunities for more meaningful approaches. Semantic chunking is the process of dividing text based on meaning and context, ensuring each chunk represents a logical unit of information. This work proposes the Dual Semantic Chunker, which represents an advancement over existing chunking methods by taking a closer look at semantic representation. We compared multiple chunking methods, including both semantic and traditional techniques, and achieved improved retrieval.*

1. Introduction

As the volume of information continues to grow (Hilbert and López, 2011), retrieving relevant data from vast and often unstructured text sources has become an increasingly complex challenge. To handle these rapidly growing databases effectively, it is crucial to segment information into manageable and meaningful chunks (Koshorek et al., 2018).

Recent advances in Natural Language Processing (NLP), particularly the emergence of Large Language Models (LLMs), have introduced powerful new mechanisms for structuring and indexing unstructured text data within database systems (Duarte et al., 2024). Transformer-based models (Vaswani, 2017) enable the extraction of dense semantic embeddings, which can be stored and queried efficiently using vector databases and similarity search frameworks. These semantic chunking techniques facilitate improved support for approximate nearest neighbor (ANN) search, scalable indexing, and hybrid retrieval pipelines, thereby bridging the gap between traditional structured data management and modern unstructured information retrieval.

Text chunking techniques face several challenges. Unstructured data convolutes the identification of meaningful segments due to the lack of clear boundaries (Gharehchopogh and Khalifelu, 2011). Evaluating text chunking is also challenging because it lacks a clear gold standard, varies by task, and its success is measured indirectly through its impact on downstream tasks such as retrieval or summarization. Additionally, context sensitivity is essential (Porzel and Gurevych, 2003), as the meaning of sentences may depend on surrounding information.

Despite its potential to improve system performance, text chunking has not been thoroughly investigated in the literature. Current solutions primarily rely on traditional

methods such as fixed-length chunking (P. Lewis et al., 2020; Izacard and Grave, 2021; Karpukhin et al., 2020) and syntactic-based chunking (Yang et al., 2019; Lee et al., 2021), which segment text based on markers such as punctuation. Though computationally inexpensive, these approaches fail to capture the semantic structure of the given corpus. This results in chunks that are not only visually unappealing but also may negatively impact retrieval performance.

We propose the Dual Semantic Chunker (DSC), a novel approach that relies on semantic representations to decide on splitting points that better preserve the inherent meaning of text. DSC was designed to improve retrieval quality by effectively extracting and utilizing semantic data from a corpus. DSC employs a two-step approach. First, the text is split into blocks. Then, the semantic representations computed for the blocks are leveraged to decide on optimal splitting points. DSC creates chunks that better represent the text into meaningfully complete units that improve retrieval.

We compared DSC to both traditional and novel methods in a diverse dataset composed of documents from several domains. We analyzed the chunkers by looking at the ranking of chunks retrieved using a nearest neighbor retriever. The results showed that DSC outperformed existing approaches in terms of F1, DCG, and Precision. This was done while preserving the flexibility to control chunk length for different retrieval needs.

2. Background and Related Work

Recently, novel approaches to text segmentation have emerged, surpassing previous methods. Traditional methods rely on syntactic structures such as token count, punctuation, and white spaces. Semantic algorithms are more sophisticated and use NLP advancements to use embedding representations to decide segmentation points in the corpus. Language model chunkers aim to take full advantage of LLMs to segment text, using query results to determine splitting points. The most commonly used chunking strategies are described below.

Fixed Token Chunker is the most straightforward approach. It relies on using a predefined chunk length to divide the text into same-length parts, often with overlapping segments. Given its simplicity, it is widely adopted (P. Lewis et al., 2020; Izacard and Grave, 2021; Karpukhin et al., 2020).

Recursive Chunking relies on syntactic structures, such as punctuation, to partition the text. This is a low-cost method that, on a well-structured text, splits it into sentences or paragraphs. Recursively splitting text serves the purpose of trying to keep related pieces of text next to each other (Yang et al., 2019; Lee et al., 2021).

