

# Small Language Model, Adaptive Fragmentation in Long-Text Summarization

Diego Uribe\*, Enrique Cuan, Iscander A. Ramirez

TecNM/Instituto Tecnológico de La Laguna,  
Mexico

{duribea, ecuand, iramirez}@lalaguna.tecnm.mx

**Abstract.** While Large Language Models (LLMs) excel at diverse NLP tasks, their computational demands limit deployment in specialized applications. Thus, Small Language Models (SLMs) come into play: this work investigates the use of an SLM for long-text summarization. As the SLM's context window is small, fragmentation of a long document is mandatory. We systematically evaluate three preprocessing strategies for the fragmentation of long texts: naive token chunking, classical TextTiling, and neural sentence-embedding segmentation, all employed within a hierarchical summarization pipeline using DistilBART. The results of the experimental evaluation based on stratified sampling exhibit how suitable a fragmentation method is for a particular length of text. Actually, experiments on CNN/DailyMail articles reveal a length-dependent performance pattern: token chunking performs best for documents under 1,584 tokens, while neural segmentation excels for documents exceeding 2,250 tokens. Our results yield a practical, adaptive fragmentation policy that balances summary quality with computational efficiency.

**Keywords.** Language model, summarization, lexical tokens.

## 1 Introduction

Large Language Models (LLMs) are very successful for a diverse range of NLP tasks -as translation, summarization, classification, etc.- making them essential for users with multiple and diverse information needs. However, the versatility and power of these models demand enormous computational resources, which limit their use in the development of specific NLP applications for users with specific information

needs. Small Language Models (SLMs) present a plausible alternative, offering reduced footprint while maintaining competency on specific tasks. This paper investigates SLMs for document summarization, focusing particularly on the challenge of processing documents that exceed the model's fixed context window.

Automatic text summarization is a complex field in NLP because producing a summary entails not only taking care of a shorter version of the original text but also taking care of repetition and semantic coherence [4]. Automated text summarization is commonly categorized in three different ways: Standard summarization, Long-document summarization, and Multi-document summarization.

Standard summarization is characterized by a single, short document (500-1000 words) where the purpose is to identify the key facts of the input text. On the other hand, Multi-document summarization is characterized by multiple-texts as input where the challenge lies in creating a summary that integrates pertinent information from various sources avoiding repetition to generate a coherent summary [19].

The main distinctive feature of Long-document summarization is certainly the length of the document. Shortening a long text efficiently entails proper management of the text length, as the salient content is sparsely distributed [27].

Research in automatic text summarization has been addressed with various techniques grouped into three approaches: extractive, abstractive, and hybrid approaches.

The purpose of extractive methods is to select relevant sentences or phrases from the input text to create the summary [3, 9], whereas abstractive approaches are focused on the production of new and equivalent sentences that indicate the original meaning [21, 22]. The hybrid approach is an amalgamation technique of the previous ones: a hybrid technique uses an extractive model to identify key sentences, followed by an abstractive model to improve the generation of new sentences based on the extracted content [8, 12, 17].

Long-Text Summarization (LTS) is a challenging task because the length of the text requires proper management of the content, as the key points are sparsely distributed. Moreover, one more demanding constraint is in our work: the use of a Small Language Model (SLM) where the length of the context window is smaller than its counterpart in a Large Language Model (LLM). For example, an LLM like LongT5 displays great performance on various NLP tasks. However, it is quite a large model whose storage and computational requirements are huge. On the other hand, the deployment of summarization models on resource-limited devices makes the use of an SLM based on the distillation technique a plausible option [14].

Since the length of the context window of the SLM used in this work (DistilBART) is limited to 1,024 lexical tokens, and the dataset contains articles with a length greater than 1,024 tokens, proper management of the text length is a crucial component of an LTS system. While truncation represents a straightforward solution, it risks discarding crucial information. So, a suitable fragmentation of the text is demanding. How can we split the text when the salient information is sparsely distributed? Fragmentation strategies must therefore balance semantic coherence with practical constraints, yet no comprehensive evaluation exists comparing fragmentation approaches for SLM-based summarization.

That is why our purpose is to investigate preprocessing strategies for handling long documents. We systematically evaluate three preprocessing strategies for the fragmentation of long texts: naive token chunking, classical TextTiling [10], and neural sentence-embedding

segmentation. In the shallow fragmentation method, a long text is divided into chunks of 1,024 tokens, the size of the context window of the SLM used, whereas in the semantic fragmentation techniques implemented with the TextTiling algorithm and neural sentence-embeddings, non-overlapping subtopic segments are extracted.

In short, we have implemented a hybrid extractive-abstractive summarization approach. In the extractive step we use the fragmentation strategies previously mentioned. In the abstractive stage we implement a hierarchical approach: the summary corresponding to each fragment (chunk or semantic segment) is obtained first, and then, the final summary is generated. Said in another way, the final summary is a summary of summaries.

Our investigation makes three primary contributions:

- it provides insights into the plausibility of eliciting good-quality summaries of long texts by using a SLM. Previous work that uses larger long-context models often requires substantial computational resources for both training and inference.
- we conduct a systematic comparison of fragmentation methods to demonstrate that optimal fragmentation choice depends fundamentally on document length (e.g. simple methods outperforming complex ones for shorter texts).
- we propose and validate an adaptive fragmentation policy that selects strategies based on document characteristics.

## 2 Fragmentation and Hierarchical Approach

LTS is a challenging task because long documents have relevant information spread throughout the text: critical information can be conveyed anywhere in the document, not just at the beginning or end. Furthermore, the use of an SLM with a small context window (1,024 lexical tokens) makes this affair a more challenging task [6].

This fundamental limitation demands the fragmentation of the long document.

We next describe the fragmentation approaches analyzed in this work, but we first formalize our task in similar terms to those presented by Zhang et al. [26].

LTS can be defined as the following mapping function:

$$f_{\phi} : T \rightarrow S, \quad (1)$$

where  $f_{\phi}$  is the language model with trainable hyperparameters  $\phi$ ,  $T$  is the set of tokens in the long input text, and  $S$  is a new smaller text that preserves the semantic of  $T$ .

Now the process (P) of LTS is formalized as a sequence of steps for transforming and analyzing data. These steps are represented by the following tuple:

$$P = (D, SLM, E), \quad (2)$$

where  $D$  denotes data preprocessing, which consists of transformations  $\phi(T)$  such as *tokenization* and *semantic fragmentation*.

