Neural Document Segmentation Using Weighted Sliding Windows with Transformer Encoders

Saeed Abbasi[1], **Aijun An**[1], **Heidar Davoudi**[2], **Ron Di Carlantonio**[3], **Gary Farmaner**[3]
[1]York University, Toronto, Canada
[2]Ontario Tech University, Oshawa, Canada
[3]iNAGO Corporation, Toronto, Canada
{saeedabc,aan}@yorku.ca, heidar.davoudi@ontariotechu.ca, {rond,garyf}@inago.com

Abstract

We introduce a novel Transformer-based method for document segmentation, tailored for practical, real-world applications. This method utilizes overlapping text sequences with a unique position-aware weighting mechanism to enhance segmentation accuracy. Through comprehensive experiments on both public and proprietary datasets, we demonstrate significant improvements, establishing new state-of-the-art standards by achieving up to a 10% increase in segmentation F1 score compared to existing methods. Additionally, we explore the application of our segmentation method in downstream retrieval-augmented question answering tasks, where it improves the quality of generated responses by 5% while achieving up to four times greater efficiency. These results underscore our model's potential as a robust and scalable solution for real-world text segmentation challenges.[1]

1 Introduction

Retrieval-Augmented Generation (RAG) enhances Large Language Models (LLMs) by incorporating relevant external information into their generation processes, leading to more accurate, contextually appropriate, and up-to-date responses. A crucial component of RAG is *text segmentation*, essential for dividing documents into coherent segments that can be efficiently retrieved and utilized in prompts for LLMs.

We develop question-answering systems for automotive drivers, providing answers based on vehicle manuals. This system employs natural language processing (NLP) techniques to interpret user queries, search for relevant information in a knowledge base, and generate answers with an LLM based on the retrieved knowledge. Although LLMs perform well at synthesizing information to produce natural, coherent responses, they usually need to be anchored in relevant knowledge to accurately address domain-specific inquiries and prevent hallucinations, thereby necessitating the integration of RAG. Our QA system's effectiveness hinges on the segmentation of vehicle manuals into semantically coherent chunks, each encapsulating a single topic or subtopic, ensuring that the LLM receives cohesive, relevant information for response generation.

Text segmentation has evolved from rule-based and statistical methods to sophisticated deep learning techniques. Traditional approaches, such as those utilizing lexical overlaps (Hearst, 1997) or semantic relatedness graphs (Glavas et al., 2016), primarily focused on surface-level text features to identify topic boundaries. In contrast, more recent developments leverage RNN and Transformer-based methods, which provide dense, context-aware representations capable of capturing subtle semantic nuances (Koshorek et al., 2018; Lukasik et al., 2020; Zhang et al., 2021; Yu et al., 2023). Many supervised methods tackle text segmentation as a sequence-labeling task to directly predict segment boundaries. In particular, Koshorek et al. (2018) introduced a hierarchical BiLSTM model trained on their automatically labeled dataset, WIKI-727K, derived from English Wikipedia, demonstrating the significance of large-scale training data. Lukasik et al. (2020) proposed BERT-based vanilla and hierarchical architectures for document and discourse segmentation, challenging the traditional reliance on RNNs. The limited input size of Transformers, however, necessitates breaking a longer document into smaller sequences for efficient processing, typically using sliding windows. Zhang et al. (2021) presented a RoBERTa-based sequence labeling framework with adaptive sliding windows that dynamically adjust the processing window based on prior segmentation decisions. Yet, its reliance on the last

[1]Our code is publicly available at https://github.com/saeedabc/WeSWin

identified boundary to initiate the next sequence can hinder scalability in processing long documents and potentially propagate errors to subsequent boundaries. Glavas et al. (2021) also employed sliding windows in their hierarchical RoBERTa-based approach, but suffered from oversimplified sequence formation with fixed-size sentences and sub-optimal aggregation. Yu et al. (2023) proposed a multi-task, sentence-level sequence labeling framework based on Longformer, achieving state-of-the-art on Wiki-727K (Koshorek et al., 2018) and WikiSection (Arnold et al., 2019) benchmarks. However, their use of one-sentence overlap in sliding windows does not fully alleviate the context cut-off problem, resulting in performance shortcomings compared to our model, despite the additional training overhead and data requirements. As a recent LLM-based approach in document segmentation, Duarte et al. (2024) proposed a dynamic sliding window method to detect semantic shifts using recurrent prompts to LLMs. However, this approach is computationally demanding, offers limited scalability, and requires initial paragraphs, which may not always be available or applicable across all domains. Despite the advances made by these methods, they all face limitations in their effective and efficient use of context, resulting in sub-optimal segmentation performance.