Kamradt Semantic Chunking (Kamradt, 2024) uses dense embeddings to group semantically similar sentences together. The method first pre-separates the text into sentences using recursive chunking. Then, a vector representation of each of these spans is extracted using a sliding window of sentences. With this arranged list of embeddings, it iterates through the collection, comparing neighboring windows to decide on splitting points.

Cluster Semantic Chunking Smith and Troynikov, 2024 is an alteration of the previous method. To maximize similarity within chunks, documents were split into blocks using the recursive splitting algorithm. Batches containing these blocks are passed to an

embedding model to achieve contextual token semantic representation. At the end of each batch process, the algorithm globally re-computes the chunks, aiming to maximize pairwise cosine similarity between all pairs of tokens within any chunk.

Proposition Language Model Chunking (T. Chen et al., 2024) generated chunks at a fine granularity called *propositions*. Each proposition is defined as an atomic expression within the text and aims to encapsulate a distinct fact. A fine-tuned text generation model for this specific task receives a passage and generates the propositions within the passage.

Lumber Language Model Chunking (Duarte et al., 2024) aims to take advantage of the capabilities of language models to analyze text to generate chunks that are as independent from one another as possible. The text is split into paragraphs and given to an LLM that is prompted to identify the first paragraph where the content clearly changes compared to previous paragraphs. The preceding paragraphs are concatenated and finalized as a chunk. This process is repeated until all the paragraphs are inside a chunk.

Table 1. Chunking Methods Intuition

Chunking Method	Intuition	Chunked Output
Fixed Token	Split by fixed token count (e.g., 7 tokens)	1. Tell me, O Muse, of that ingenious 2. hero who travelled far and wide after 3. he had sacked the famous town of Troy.
Recursive	Use punctuation (. , ! ?) to split	1. Tell me, 2. O Muse, 3. of that ingenious hero who travelled far and wide after he had sacked the famous town of Troy.
Kamradt Semantic	Group semantically related *sentences* using embeddings.	1. Tell me, O Muse, of that ingenious hero 2. who travelled far and wide 3. after he had sacked the famous town of Troy.
Cluster Semantic	Form clusters maximizing semantic similarity	1. Tell me, O Muse, of that ingenious hero who travelled far and wide 2. after he had sacked the famous town of Troy.
Proposition LM	Split into minimal atomic facts using LM	1. There is an ingenious hero. 2. He travelled far and wide. 3. He sacked the famous town of Troy.
Lumber LM	Asks an LLM to detect meaningful paragraph shifts	1. Tell me, O Muse, of that ingenious hero who travelled far and wide. 2. After he had sacked the famous town of Troy.

We present in Table 1 showing the intuition behind each chunking technique by showing how we would expect them to behave.

While these methods offer various strategies for text segmentation, they still present limitations that impact their effectiveness. Fixed token chunking enforces rigid segment lengths, often disregarding semantic continuity, while recursive chunking depends on syntactic cues that may not align with semantic boundaries. Semantic chunking approaches improve upon these by leveraging embeddings, yet they typically rely on pre-segmentation strategies like recursive chunking, which can be enhanced. Additionally, existing methods provide limited control over chunk length without compromising the semantic integrity of the chunks. This gap highlights the need for approaches that offer more flexible chunk length control while preserving semantic coherence, as well as more sophisticated initial segmentation strategies that enhance the quality of chunking decisions.

3. Dual Semantic Chunker

This section introduces our proposed semantic chunker, the Dual Semantic Chunker (DSC). The process begins by segmenting the text into smaller, semantically meaningful blocks. Each block refers to a single sentence or fragment of sentence that is then encoded to generate an embedding vector. These blocks are then analyzed and grouped based on their similarity, leading to the formation of chunks that preserve the integrity and coherence of the original text. By combining linguistically informed block generation with a statistical thresholding method, our approach produces coherent chunks that improve downstream retrieval performance. The process is depicted in Figure 1, and each step is detailed in the following sections. Our code is freely available at `https://github.com/btoresan/Advanced-chunking-Techniques.git`

3.1. Extracting Semantic Data

In pursuit of obtaining embeddings that accurately capture semantic similarity and contextual relevance, semantic chunkers often use strategies such as sentence level embeddings Smith and Troynikov, 2024 or context windows Kamradt, 2024 to initially partition the text into more manageable blocks. In our approach, we aimed to obtain blocks that serve as a starting point of coherent portions of text that should be kept in the same chunk and then combine these blocks chunks respecting the order in which they appear in the text and the length parameters.