SLM is the modeling step where a pre-trained SLM (*DistilBart* in our case) is applied to produce the summary, and  $E$  is the evaluation stage where metrics as ROUGE and BERTScore are used to measure the syntactic and semantic match between  $T$  and  $S$ .

## 2.1 Shallow Fragmentation

Shallow fragmentation is basically a rudimentary splitting approach to cope with the length of the text. As part of the pre-processing of a long document, this splitting approach represents the text in terms of the number of tokens obtained from the tokenization process. Thus, in this shallow fragmentation a long text is divided into chunks of tokens:

$$Chunking(T) \rightarrow c_1, c_2, \dots, c_n, \quad (3)$$

where:

$$c_i = tk/cwl, \quad (4)$$

with  $tk$  being total number of tokens, and  $cwl$  being context window length, where  $cwl = 1,024$  tokens for *DistilBart*.

## 2.2 Semantic Fragmentation

Semantic fragmentation is basically a fragmentation technique based on the content rather than the length of the text. We implement this fragmentation approach with two semantic methods: classical TextTiling and neural sentence-embedding segmentation.

### 2.2.1 TextTiling

TextTiling is basically an algorithm designed for text segmentation by detecting subtopic boundaries [10]. Indeed, TextTiling's goal is to divide a long document into smaller, topically coherent segments (or "tiles") by looking for shifts in the vocabulary (lexical co-occurrence) between adjacent blocks of text.

Thus, TextTiling is a crucial pre-processing step or component to break the long text ( $t$ ) into a sequence of smaller, topically coherent segments  $tt_1, tt_2, \dots, tt_n$ :

$$TextTiling(t) \rightarrow tt_1, tt_2, \dots, tt_n, \quad (5)$$

where  $tt_i$  and  $tt_{i+1}$  denote adjacent blocks of token-sequences to be compared for lexical similarity.

Two parameters for customization are:  $w$  is the size of the token-sequence,  $k$  is the number of token-sequences grouped together into a block to be compared.

When the algorithm detects a significant drop in the similarity scores, a potential topic boundary is identified.

### 2.2.2 Sentence Embeddings

Sentence embeddings are widely used in NLP, particularly in the summarization task. For example, the use of sentence embeddings to extract relevant and salient sentences in [7]. But while Gao et al. make use of sentence embeddings to capture associations between neighboring sentences to prediction purposes, we look for associations between contiguous sentences to create cohesive segments.

Sentence Embedding's aim is to identify boundaries based on similarity drops. The mean

and standard deviation of similarities are calculated and a threshold is defined for identifying segment boundaries. Hence, a lower similarity indicates a greater semantic shift, suggesting a boundary.

Thus, Sentence Embedding is a crucial pre-processing step or component to break the long text ( $t$ ) into a sequence of smaller, cohesive and coherent segments  $se_1, se_2, \dots, se_n$ :

$$\text{SentenceEmbedding}(t) \rightarrow se_1, se_2, \dots, se_n, \quad (6)$$

where:

$$\text{segment}(s_k) = \begin{cases} s_k \in se_i & \text{if } s_k \geq \text{threshold} \\ \text{Create new } se_j & \text{otherwise,} \end{cases}$$

and:

$$\text{threshold} = \text{mean} - (ts * \text{stddev}), \quad (7)$$

where  $ts$  determines the size of the similarity window. For example, a value of 0.5 is used for a moderate threshold.

## 2.3 Hierarchical Approach

LTS is a challenging task because the length of the text requires proper management of the content, as the key points are sparsely distributed, and one more demanding constraint is the use of *DistilBart* with a CWL limited to 1,024 tokens.

This constraint is the reason for a pre-processing based on fragmentation and the implementation of a Hierarchical Approach. The workflow for this approach is:

$$\text{Pipeline}(T) \rightarrow f_1, f_2, \dots, f_n | \text{DistilBart}(f_i), \quad (8)$$

$$\text{Fragments}(T) \rightarrow \text{Pipeline}(T) \rightarrow S_1, S_2, \dots, S_n, \quad (9)$$

where  $f_i$  denotes a particular  $c_i$  (generated by tokenization), or  $s_i$  (generated by TextTiling), and  $S_i$  denotes the summary generated by *DistilBart* corresponding to a particular  $f_i$

The final step in the process is:

$$\text{DistilBart}(S_i) \rightarrow \text{FinalSummary}, \quad (10)$$

In this way, we have in  $S_1, S_2, \dots, S_n$  a sequence of all main ideas represented by

individual topic summaries. Then, this sequence is submitted to *DistilBart* to connect and remove redundancy across fragments, and to create a final summary ensuring both completeness (all topics are covered) and coherence (the final text flows well).

## 3 Experimental Setup

In this section we describe the experimentation conducted in this work to figure out the implications of fragmentation techniques for long-text summarization (LTS). So we first briefly describe the compact summarization model used in this investigation: *DistilBart*. As a fundamental part of the long-text summarization process, the main characteristics of this small language model (SLM) are detailed. Then, the dataset characteristics are presented, and the results of each fragmentation technique are exhibited.

### 3.1 DistilBart: A Compact Summarization Model

For our summarization tasks, we employ *DistilBart*, a distilled version of the BART model [5]. Through knowledge distillation, it retains 97% of BART's summarization performance while being 40% smaller and 60% faster. This encoder-decoder transformer is pre-trained for text generation and is particularly effective at abstractive summarization, making it an efficient and powerful choice for our evaluation.

In fact, abstractive summarization aims to condense a source text by generating novel phrases and sentences that capture its core meaning, rather than simply extracting key sentences verbatim. Unlike extractive methods, it can paraphrase, generalize, and synthesize information to produce a more concise and fluent summary, much like a human would [22].

Moreover, we selected this model for its optimal balance of performance and efficiency. Its standard context window of 1,024 tokens necessitates the fragmentation of longer input documents, allowing us to cleanly evaluate the impact of different preprocessing strategies. Using a smaller model ensures that any differences in summary quality

are due to the fragmentation method itself, and not simply the model's ability to overpower the problem with its large size.