In this work, we propose an overlapping sliding-window technique for document segmentation that aggregates position-weighted sentence predictions across multiple windows during inference. This method implicitly increases the effective context visibility for individual sentence predictions within a document, without relying on models with large context sizes that would be prohibitively expensive for most practical applications. To optimize the aggregation of these sentence predictions, we introduce position-aware weighting methods that adjust their contribution toward the final decision. We conduct comprehensive experiments using both publicly available datasets and a vehicle manual dataset, demonstrating that our proposed method consistently outperforms existing state-of-the-art approaches.

2 Problem Statement

Given a document $D = \langle s_1, s_2, \ldots, s_n \rangle$ consisting of n sentences, document segmentation aims to divide D into semantically cohesive segments or chunks. To this end, we frame the problem as a sentence-level binary classification task that predicts the probability p_i of each sentence s_i in D being the last sentence of a cohesive segment.

3 Methodology

We introduce WeSWin, a text segmentation method based on **We**ighted **S**liding **Win**dows. This end-to-end Transformer-based model is trained on sequences with sentence-level labels. During the inference phase, overlapping sliding windows are generated from the input document, and decisions regarding each sentence from multiple windows are aggregated using a specialized weighting scheme.

3.1 WeSWin Model Training

We fine-tune and evaluate three commonly-used pretrained Transformer encoders—BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020)—for document segmentation. Since the input size of these models is typically much smaller than the token count of a document, the document must first be partitioned into smaller sequences of tokens for processing.

Training Sequence Formation. Given a document $D = \langle s_1, s_2, \ldots, s_n \rangle$ consisting of n sentences, and their respective binary class labels $\langle y_1, y_2, \ldots, y_n \rangle$, $y_i = 1$ indicates that sentence s_i is the last sentence of a semantic segment, while $y_i = 0$ indicates otherwise. To obtain training sequences, we start from the first sentence of a document. Each sequence includes as many sentences as possible to fill the Transformer's token capacity. The next sequence begins with the last sentence of the previous sequence, and this process continues until all sentences in the document are covered.

Inspired by previous works (Glavas and Somasundaran, 2020; Zhang et al., 2021; Yu et al., 2023), we introduce a special token, $[SNT]$, appended after each sentence in the sequence to encode contextual information. A tokenized sequence S over D is formulated as:

$$\langle [CLS], s_1, [SNT], s_2, [SNT], \ldots s_m, [SNT], [EOS] \rangle$$

where s_i is the ith sentence in the sequence and is tokenized as $t_{i,1}, t_{i,2}, \ldots, t_{i,|s_i|}$ (where $|s_i|$ is the number of tokens in s_i) ,$[CLS]$ and $[EOS]$ mark the beginning and end of the sequence respectively, and $|S| \leq T$, where T is the maximum input size of the Transformer in terms of tokens. For example, T is 512 for BERT and RoBERTa, but it can be set as high as 4096 for Longformer.

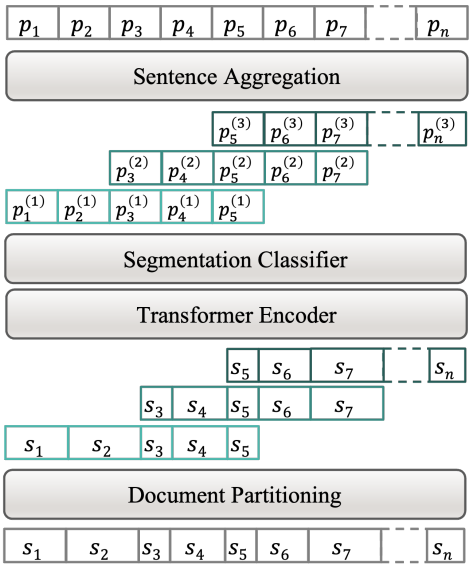

Figure 1: WeSWin inference pipeline. $p_i^{(j)}$ represents the prediction for sentence s_i when observed in sequence j, while p_i represents the final prediction.

Training Pipeline. Given the training sequences along with their sentence labels, we fine-tune a pretrained Transformer encoder model with a segmentation classification head. The transformer takes a sequence as input and computes the contextual representations of its individual tokens. The constructed embedding of each $[SNT]$ token is then fed into the segmentation head, which is a binary Softmax classifier that outputs the probability of each $[SNT]$ token marking the end of a semantic segment. The standard binary cross-entropy loss is used over all $[SNT]$ predictions in the training batch. The exception to this is the last (i.e., rightmost) sentence of a sequence which is not included in loss calculation because it lacks the appropriate context for detecting whether the topic shifts afterward. However, such a sentence is also the first sentence in the next sequence. Thus, the sentence label participates in training in the next sequence, while serving solely as context for the current one.