Chunking procedures that rely on sentence embeddings, like ours, commonly initially segment text into blocks via simpler algorithms akin to recursive chunking. DSC

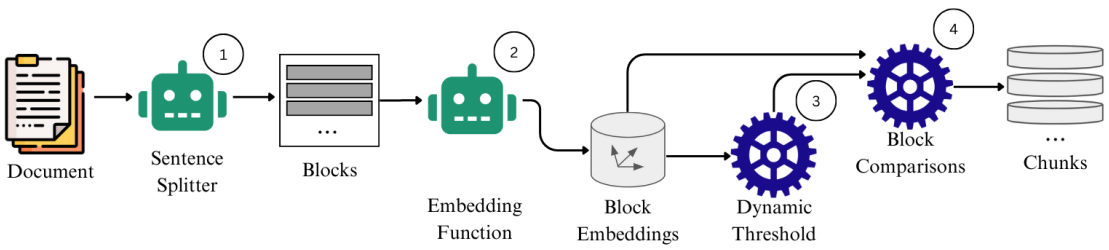

Figure 1. Overview of the Dual Semantic Chunker

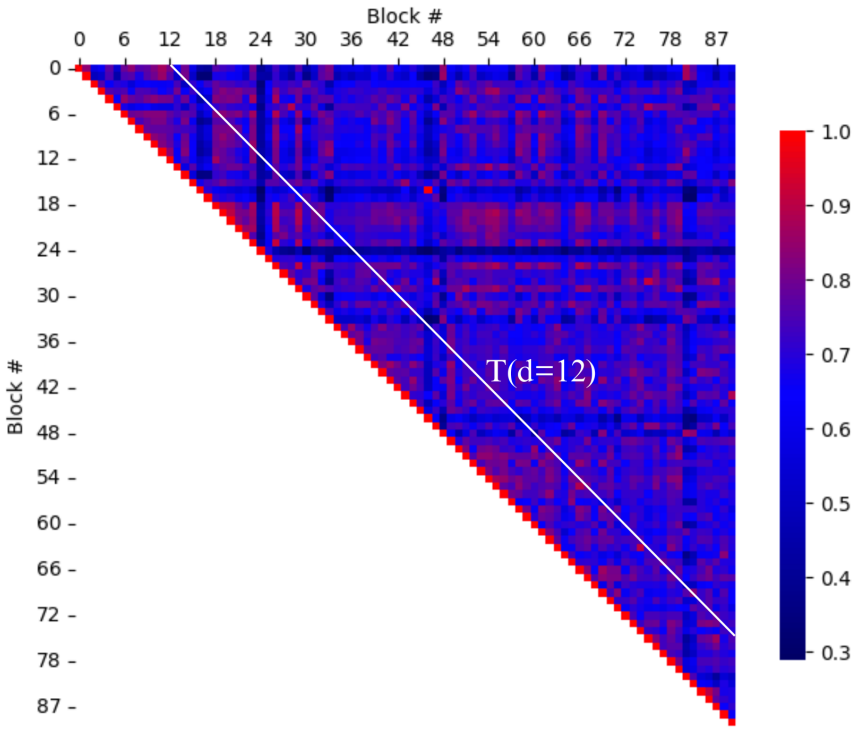

Figure 2. Example of a similarity matrix

aims to achieve more refined blocks by splitting the text with pre-trained linguistic models from spacy[1] (Step 1 in Fig 1). Finally, in Step 2, with the aim of obtaining semantically cohesive chunks, we aggregate subsequent blocks based on their similarity. For that, DSC leverages a Sentence Transformer model (Reimers and Gurevych, 2019) to generate high-quality sentence embeddings, providing fixed-length representations for each block. This results in a sequence of ordered text blocks, each paired with a contextually meaningful embedding.