### 3.2 Dataset Description

The dataset used in this experimentation is CNN-DM, a dataset containing just over 300k unique news articles as written by journalists at CNN and the Daily Mail [11]. Each news article is described by a body content and its corresponding summary (highlights) created by the editors. Training documents average 766 words across 29.74 sentences, with summaries averaging 53 words and 3.72 sentences. Figure 1 shows the distribution of the dataset in terms of article-tokens-length, where most of the news articles are characterised by no more than 1024 tokens. However, the news dataset also contains 76k (out of 300k) articles denoted by more than 1024 tokens.

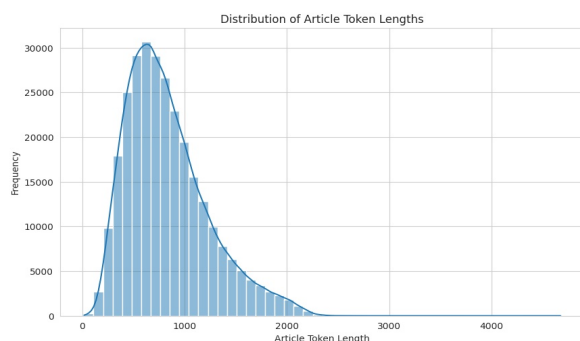


Fig. 1. Article-Token-Length-distribution

Since we are concerned in this work with the summarization task of long texts (LTS), a pertinent question is: how is a text considered as a long text? But to define a document as a long document is a matter for debate. According to Koh et al., a document is considered as a long document based on the current state-of-the-art models [15]. In 2018, a research work about abstractive summarization made use of the CNN/DM dataset when the articles were considered as long-text [2].

However, recent work that makes use of transformers for long sequences considers a document with 3,000 lexical tokens at least, as

long documents [23]. In our research work, a document is considered as a long document if the tokens-length exceeds 1,024 lexical tokens. Such definition is based on the size of the DistillBart (SLM) context window, as a small context window demands a suitable fragmentation of a text characterised by more than 1,024 lexical tokens.

In a previous work, the dataset for experimentation was obtained from a random sampling method. But this time, we propose and validate an adaptive fragmentation policy that selects segmentation strategies based on document characteristics. Thus, a stratified sampling has been implemented for the preparation of the dataset for experimentation.

Three groups have been defined in terms of the text's token length: **small** ( $tokenlength < 1,750$ ), **medium** ( $tokenlength < 2,250$ ) and **long** ( $tokenlength \geq 2,250$ ).

Table 1. Token Length distribution

| Sample        | Bins  | From | To   |
|---------------|-------|------|------|
| <i>Small</i>  | Bin 1 | 1250 | 1416 |
|               | Bin 2 | 1417 | 1583 |
|               | Bin 3 | 1584 | 1750 |
| <i>Medium</i> | Bin 1 | 1751 | 1916 |
|               | Bin 2 | 1917 | 2083 |
|               | Bin 3 | 2084 | 2250 |
| <i>Long</i>   | Bin 1 | 2251 | 2683 |
|               | Bin 2 | 2684 | 3116 |
|               | Bin 3 | 3117 | 3550 |

We have 1,000 articles for small and medium respectively, and 235 for long articles (Figure 1 shows how the available number of articles greater than 2,250 is scarce). Furthermore, to conduct a more detailed analysis of the fragmentation techniques previously described (Section 2), each group has been divided into subgroups or *bins*. Table 1 shows the token length distribution for each group and its corresponding bins.

### 3.3 Metrics

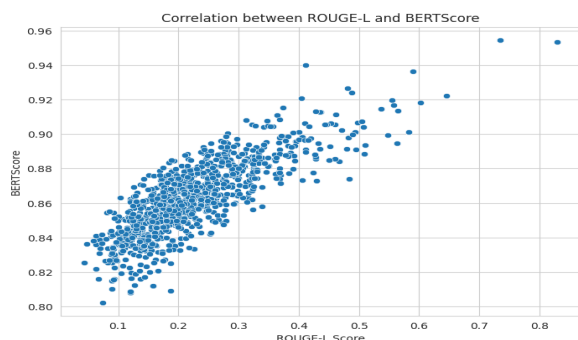
To assess the quality of generated summaries, we apply two widely used evaluation metrics:

ROUGE.L [16] and BERTScore [24]. While the latter is based on contextual embeddings to identify semantic similarity, the former makes use of the longest common subsequence to detect lexical matching. By making use of these metrics, we aim to provide a comprehensive assessment of summarization performance, ensuring both content relevance and linguistic quality.

## 4 Results and Analysis

Before presenting the results for each fragmentation alternative, and regardless of the fragmentation implemented, we observe a common pattern in the obtained results: a positive correlation between the evaluation metrics used in this work.

Figure 2 shows how, as the ROUGE.L value increases, the BERTScore also increases. In other words, as the lexical match with the highlights (summary created by the editors) increases, the semantic similarity also increases. Figure 2 shows how this correlation pattern is observed in the three analysis cases.



**Fig. 2.** Correlation between ROUGE.L and BERTScore

### 4.1 Fragmentation Results

As we previously said, to conduct a deep analysis of the fragmentation techniques, a stratified sampling on the news articles was implemented. We define three groups in terms of the text's token length: small ( $tokenlength < 1,750$ ), medium ( $tokenlength < 2,250$ ) and long ( $tokenlength \geq 2,250$ ), so the obtained

results for each fragmentation alternative and the corresponding length groups are presented. In other words, this presentation shows how much the quality of a summary generated by a fragmentation method varies for each category.

**Shallow Fragmentation:** in this case, the unit of fragmentation is a chunk of tokens. As the expression (3) indicates, in this shallow fragmentation a long text is divided into chunks of 1,024 lexical tokens. With this fragmentation method, we determine whether the salient information is distributed from the beginning to the end of the article. In this fragmentation, the hierarchical approach, represented in expression (9), is applied to obtain the summary of each chunk first, and then to create the final summary.

Figure 3 shows the distribution of the BERTScores through the three categories of the text's token length. In the small category we see how the median BERTScore decreases as the text length increases.

Then, in the transition from the small to the medium category, we see how the median is similar but there is high spread of values in the first bin indicated by longer whiskers. In this category we also see how the median BERTScore decreases as the text length increases.

In the transition from the medium to the long category, the median BERTScore is higher but there is high variability in the first bin indicated by longer whiskers and various outliers. We also see in this long category how the median BERTScore decreases in the second and third bins.