3.2 WeSWin Model Inference

We propose an inference method using sliding windows over documents, with an adjustable degree of sentence overlap between consecutive sequences. This method allows multiple predictions for individual sentences, effectively increasing the overall context visibility for a more informed, aggregated decision. Figure 1 illustrates the inference pipeline, where tokenized sequences are derived from document partitioning and fed to the Transformer encoder with a segmentation classifier to derive initial sentence predictions. For each sentence s_i, sentence aggregation is applied to the overlapping predictions $p_i^{(j)}$ to derive the final decision p_i.

3.2.1 Inference Sequence Formation

We propose a sliding-window sequence formation method with a *k-Sentence Stride*, referred to as SS-k. With a stride of $k > 0$, each new sequence begins at the $(k+1)$th sentence of the previous sequence (if applicable) and continues up to a maximum of T tokens. This process repeats until the entire document is covered. Due to sequence overlap, a sentence may receive multiple predictions from its inclusion in several sequences. These probabilities are then aggregated to derive a single decision determining the probability of a topic shift for each sentence.

3.2.2 Weighted Aggregation of Multiple Sentence Predictions

Since near-boundary sentences in a sequence experience abrupt context cut-offs (either to the right or left), their predictions may be less robust than those of sentences positioned farther from the boundaries. Therefore, we propose a position-aware weighting mechanism to effectively aggregate the estimated probabilities:

$$p_i = \sum_{j:\, s_i \in S_j} w_i^{(j)} p_i^{(j)}$$

$p_i^{(j)}$ represents the topic shift probability of sentence s_i within the sequence span j, with a corresponding weight $w_i^{(j)}$. The main idea is to adjust the influence of each sentence's prediction on the final aggregated result based on its position within the sequence: Near-boundary sentences receive relatively lower contribution weights. We propose three position-aware weighting functions, selecting the best one based on validation performance.

Linear Positional Weights. The *linear* weight for the ith sentence in a sequence containing m sentences is defined as:

$$\text{lin}(i, m \mid k, \epsilon) = \epsilon + (1-\epsilon) \cdot \left(\frac{\min(d_{i,m}, k)}{k} \right)$$

where $d_{i,m} = \min(i-1, m-i)$ is the distance of the ith sentence to its closer boundary of the sequence, $\epsilon > 0$ is the assigned weight for the boundary sentences, and k is a positive integer controlling how fast the weight increases as the sentence index

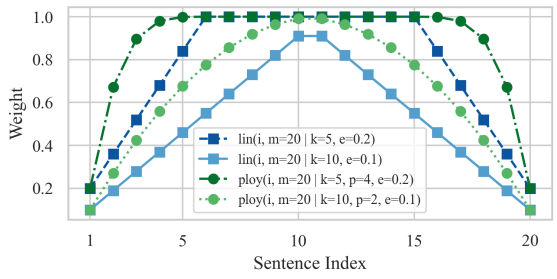

Figure 2: Alternative Linear and Polynomial weighting functions for a sample sequence.

moves from a boundary position to the center. (See the blue curves in Figure 2)

Polynomial Positional Weights. The *polynomial* weight assignment function is defined as:

$$\text{poly}(i, m \mid k, p, \epsilon) = \epsilon + (1-\epsilon) \cdot \left(1 - \left(1 - \frac{\min(d_{i,m}, k)}{k}\right)^p\right)$$

where i, m, $d_{i,m}$, and ϵ are the same as in the Linear function. Here, both k and p control how fast the weight changes. (See the green curves in Figure 2)

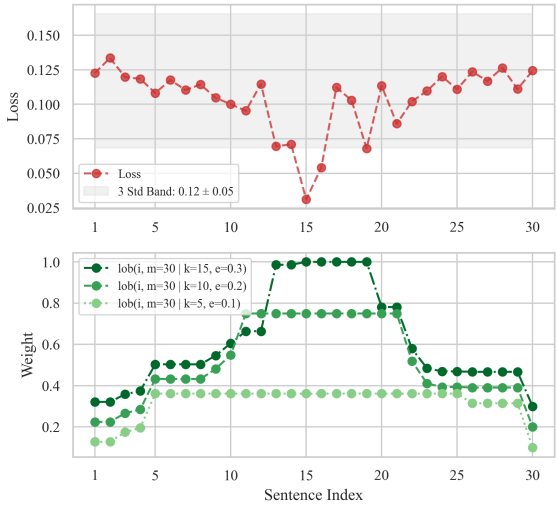

Figure 3: a) A sample average position-loss distribution, $l(.)$, displayed for the first and last 15 sentence positions from each side b) Sample Loss-based positional weighting functions for a sequence of 30 sentences.