3.2. Statistical Threshold and Similarity Matrix

Semantic splitters traditionally check if the similarity between two blocks is below a threshold to decide on splitting points (Step 3 in Fig. 1). Due to the varying nature of similarity distributions within texts, we explored statistical methods to define a threshold.

We build a similarity matrix S where each element S_{ij} corresponds to the cosine between blocks i and j and is calculated as $S_{ij} = \frac{(VV^\top)_{ij}}{\|\mathbf{v}_i\|\|\mathbf{v}_j\|}$, where V is the matrix of block embeddings. The matrix is used to calculate the threshold that defines when to stop aggregating blocks into a chunk. The main diagonal of S is composed of 1's and the elements close to the diagonal represent similarities between blocks that are nearby in the text.

Given the similarity matrix, we define a population of similarity values to apply statistical methods and determine a threshold that effectively distinguishes meaningful segment boundaries. To ensure the population is representative, we include only elements within a specified distance, d, from the main diagonal of S, focusing on pairs of blocks

[1]https://spacy.io/

that are closer together in the text and thus more likely to belong to the same chunk. The ability to adjust d allows for control over the population of sentence similarities based on text structure. Ideally, d should be chosen such that the resulting population includes only sentences close enough to one another to reasonably have a chance to be placed in the same chunk. The population of target elements, T, is then calculated as in Eq. 1, where d defines the range of relevant similarities, and N represents the total number of blocks.

$$T = \left\{ S_{i,j} \;\middle|\; i < j < \min(i + d, N - 1) \right\} \tag{1}$$

Finally, the threshold θ given a population T is its mean μ_T plus its standard deviation $\sqrt{\sigma_T}$ multiplied by the parameter k, as in Eq. 2.The parameter k controls how strict the threshold is: higher values of k make the threshold more selective by requiring greater similarity to be considered semantically relevant. This approach ensures that the threshold adapts to the similarity distribution, providing a limit that distinguishes between blocks that are semantically similar and those that are not and, therefore, should be in a separate chunk.

$$\theta = \mu_T + k\sqrt{\sigma_T} \tag{2}$$

3.3. Comparing Semantic Blocks

Given the similarity matrix S and the threshold θ, Step 4 Fig. 1 decides how to group blocks into chunks. Throughout this process, as a means to respect the source text integrity, blocks are iterated through in the same order as they appear in the text. Each chunk (C_k) is handled individually, appending blocks following the flowchart in Fig 3 until all blocks are inside a chunk. DSC also receives parameters for chunk granularity (minimum and maximum tokens for each chunk) that are taken into consideration. The first step is to append the current block (b_i) to C_k if C_k is empty or smaller than the minimum chunk length. If appending b_i to C_k makes it longer than the maximum chunk length, C_k is finalized and the block is not added. Finally, for all blocks b_j in the chunk, if their similarity with the current block b_i is greater than the statistical threshold (Eq. 3), then b_i is added to the chunk. Otherwise, the chunk is finalized.

$$C_k \leftarrow C_k \cup \{b_i\} \quad \text{if } \forall b_j \in C_k,\ \mathrm{S}_{b_i,b_j} > \theta \tag{3}$$

4. Experimental Setup

To evaluate DSC performance, we compared it with a number of baselines in an evaluation experiment described next. The chunks are stored in a vector-oriented database, where they are retrieved in response to queries via K-Nearest Neighbor (**bhatia2010survey**) search.