In short, Figure 3 shows how this fragmentation struggles with the text length.

**TextTiling Fragmentation:** in this case, the unit of fragmentation is a semantic segment i.e. a subtopic segment. As the expression (5) indicates, in this semantic fragmentation a long text is divided into topically coherent segments. The values of the TextTiling's parameters used in this experimentation are  $w = 30$  (window size) and  $k = 10$  (number of blocks). Since this method emphasizes the identification of subtopic boundaries, we don't know in advance how many segments represent an article. We also apply the hierarchical approach (expression 9) to obtain the

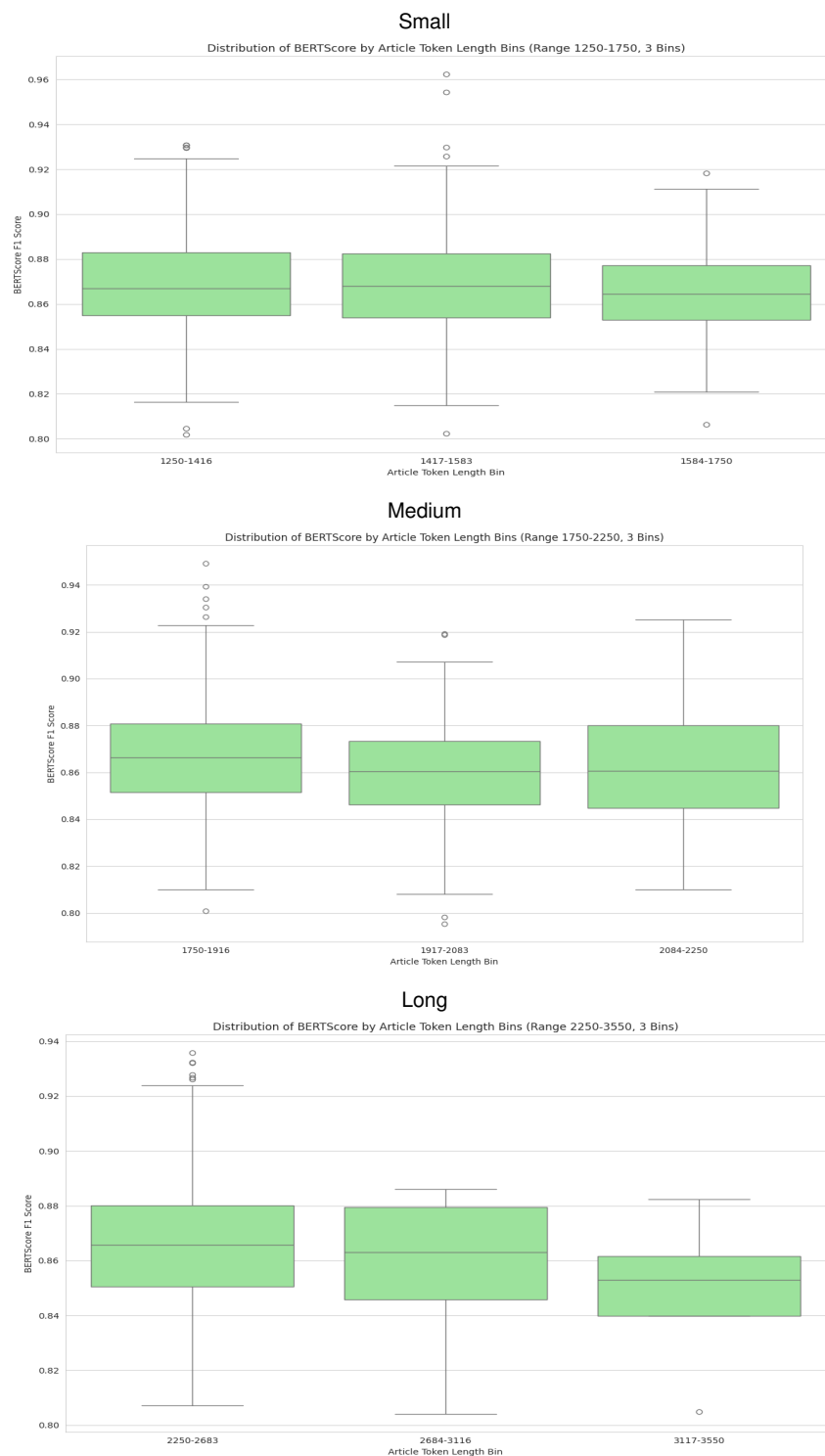
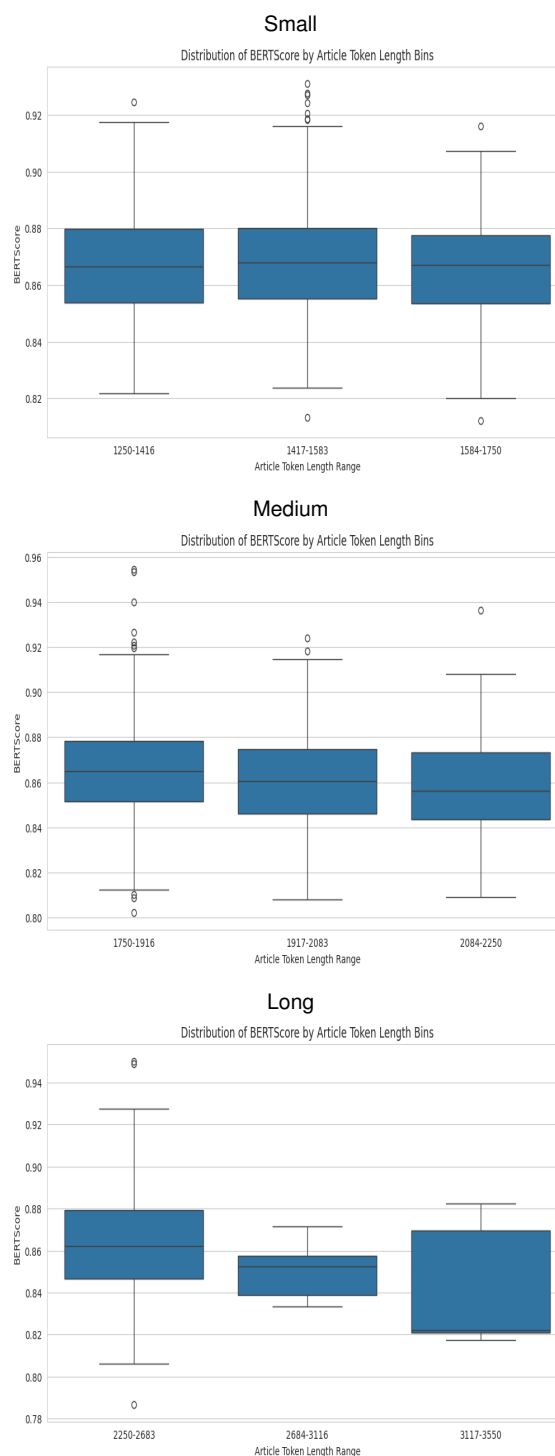
**Fig. 3.** Tokenization performance for the three groups of samples

Figure 4 shows the distribution of the BERTScores through the three categories of the text’s token length. In the small category we see how the three bins exhibit similar median BERTScores but the second bin shows variability with various outliers.