Loss-based Positional Weights. We propose a weighting function proportional to the model's performance at individual sentence positions in the validation set. Specifically, we calculate the average validation loss for each sentence position relative to the nearest sequence boundary, denoted as the position-loss distribution $l(.)$. The *loss-based* weight of sentence i in a sequence with m sentences is defined as:

$$\text{lob}(i, m \mid k, \epsilon) = \epsilon + (1-\epsilon) \cdot \begin{cases} \max\limits_{1 \le i' \le i} c(i', m) & \text{if } i \le k, \\ \max\limits_{i \le i' \le m} c(i', m) & \text{if } i > m-k, \\ \max\limits_{k < i' \le m-k} c(i', m) & \text{otherwise.} \end{cases}$$

where $\epsilon > 0$ sets the minimum weight for boundary positions, and the positive integer k controls the (asymmetric) growth rate from boundary positions to the center. $c(.)$ is the normalized complement of $l(.)$, for which the max function serves as a smoothing operator, ensuring weights are non-decreasing from the sides to the center. Figure 3 shows a sample position-loss mass, derived from SS-4 partitioned sequences over 1000 randomly selected validation documents from Wiki-727K, at the top and a few derived weighting functions at the bottom.

4 Experiments

We evaluate our WeSWin model on three publicly available segmentation benchmarks: Wiki-727K (Koshorek et al., 2018) and en_city and en_disease subsets of WikiSection (Arnold et al., 2019). Comprised of Wikipedia articles, Wiki-727K serves as an open-domain benchmark, whereas en_city and en_disease are domain-specific. Additionally, our method is applied and tested on a proprietary dataset known as AutoManual, which features human-labelled segmentation information. We further propose evaluating our method on a downstream RAG task using another proprietary dataset, AutoQA, which contains question-and-answer pairs derived from AutoManual. Further details and statistics of these datasets are provided in Appendix A.1.

4.1 Comparison with SoTA Methods

We compare our proposed model with several state-of-the-art (SoTA) baselines on the Wiki-727K (Koshorek et al., 2018) segmentation benchmark. We trained our model using three different Transformer models: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020). Table 1 shows the results on Wiki-727K, where the models are grouped by the backbone model used. With BERT or RoBERTa

Backbone	Model	Wiki-727K		
		F1	Prec	Rec
RNN	Bi-LSTM (Koshorek et al., 2018)	57.7	69.3	49.5
BERT	Hier. BERT (Lukasik et al., 2020)	66.5	69.8	63.5
	SeqModel:BERT-Base (Zhang et al., 2021)	68.2	70.6	65.9
	WeSWin:BERT (**ours**)	**75.17**	75.35	74.99
RoBERTa	SeqModel:RoBERTa-Base (Zhang et al., 2021)	70.2	66.2	74.7
	WeSWin:RoBERTa (**ours**)	**77.74**	77.97	77.51
Longformer	Longformer-Base+TSSP+CSSL (2048) (Yu et al., 2023)	77.16	-	-
	WeSWin:Longformer-1024 (**ours**)	77.38	77.36	77.39
	WeSWin:Longformer-2048 (**ours**)	**77.91**	79.7	76.2

Table 1: F1, Precision, and Recall comparison of WeSWin with SoTA segmentation models on Wiki-727K. The best and second-best F1 results in each group are highlighted in bold and underlined, respectively. Results for our models are derived from 1000 randomly selected test documents, calculated at the sentence level. Results for the baselines are taken from their respective papers, where a dash ('-') indicates that no data were reported.

as the backbone, our model significantly outperforms the best baseline, achieving more than a 10% relative improvement in F1 score. Specifically, our WeSWin:BERT achieves an F1 score of 75.17, while our WeSWin:RoBERTa achieves 77.74. We trained and tested our Longformer checkpoints with two alternative context size of 1024 or 2048 tokens, both of which outperforming the Longformer-based SoTA with the 2048 context size, achieving F1 scores of 77.38 and 77.91, respectively.[2]

In Table 2, we further evaluate our method on the domain-specific en_city and en_disease datasets (Arnold et al., 2019), comparing it to SoTA methods. The listed checkpoints are pre-trained on Wiki-727K and then fine-tuned on domain-specific data. Notably, our WeSWin:RoBERTa model, despite having an input size of only 512, outperforms the costlier Longformer baseline with an input size of 2048 on the en_city dataset (86.8 vs 85.14 F1 score), while also performing competitively on en_disease (77.26 vs 77.33 F1 score).