Dataset. The dataset consists of 328,208 token-long plain text source documents and 472 queries. The plain text is composed of five corpora from diverse domains and a mix of both clean and messy text sources, including State of the Union Address 2024 (The White House, 2024)(10,444 tokens long), Wikitext (Merity et al., 2022) (26.649 tokens), Chatlogs (Ding et al., 2023) (7,727 tokens), Finance (Z. Chen et al., 2022) (166,177 tokens), and PubMed (National Library of Medicine, 2023) (117,211 tokens). Gold standard passages are provided with the dataset and contain the relevant passages for each

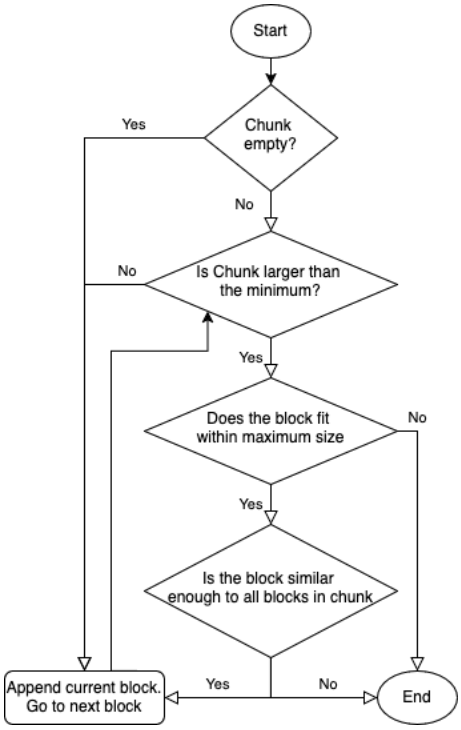

Figure 3. Grouping blocks in a chunk

query. Relevant passages are annotated at the token level and relevance is binary (the tokens are considered as relevant or not relevant). Factual queries and corresponding passages were synthetically generated in Chroma's evaluation framework (Smith and Troynikov, 2024) by leveraging an LLM, which was prompted with corpus documents and up to 50 prior queries to ensure uniqueness. Only excerpts with exact matches in the corpus were retained, and any query associated with an invalid excerpt was discarded.

Evaluation Framework. Each chunking function segments the source text into chunks. These chunks are stored in a vector-oriented database that is consulted for each query. The top-5 chunks with the highest scores for the query are retrieved using the exact K-Nearest Neighbors Search method and compared against the gold standard for each query. This process is repeated for all queries, and the final result for each metric is its mean across all queries. Beyond the metrics already implemented in the framework (Recall, Precision, F-1) we added the Discounted Cumulative Gain (DCG), mean chunk length in tokens, as well as the time to process the source text. Since each chunker splits the text differently, the evaluation metrics are calculated by comparing the token-level gold standard to the chunks retrieved for each query. No relevance information is given to the chunkers.

Baselines. The methods described in Section 2 (Fixed, Recursive, Kamradt, Cluster, and Lumber) were used as baselines. The Proposition Chunker (T. Chen et al., 2024) was not suitable for this experiment, as its method for splitting text into propositions involves an LLM-driven transformation of the original tokens. This alters the source text in a way that precludes direct comparison with the gold standard tokenization, which is essential for our evaluation.

Tools and configurations. The recursive and fixed chunkers were taken from the

popular implementation by langchain[2]. The Kamradt and Cluster methods were taken from the Chroma chunk evaluation repository (Smith and Troynikov, 2024). The Lumber chunker was slightly modified to receive plain text (splitting into paragraphs in execution time) and set to use the Deep Seek R1 model (DeepSeek-AI et al., 2025) as a free open source alternative to the generative model. DSC was configured to use a distance d = 10 from the diagonal and a standard deviation multiplier k = 3 to generate the statistical threshold θ. We used the en_core_web_sm spaCy model for block extraction. To generate and store the embeddings into the database, we used the sentence-transformers/stsb-mpnet-base-v2 (Reimers and Gurevych, 2019) from HugginFace. Fixed, Recursive, Cluster, and DSC have parameters to control chunk length. We set the maximum length to 100 tokens in order to minimize the effects of chunk granularity (Kamradt's method only provides a target length, and Lumber chunker does not have any parameters for length).