We also see in this long category how the median BERTScore decreases in the second and third bins with a big IQR in the third bin. In general, Figure 4 shows how this fragmentation keeps a stable and decent performance in the first two groups, but only in the third group struggles with the text length.

While a lower value produces many segments that result in lower summary quality, this value generates a moderate number of semantic segments. As this method emphasizes the identification of similarity drops, we don't know in advance how many segments represent an article. We also apply the hierarchical approach (expression 9) to obtain the summaries of the semantic segments first, and then to create the final summary.

Figure 5 shows the distribution of the BERTScores through the three categories of the text’s token length. In the small category we see how the three bins exhibit similar median



**Fig. 4.** TextTiling performance for the three groups of samples

BERTScores but the first two bins show variability with outliers. The third bin display a more stable performance. Then, in the transition from the small to the medium category, the median BERTScore is similar but longer whiskers show variability in the first bin.

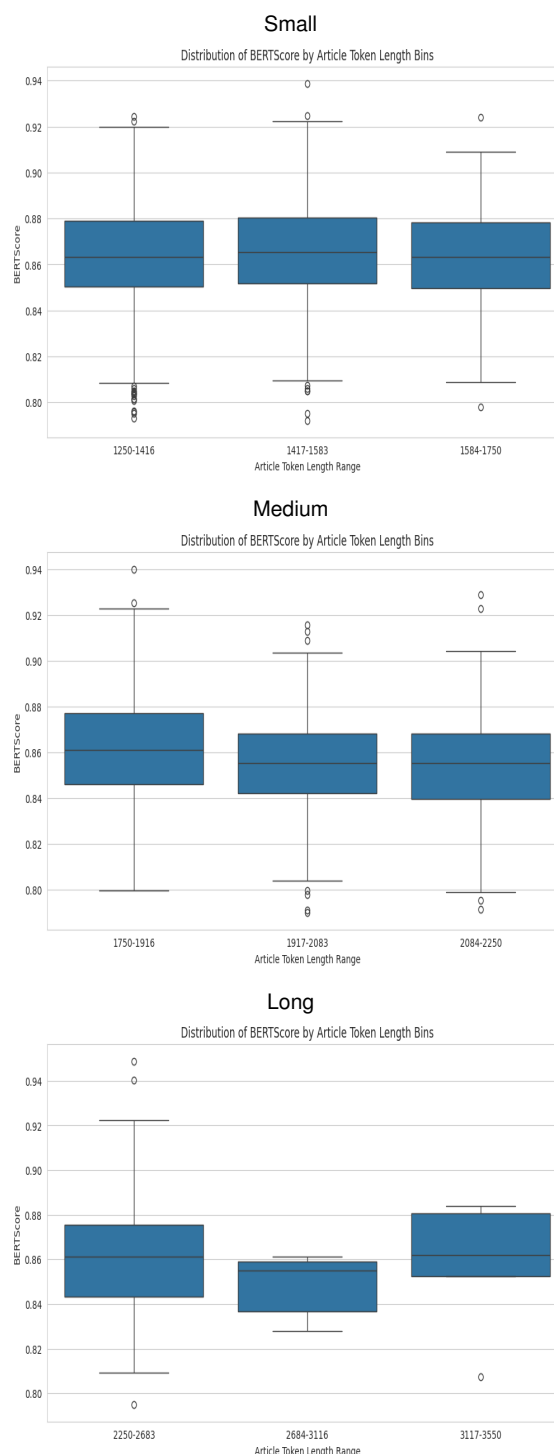
In the last two bins we also see how the median BERTScore decreases as the text length increases. In the transition from the medium to the long category, the median BERTScore is a bit higher and the box shows higher scores in general as an indicator of good performance for long texts.

In the second bin the median decreases just a bit but the best scores are obtained in the third bin with the longest text length. Figure 5 shows how this fragmentation keeps a stable and decent performance in general. But the most important thing is not noticing a significant drop as the text length increases in the last bin.