4.2 Impact of Overlapping Windows

Trained and tested on Wiki-727K, Figure 4 illustrates the F1 scores across different Transformer baselines when employed with different sliding-window document partitioning methods, denoted by SS-k (for $k = 6, 4, 2$), introduced in Section 3.2.1. Sequence overlap increases as k decreases. SP, or *Single Prediction*, partitions the document using the same sequence formulation method employed during training, as discussed in Section 3.1. In this approach, adjacent sequences share one sentence, and predictions for the last sentence in a sequence are deferred to the next sequence, where it becomes the first sentence.

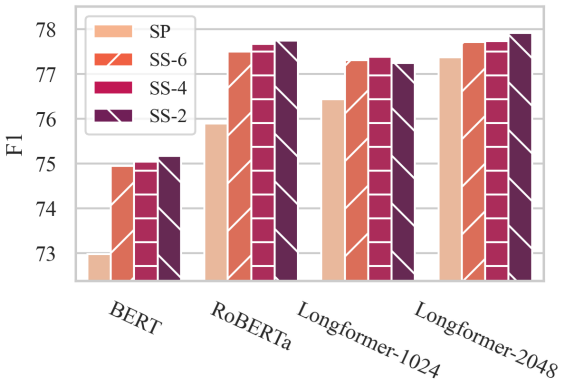

Figure 4: F1 score comparison of our WeSWin:BERT, RoBERTa, and Longformer across SP and SS-k partitioning methods when trained and tested on Wiki-727K.

Figure 4 shows that aggregating predictions for a sentence across overlapping sequences (from SS-k) significantly improves performance compared to the single prediction (SP) setting. Generally, a higher degree of overlap results in higher F1 scores across all Transformer models. The two smaller context models, BERT and RoBERTa, benefit most from overlapped partitioning. However, as the context size increases, as seen with Longformer-1024 and Longformer-2048, the gains from increased overlap become less pronounced.

4.3 Effect of Weighted Aggregation

We evaluate the impact of proposed weighting functions on aggregated predictions from overlapping windows. Figure 5 compares F1 scores from different weighting functions (including *uniform*) on Wiki-727K using WeSWin:BERT. SS-k document partitioning is employed in this experiment with five different stride values. Results generally indicate that all proposed weighting methods im-

[2]Consistent with the literature, the last sentence in a document is always excluded from evaluation in this work, and all section headers are simply omitted.

Model	en_city			en_disease		
	F1	Prec	Rec	F1	Prec	Rec
SECTOR >T+bloom (Arnold et al., 2019)	74.9	-	-	59.3	-	-
LongT5-Base-SS (Inan et al., 2022)	82.3	-	-	68.3	-	-
Longformer-Base+TSSP+CSSL (2048) (Yu et al., 2023)	85.14	-	-	**77.33**	-	-
WeSWin:RoBERTa (512) (**ours**)	**86.8**	88.82	84.86	77.26	76.51	78.03

Table 2: F1 score comparison of WeSWin:RoBERTa with SoTA segmentation models on en_city and en_disease.

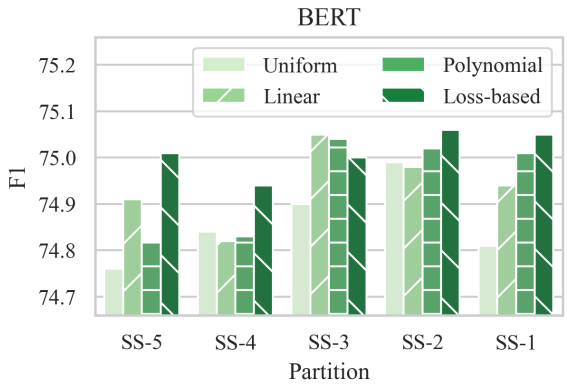

Figure 5: F1 score comparison of Uniform, Linear, Polynomial, and Loss-based weighting methods across several partitioning settings, tuned and tested with WeSWin:BERT.

prove the model's accuracy compared to Uniform aggregation. Among these methods, Loss-based weighting tends to outperform Linear and Polynomial approaches. However, since this is not always the case across different models, we consider all methods as potential candidates and select the best function based on the corresponding validation performance for the specific partitioning method used.