Table 2. Chunking evaluation results

chunker	↑ DCG	↑F1	↑P@5	↑R@5	↓time(s)	avg length
Fixed	0.291	0.070	0.077	0.641	**0.110**	92.201
Recursive	0.384	0.092	0.101	0.633	0.481	71.885
Kamradt	0.300	0.056	0.075	0.593	22.521	115.936
Cluster	0.462	0.105	0.120	**0.646**	14.785	62.185
Lumber	0.054	0.010	0.014	0.454	96,970.506	657.953
DSC (ours)	**0.571**	**0.110**	**0.148**	0.484	465.282	44.368

5. Results

Our results are summarized in Table 2. DSC has superior retrieval performance in terms of F1, DCG, and Precision@5. The creation of shorter, more granular chunks, though semantically rich, led to lower recall scores as it retrieved less text overall. However, DSC's improvement in quality comes with an overhead in terms of time, which is expected since we use two inference processes. We argue that the gains in quality outweigh the higher computational costs, considering that chunking is an off-line process that is typically run once. Furthermore, the time taken by DSC is only a small fraction of the time taken by the Lumber chunker.

Unlike the other metrics, chunk length does not have a predetermined ideal value. Intuitively, one wants to avoid chunks that are too small and take up more storage space in the vector database and, at the same time, avoid having very long chunks that include noise and are not ideal for vector representation.

In order to assess the impact of DSC's chunk length on retrieval quality, we ran an experiment in which we varied token length from 46 to 283 and computed the evaluation metrics. The results are shown in Fig 4. We can see that chunk length has a noticeable impact on quality metrics. While DCG, F1, and Precision decrease as the chunks get longer, recall increases and peaks (at 62%) around 100 tokens.

This control of chunk length offered by DSC brings more flexibility to applications, yielding better control over the desired behavior of the retriever.

[2]https://python.langchain.com/v0.2/docs/introduction/

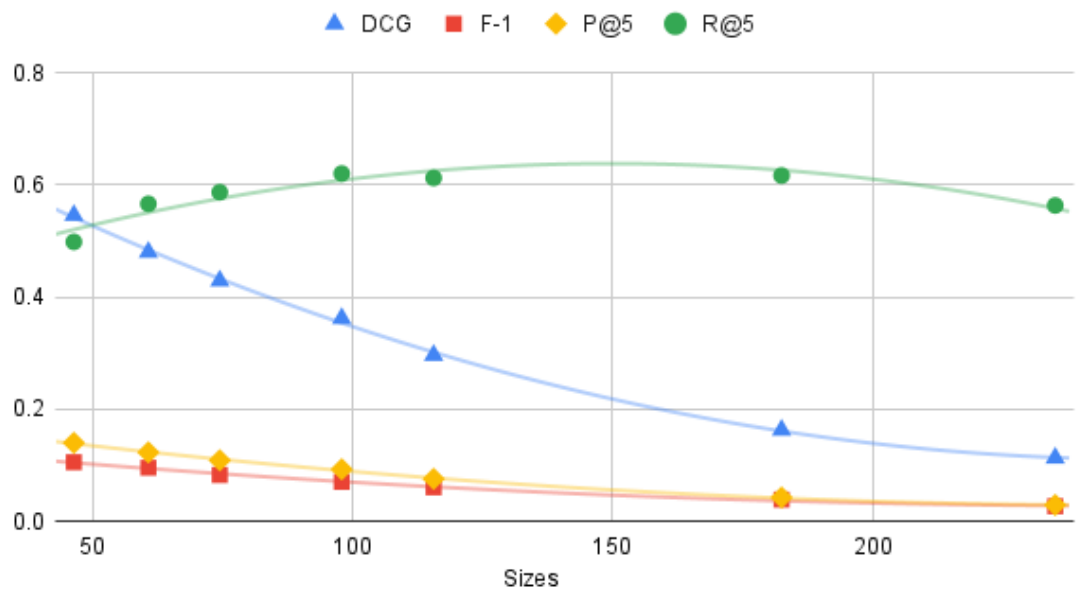

Figure 4. DSC chunk length impact on retrieval quality

6. Conclusion

This work introduced DSC, a novel approach for semantic chunking developed to produce better-performing chunks in IR systems by using recent advances in semantic representation. We aimed to address the challenges in handling large unstructured text to allow for effective retrieval performance. Our findings suggest that adopting sophisticated chunking methods can lead to systems that better understand and process complex datasets. Future work could further explore statistical methods in the similarity distribution of the corpus to better decide on splitting points, as well as compare the performance of different language models on chunking tasks.