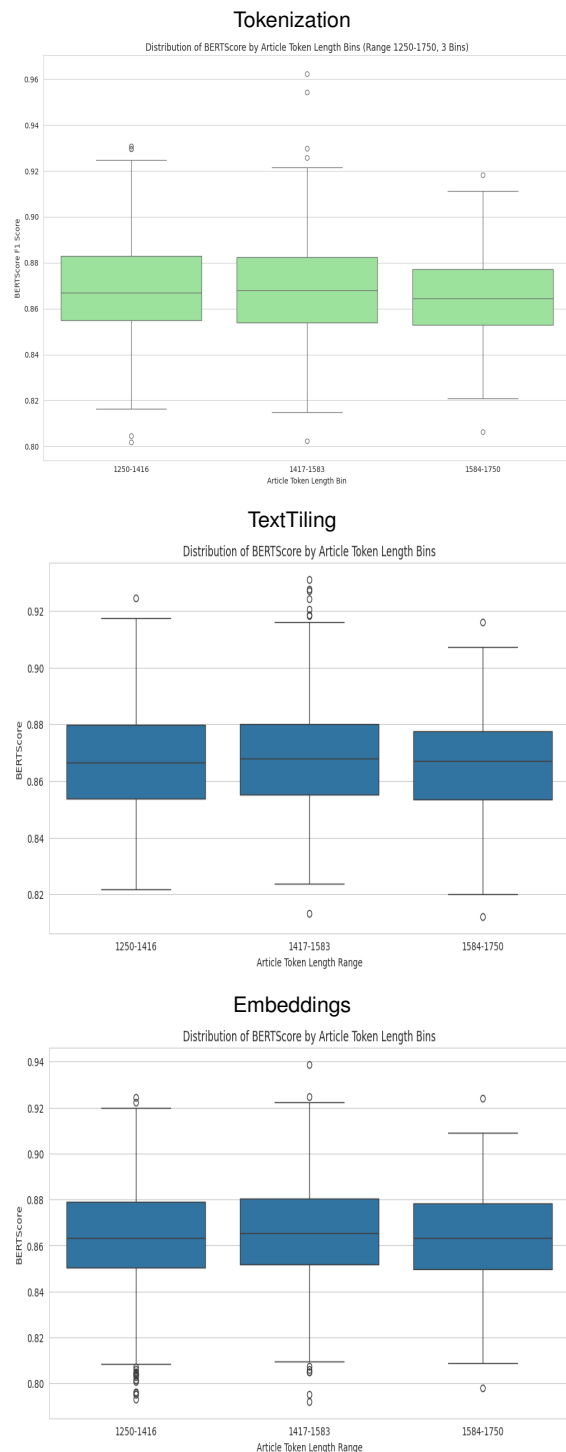
## 4.2 Discussion

While the fragmentation results show how much the quality of a summary generated by a fragmentation method varies for each category, we intend, in this discussion, to determine how much the quality of a summary generated for a particular category varies by the fragmentation method that generates such a summary. In other words, we compare for a particular token-length group, the performance of each fragmentation method.

**Small:** Figure 6 shows the distribution of the BERTScores through the three fragmentation methods for the *small* token-length category. In the Tokenization method we see how the median BERTScore decreases as the text length increases (see bin 3). Now, in the TextTiling method we see a stable and predictable performance in the first and third bins. The whiskers are smaller than in Tokenization. Then, in the Embeddings fragmentation, the median BERTScores are a bit lower than in TextTiling. Moreover, in contrast to TextTiling, more outliers in the first bin make more unstable the results for the final summary. In short, the obtained results show the TextTiling fragmentation as the best option for this group but the Tokenization method also displays good results for the first two bins.



**Fig. 5.** Sentence Embeddings performance for the three groups of samples



**Fig. 6.** Fragmentation methods performance for the *small* group

Fragmentation Selection Logic is as follows:

$$f(t, l) = \begin{cases} \text{Tokenization}(t) & \text{if } l < 1584 \\ \text{TextTiling}(t) & \text{if } 1584 \leq l < 2250 \\ \text{Embeddings}(t) & \text{if } l \geq 2250 \end{cases} \quad (11)$$

**Medium:** Figure 7 shows the distribution of the BERTScores through the three fragmentation methods for the *medium* token-length category.

In the Tokenization method we see how the median BERTScore decreases as the text length increases (see bins 2 and 3). Now, in the TextTiling method we see how the median BERTScore also decreases as the text length increases.

In the third bin, the median BERTScore is almost similar to Tokenization but the scores in TextTiling are most stable and predictable as the IQR in T is greater and the whiskers are longer. Then, in the Embeddings fragmentation, the median BERTScores also decreases as the text length increases. The performance is a bit lower than in Tokenization and TextTiling. In short, the obtained results show the TextTiling fragmentation as the best option for this medium group as the scores in Tokenization shows more variability.

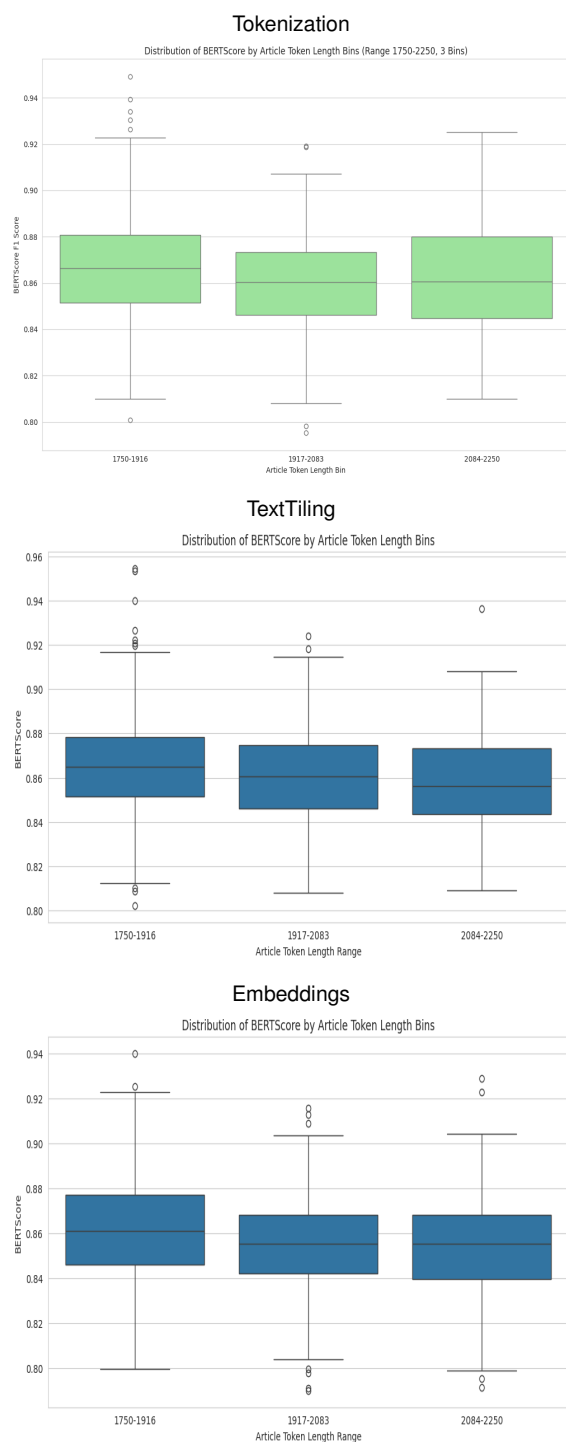
**Long:** Figure 8 shows the distribution of the BERTScores through the three fragmentation methods for the long token-length category. In the Tokenization method we see how the median BERTScore decreases as the text length increases with a significant drop in bin 3.