4.4 Efficiency Comparison of Transformers

Figure 6 illustrates the F1 score against inference speed (measured in documents processed per minute, or Doc/Min) for our WeSWin:RoBERTa and Longformer-2048. When comparing each model at the lightest partitioning baseline (SP), RoBERTa, with a speed of 11.69 Doc/Min, performs 110% faster than Longformer-2048, which processes at 5.56 Doc/Min. The difference in runtime becomes more pronounced in SS-k partitioning settings where RoBERTa performs up to 170% faster than Longformer-2048 (3.28 vs 1.22 Doc/Min) in SS-3. This demonstrates that RoBERTa serves as a competitive baseline to Longformer-2048 while being more than twice as efficient in inference. Notably, the smaller context size also makes the training process significantly faster and less GPU-intensive.

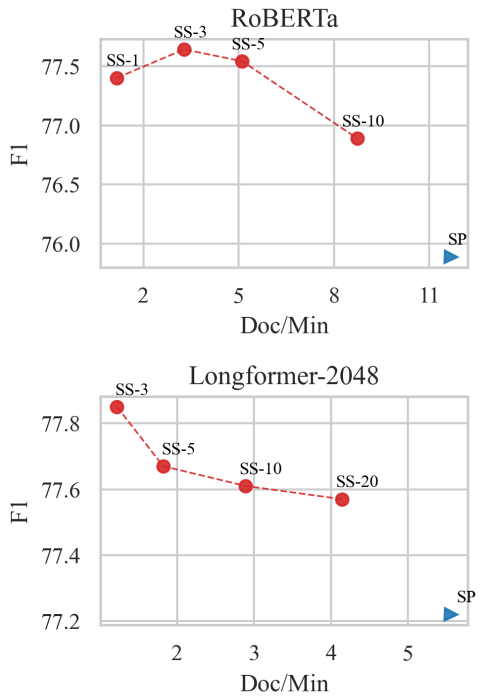

Figure 6: F1 score trend versus runtime efficiency (Doc/Min) across different partitioning overlaps for WeSWin:RoBERTa and Longformer-2048.

4.5 Segmentation on Industry Data

We apply and evaluate our segmentation model on AutoManual, a real-world dataset with human-labeled segments from an automotive user manual. AutoManual includes 12 chapters (9 for training, 1 for validation, and 2 for testing) and features longer, more diverse text than Wikipedia-based articles.

Given that the code or checkpoints for the baselines presented in Section 4.1 were not made publicly available, we employed two LLM-based chunking methods as baselines for our WeSWin:RoBERTa. Firstly, denoted as LLM-TextTiling, we extended the TextTiling (Hearst, 1997) algorithm to utilize LLM embeddings. Specifically, we used a k-sentence window on each side of a candidate break, applied max-pooling to the cosine similarities, and then used thresholding to identify topic shifts. As a second baseline, we implemented a modified version of LumberChun-

Model	AutoManual			AutoQA				
	F1	Prec	Rec	GPT Score	BERT Score	RougeL	Prompt Size (tokens)	Chunking Runtime (s)
LLM-TextTiling	34.57	27.58	46.32	5.32	51.05	41.30	254	153
MultiSent-LumberChunker	64.83	69.12	61.04	6.92	64.33	**50.86**	253	161
WeSWin:RoBERTa (**ours**)	**81.21**	**87.5**	**75.76**	**7.28**	**64.75**	49.23	245	**40.3**

Table 3: Comparison of WeSWin:RoBERTa against baselines in document segmentation using the AutoManual dataset and in downstream RAG using the AutoQA dataset.

ker (Duarte et al., 2024), denoted as MultiSent-LumberChunker, which performs prompt-based segmentation at the sentence level with multiple outputs—unlike the original LumberChunker, which operates at the paragraph level with a single output per iteration. For this, we utilize GPT-4o (OpenAI, 2023) to predict sentence IDs that mark the beginning of new topics or subtopics within an adaptive sliding window of k sentences. The prompt used in this baseline is included in Appendix A.3. Hyperparameters for each method were tuned on the validation set (detailed in Appendix A.2). Segmentation results presented in Table 3 demonstrate that WeSWin significantly outperforms both baselines in terms of F1 score, Precision, and Recall on the AutoManual dataset.

4.6 RAG Evaluation on Industry QA Data

We demonstrate the effectiveness of our segmentation model in the downstream task of question answering through the RAG framework. Specifically, we utilize our WeSWin:RoBERTa chunker to perform retrieval-augmented question answering on the AutoQA dataset, which comprises 50 human-labeled question and long-form answer pairs derived from the test set of AutoManual.