Acknowledgments

This work has been partially funded by CENPES Petrobras, CNPq-Brazil, and Capes Finance Code 001.

References

Chen, Tong, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu (Nov. 2024). "Dense X Retrieval: What Retrieval Granularity Should We Use?" In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, pp. 15159–15177. DOI: 10.18653/v1/2024.emnlp-main.845. URL: https://aclanthology.org/2024.emnlp-main.845/.

Chen, Zhiyu, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang (2022). "ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering". In: *Proceedings of the EMNLP*. Association for Computational Linguistics, pp. 6279–6292.

DeepSeek-AI et al. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/abs/2501.12948.

Ding, Ning, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou (2023). "Enhancing Chat Language Models by Scaling High-quality Instructional Conversations". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051.

Duarte, André, João Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo Oliveira (2024). "LumberChunker: Long-Form Narrative Document Segmentation". In: *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6473–6486.

Gharehchopogh, Farhad Soleimanian and Zeinab Abbasi Khalifelu (2011). "Analysis and evaluation of unstructured data: text mining versus natural language processing". In: *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, pp. 1–4.

Hilbert, Martin and Priscila López (2011). "The world's technological capacity to store, communicate, and compute information". In: *science* 332.6025, pp. 60–65.

Izacard, Gautier and Édouard Grave (2021). "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880.

Kamradt, Greg (2024). *Semantic Chunking*. https://github.com/FullStackRetrieval-com/RetrievalTutorials/tree/main/tutorials/LevelsOfTextSplitting.

Karpukhin, Vladimir, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen Tau Yih (2020). "Dense passage retrieval for open-domain question answering". In: *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*. Association for Computational Linguistics (ACL), pp. 6769–6781.

Koshorek, Omri, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant (2018). "Text Segmentation as a Supervised Learning Task". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp. 469–473.

Lee, Jinhyuk, Alexander Wettig, and Danqi Chen (2021). "Phrase Retrieval Learns Passage Retrieval, Too". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3661–3672.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. (2020). "Retrieval-augmented generation for knowledge-intensive NLP tasks". In: *Advances in Neural Information Processing Systems* 33, pp. 9459–9474.

Merity, Stephen, Caiming Xiong, James Bradbury, and Richard Socher (2022). "Pointer Sentinel Mixture Models". In: *International Conference on Learning Representations*.

National Library of Medicine (2023). *PMC Open Access Subset*. Dataset retrieved from Hugging Face Datasets (PubMed Central Open Access dataset, Version 2023-06-17). Available: `https://huggingface.co/datasets/pmc/open_access`, cited 2024-05-08.

Porzel, Robert and Iryna Gurevych (2003). "Contextual coherence in natural language processing". In: *International and Interdisciplinary Conference on Modeling and Using Context*. Springer, pp. 272–285.

Reimers, Nils and Iryna Gurevych (Nov. 2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, pp. 3982–3992. DOI: `10.18653/v1/D19-1410`. URL: `https://aclanthology.org/D19-1410/`.

Smith, Brandon and Anton Troynikov (July 2024). *Evaluating Chunking Strategies for Retrieval*. Tech. rep. `https://research.trychroma.com/evaluating-chunking`. Chroma.

The White House (2024). *State of the Union 2024*. Accessed: 2024-05-02.

Vaswani, A (2017). "Attention is all you need". In: *Advances in Neural Information Processing Systems*.

Yang, Wei, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin (2019). "End-to-End Open-Domain Question Answering with BERTserini". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 72–77.