In the TextTiling method we now see how the median BERTScore also decreases as the text length increases.

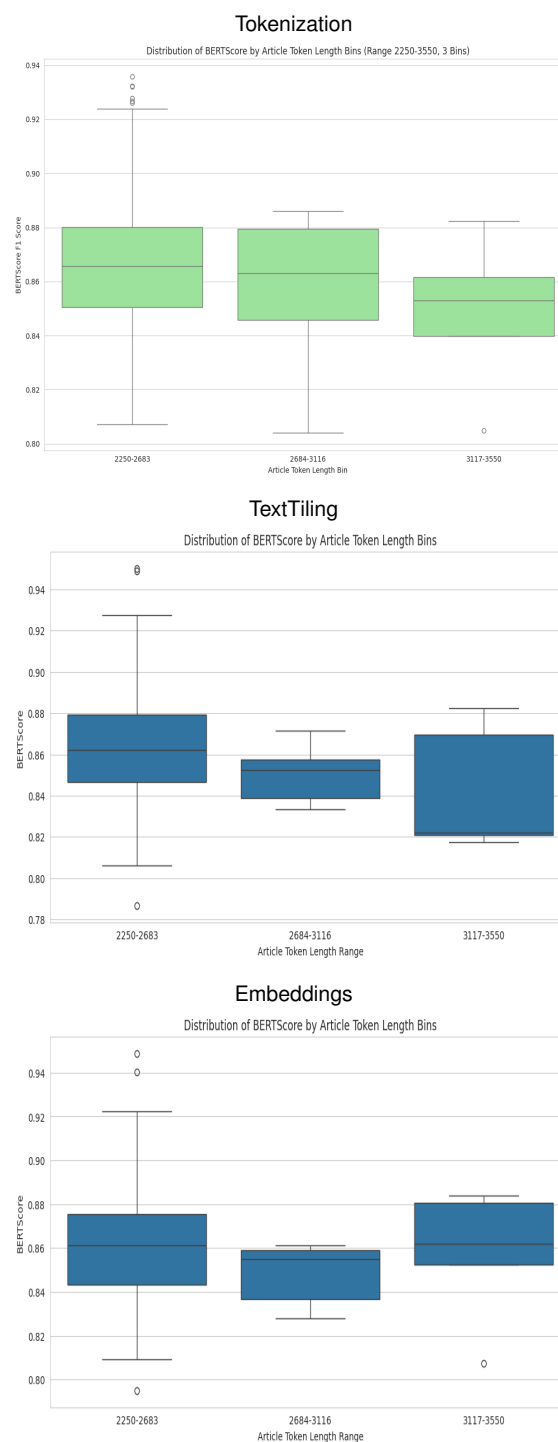
There is also a significant drop in the last bin that shows high variability as the IQR is a big box. Then, in the Embeddings fragmentation, the median BERTScores keeps the performance.

In contrast to Tokenization, the scores are consistent and stable as the boxes and whiskers in Tokenization and TextTiling are bigger. In short, the obtained results show the Embeddings fragmentation as the best option for this long group as the scores are consistent and stable.

Once we have compared for each particular token-length group, the performance of each fragmentation method, and based on the results



**Fig. 7.** Fragmentation methods performance for the *medium* group



**Fig. 8.** Fragmentation methods performance for the *long* group

of our empirical experimentation, we assert a fragmentation adaptive guideline for long-text summarization by using an SLM.

As defined in Equation 11, the system selects the *fragmentation* method based on text's token-length.

In this way, our results yield a practical, adaptive fragmentation policy that balances summary quality with computational efficiency. For example, semantic fragmentation, as classical TextTiling and neural sentence-embedding segmentation, demands greater computational time compared to naive token chunking fragmentation.

While tokenization divides a long text into just four fragments at maximum, semantic fragmentation produces more segments that demand more computational time to generate the final summary. Thus, the tradeoff between the number of fragments and computational time is crucial to produce a good-quality summary for a long text.

## 5 Related Work

Since the focus of our work lies on the intersection of long-document summarization and text segmentation, specifically investigating how preprocessing strategies impact summary quality, this section reviews some relevant advances in efficient model architectures, text segmentation techniques, and the foundational role of content selection, thereby framing the contribution of our empirical study.

A primary direction in addressing long-text summarization has been the development of novel, efficient model architectures to cope with the challenge of generating coherent summaries from documents that exceed the context window of modern transformer models. Models like Longformer [1] introduce sparse attention mechanisms to handle sequences of up to 16,000 tokens.

More recently, research such as that by Huang et al. [13] has further refined these efficient attentions specifically for summarization, demonstrating that architectural innovations can successfully process long-form content.

Even though these approaches are powerful, they often require substantial computational resources for both training and inference.

A critical preprocessing step for long-text summarization is segmentation, or chunking, of the document into manageable pieces. Text segmentation has evolved from classical lexical-cohesion-based algorithms like TextTiling [10] to modern neural approaches.

For instance, Lukasik et al. [18] proposed a cross-segment attention model for segmentation, representing the state-of-the-art in leveraging deep learning to identify topical boundaries. These neural methods offer high accuracy but can introduce complexity and require training data.

Our study deliberately employs the classical TextTiling algorithm to investigate whether a lightweight, unsupervised semantic segmentation method can provide a sufficient advantage over naive chunking, making it a practical choice for many real-world applications.

An interesting work that makes use of *sentences* as the basic units for document summarization is HIBERT. This system is a hierarchical transformer model designed specifically for document-level summarization. The core innovation lies in its two-level pretraining strategy: it first encodes individual sentences with a sentence-level transformer, then aggregates these sentence representations with a document-level transformer to capture the overall discourse structure.

This hierarchical architecture directly addresses the fundamental challenge of modeling long-range dependencies in documents, making it particularly relevant for abstractive summarization where understanding both local context and global narrative flow is essential. The pretraining of the model is based on a novel document reconstruction: instead of predicting masked tokens at the word level like standard BERT, masks entire sentences and trains the model to reconstruct them based on the surrounding sentence representations [25].

As well, *sentence embeddings* are widely used in NLP, particularly in the summarization task. For example, sentence embeddings are used to cope with query-focused extractive summarization, where the goal is balancing

relevance—ensuring the summary addresses the user’s specific needs—and salience—ensuring the summary captures the most important information [7]. To address the challenge of capturing sentence salience, Gao et al. develop a model to represent latent associations between neighboring sentences by predicting a sentence based on its context, rather than relying solely on similarity between queries and sentences. This joint modeling of sentence content and context produces more informative sentence embeddings.