The retrieval process involves segmenting AutoManual chapters into chunks. Embedding vectors for these chunks are derived using OpenAI Embeddings (OpenAI, 2023). During inference, given a question, a FAISS search (Johnson et al., 2019) is conducted on the embedding vector against the chunk database to find the top chunk with the highest similarity. For answer generation, we create a prompt incorporating the input query and the retrieved chunk as context, and ask GPT-4o (OpenAI, 2023) to identify the span of text from the context that best answers the question. The prediction is then evaluated against the ground truth using three metrics: GPTScore (employing GPT-4o as a judge to provide matching scores out of 10), BERTScore (Zhang et al., 2020), and RougeL (Lin, 2004). RAG and GPTScore prompts are included in Appendix A.4.

In Table 3, we compare our WeSWin:RoBERTa chunker with the two previously introduced baselines on the AutoQA dataset. Operating under a comparable context size (or prompt size) setting, our model outperforms the best baseline by over 5% in GPTScore (7.28 vs 6.92) and operates four times faster in runtime (40.3 vs 161 seconds).[3] In terms of BERTScore and RougeL, WeSWin significantly surpasses LLM-TextTiling and performs comparably to MultiSent-LumberChunker. Furthermore, our RoBERTa-based model is considerably smaller than GPT-4 and incurs no monetary cost.

5 Conclusion

In this work, we introduced WeSWin, a sentence-level sequence labeling framework for document segmentation that is based on and compatible with various Transformer encoders, including BERT, RoBERTa, and Longformer. WeSWin employs a position-aware aggregation of sentence decisions from overlapping sliding windows to accurately predict topic shifts. We achieved state-of-the-art results with all three Transformer models on two public segmentation benchmarks. Additionally, when applied to an automotive user manual within a QA system for drivers, WeSWin significantly outperformed existing baselines on two proprietary datasets in both segmentation and RAG-based question-answering. Notably, our solution operates up to four times faster and is much more cost-effective compared to the baselines.

Acknowledgements

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Alliance Program, and by the Ontario Centre of Innovation through the TalentEdge Program.

[3] Retrieved context consists solely of a single chunk to maximize RAG efficiency, and the average chunk size is maintained consistently across models to ensure a fair comparison.

References

Sebastian Arnold, Rudolf Schneider, Philippe Cudré-Mauroux, Felix A. Gers, and Alexander Löser. 2019. Sector: A neural model for coherent topic segmentation and classification. *Transactions of the Association for Computational Linguistics*, 7:169–184.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

André V. Duarte, João Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L. Oliveira. 2024. Lumberchunker: Long-form narrative document segmentation. *Preprint*, arXiv:2406.17526.

Goran Glavas, Ananya Ganesh, and Swapna Somasundaran. 2021. Training and domain adaptation for supervised text segmentation. In *Workshop on Innovative Use of NLP for Building Educational Applications*.

Goran Glavas, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 125–130, Berlin, Germany. Association for Computational Linguistics.

Goran Glavas and Swapna Somasundaran. 2020. Two-level transformer and auxiliary coherence modeling for improved text segmentation. *CoRR*, abs/2001.00891.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Hakan Inan, Rashi Rungta, and Yashar Mehdad. 2022. Structured summarization: Unified text segmentation and segment labeling as a generation task. *ArXiv*, abs/2209.13759.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Omri Koshorek, 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)*, pages 469–473, New Orleans, Louisiana. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Michal Lukasik, Boris Dadachev, Kishore Papineni, and Gonccalo Simões. 2020. Text segmentation by cross segment attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4707–4716, Online. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. ArXiv preprint arXiv:2303.08774.

Hai Yu, Chong Deng, Qinglin Zhang, Jiaqing Liu, Qian Chen, and Wen Wang. 2023. Improving long document topic segmentation models with enhanced coherence modeling. In *Conference on Empirical Methods in Natural Language Processing*.

Qinglin Zhang, Qian Chen, Yali Li, Jiaqing Liu, and Wen Wang. 2021. Sequence model with self-adaptive sliding window for efficient spoken document segmentation. *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 411–418.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Appendix

A.1 Datasets

The characteristics of document segmentation datasets used in this work are shown in Table 4.

The Wiki-727k dataset, introduced by (Koshorek et al., 2018), comprises 727,746 Wikipedia documents segmented according to their table of contents. As a crucial dataset in the field of text segmentation, Wiki-727k provides a vast corpus for training and evaluating models, designed to overcome the limitations of previous datasets by offering a large, natural, and open-domain collection of documents with well-defined segmentation. Each document has been preprocessed to remove

Dataset	#Train	#Validation	#Test	#Segment/Doc	#Sentence/Doc	#Word/Doc
Wiki-727k	582,160	72,354	73,232	6.2	52.6	1,117.3
en_city	13,679	1,953	3,907	6.8	53.2	1058.3
en_disease	2,513	359	718	7.7	54.6	1122.8
AutoManual	9	1	2	112.8	755.9	9,797.0

Table 4: Cardinality and length statistics of text segmentation datasets. #Train, #Validation and #Test denote the number of documents in the training, validation, and test sets, respectively.

non-text elements such as images and tables and ensure that each segment is properly divided into sentences using the Punkt tokenizer (Bird et al., 2009).