Now, the way we select and order content before the abstractive step profoundly impacts the final summary’s quality and coherence. Nayeem et al. [20] argued for the importance of “extract with order” to maintain coherence, highlighting that the quality of the input units submitted to an abstractive model is paramount. This principle directly informs our hypothesis: the strategy used to fragment a long document—whether it preserves semantic units or not—should materially affect the fluency and informativeness of the final summary.

In short, and as far as we know, previous work often integrates segmentation within a larger, proprietary pipeline, making it difficult to discern its specific contribution. Our study fills this gap by conducting a controlled comparison of fragmentation techniques by making use of a small language model on abstractive summarization.

## 6 Conclusions

In this paper, we have proposed an adaptive text fragmentation policy for long-text summarization having as challenge the use of an SLM as textitDistilBart. We conduct a systematic comparison of fragmentation methods to demonstrate that optimal fragmentation choice depends fundamentally on document characteristics such as document length. In fact, while a shallow fragmentation for documents under 1,584 tokens is enough, semantic fragmentation, like TextTiling and sentence-embedding, is the best option to divide long documents into cohesive and coherent segments to generate a good-quality summary. In this way, our results yield a practical, adaptive fragmentation policy that balances summary quality with computational efficiency.

## References

1. **Beltagy, I., Peters, M. E., Cohan, A. (2020).** Longformer: The long-document transformer. arXiv preprint. ArXiv:2004.05150.
2. **Celikyilmaz, A., Bosselut, A., He, X., Choi, Y. (2018).** Deep communicating agents for abstractive summarization. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1662–1675.
3. **Cheng, J., Lapata, M. (2016).** Neural summarization by extracting sentences and words. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Berlin, Germany, pp. 484–494. DOI: 10.18653/v1/P16-1046.
4. **Cohan, A., Derroncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., Goharian, N. (2018).** A discourse-aware attention model for abstractive summarization of long documents. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), Association for Computational Linguistics, New Orleans, Louisiana, pp. 615–621. DOI: 10.18653/v1/N18-2097.
5. **DistilBart (2021).** Distilbart: sshleifer/distilbart-cnn-12-6. HuggingFace model page. Model card.
6. **Fu, X. Y., Laskar, T. R., Khasanova, E., Chen, C., Bhushan, S. (2024).** Tiny titans: Can smaller large language models punch above their weight in the real world for meeting summarization? Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 387–394.
7. **Gao, Y., Xu, Y., Huang, H., Liu, Q., Wei, L., Liu, L. (2020).** Jointly learning topics in sentence embedding for document

- summarization. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 32, No. 4, pp. 688–699.
8. **Gehrmann, S., Deng, Y., Rush, A. M. (2018).** Bottom-up abstractive summarization. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Brussels, Belgium, pp. 4098–4109. DOI: 10.18653/v1/D18-1443.
  9. **Gong, Y., Liu, X. (2001).** Generic text summarization using relevance measure and latent semantic analysis. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, New Orleans, Louisiana, USA, pp. 19–25. DOI: 10.1145/383952.383955.
  10. **Hearst, M. A. (1997).** Texttiling: segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, Vol. 23, No. 1, pp. 33–64.
  11. **Hermann, K. M., Kociský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P. (2015).** Teaching machines to read and comprehend. *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS 2015)*, pp. 1693–1701.
  12. **Hsu, W.-T., Lin, C.-K., Lee, M.-Y., Min, K., Tang, J., Sun, M. (2018).** A unified model for extractive and abstractive summarization using inconsistency loss. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, pp. 132–141.
  13. **Huang, L., Cao, S., Parulian, N., Ji, H., Wang, L. (2021).** Efficient attentions for long document summarization. *arXiv preprint*.
  14. **Johnson, J. (2025).** Small language models (slm): A comprehensive overview. *Online blog article*.
  15. **Koh, H. Y., Ju, J., Liu, M., Pan, S. (2022).** An empirical survey on long document summarization: Datasets, models, and metrics. *ACM Computing Surveys*, Vol. 55. DOI: 10.1145/3545176.
  16. **Lin, C.-Y. (2004).** Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, pp. 74–81.
  17. **Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N. (2018).** Generating wikipedia by summarizing long sequences. *arXiv preprint*. ArXiv:1801.10198.
  18. **Lukasik, M., Dadachev, B., Simões, G., Papineni, K. (2020).** Text segmentation by cross segment attention. *arXiv preprint*.
  19. **Ma, C., Zhang, W.-E., Guo, M., Wang, H., Sheng, Q. Z. (2022).** Multi-document summarization via deep learning techniques: A survey. *ACM Computing Surveys*, Vol. 55. DOI: 10.1145/3529754.
  20. **Nayeem, M. T., Chali, Y. (2017).** Extract with order for coherent multi-document summarization. *TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*.
  21. **Rush, A. M., Chopra, S., Weston, J. (2015).** A neural attention model for abstractive sentence summarization. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, pp. 379–389.
  22. **See, A., Liu, P. J., Manning, C. D. (2017).** Get to the point: Summarization with pointer-generator networks. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Vancouver, Canada, pp. 1073–1083. DOI: 10.18653/v1/P17-1099.
  23. **Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L. (2020).** Big bird: Transformers for longer sequences. *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

24. **Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., Artzi, Y. (2020).** Bertscore: Evaluating text generation with bert. arXiv preprint. automatic text summarization with exploration of llm-based methods. *Neurocomputing*. DOI: 10.1016/j.neucom.2025.131928.
25. **Zhang, X., Wei, F., Zhou, M. (2019).** HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. **Korhonen, A., Traum, D., Màrquez, L.**, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5059–5069.
26. **Zhang, Y., Jin, H., Meng, D., Wang, J., Tan, J. (2025).** A comprehensive survey on
27. **Zhang, Y., Ni, A., Yu, T., Zhang, R., Zhu, C., Deb, B., Celikyilmaz, A., Awadallah, A. H., Radev, D. (2021).** An exploratory study on long dialogue summarization: What works and what's next. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Findings*.

*Article received on 08/01/2026; accepted on 09/07/2026.*

*\* Corresponding author is Diego Uribe.*