The WikiSection dataset (Arnold et al., 2019) consists of segmented Wikipedia articles in domain-specific settings. Within this dataset, the en_city subset includes 19.5k articles about various city-related topics, while the en_disease subset contains 3.6k medical and health-related documents with scientific details from Wikipedia.

AutoManual is a proprietary dataset containing information on vehicle operation, maintenance, safety features, technical specifications, and troubleshooting, organized hierarchically into chapters, sections, subsections, paragraphs, and sub-elements such as text, list items, tables, and figures. The dataset is processed into flattened chapter texts, each human-labeled with segmentation information indicating the segment boundaries.

AutoQA consists of 50 human-labeled question and long-form answer pairs derived from the test set of AutoManual. Each answer is a multi-sentence span or paragraph that directly addresses the paired question, ensuring relevance and completeness.

A.2 Hyperparameter Setting

We train our Transformer baselines on Wiki-727k using a learning rate of 1e-5 for a maximum of three epochs, employing early stopping to prevent overfitting. For the en_city, en_disease, and AutoManual datasets, we adjust the learning rate to 5e-6 and extend training to five epochs, also utilizing early stopping. The batch size is set at 8 for BERT and RoBERTa models, and at 4 for Longformer baselines. The BERT and RoBERTa models are trained on a GTX 1080 Ti GPU, while the Longformer baselines utilize an RTX A6000.

Inference hyperparameters are set based on performance over the validation set. To determine the optimal document partitioning method, we test and compare the Single Prediction (SP) method as well as several SS-k methods, including $k = 6$, 4, and 2. When aggregating multiple predictions from overlapping sequences, we experiment with various weighting functions as hyperparameters—Uniform, Linear, Polynomial, and Loss-based—to find the best fit. In this context, we set ϵ to 0.1, k to either 5, 8, 10, or 12, and p to 2 for the Polynomial function. We also explore decision thresholds within the range of [0.3, 0.7] to derive binary sentence labels from the final output probabilities.

In the RAG experiment (Section 4.6), we employ SS-10 partitioning with $lob(.|k = 5, \epsilon = 0.1)$ as the weighting function for sentence aggregation using our WeSWin:RoBERTa. For MultiSent-LumberChunker, we set k to 50 sentences as the optimal sequence size for the sliding window provided to the LLM for prediction. For LLM-TextTiling, we optimized $k = 3$ sentences to the left and right, applying a max-pooling operation over their embeddings.

A.3 Prompt used in MultiSent-LumberChunker

The prompt used in MultiSent-LumberChunker to perform text segmentation is provided in Table 5.

A.4 Prompt used in RAG Inference and GPTScore

The prompt for the generation component of RAG is provided in Table 6, while the prompt used for deriving GPTScore in RAG evaluation is shown in Table 7.

You are an intelligent assistant. Your task is to predict the points within an input text document where the topic or subtopic undergoes a shift.

Details:
- The input consists of consecutive sentences, each provided in a new line in the format <ID>: <Sentence>.
- Your goal is to identify the IDs of sentences where a relatively distinct or even slightly different topic or subtopic begins, excluding the very first sentence of the document.

Additional Consideration:
- Avoid very long groups of sentences. Aim for a good balance between identifying the topic shifts and keeping groups manageable.

Output Format: A list of sentence IDs (only), each on a new line in document order.

<ID 1>: <Sentence 1>
<ID 2>: <Sentence 2>
...
<ID m>: <Sentence m>

Table 5: MultiSent-LumberChunker Prompt

You are an intelligent assistant. Your task is to answer the given question solely based on the information provided in the context.
Extract the span of sentences from the given context that most accurately and relevantly answers the given question. If no relevant answer can be derived from the context, respond with "Not found."

Question:
<question>

Context:
<context>

Table 6: RAG Prompt

You are an intelligent evaluator. Your task is to assess how well the candidate answer aligns with the provided ground-truth context while accurately addressing the question.
Focus on factual correctness strictly in relation to the given context.
Assign a score between 0 and 10, where 10 represents a perfect answer.
Do not provide explanations—only the score.

Question:
<question>

Context:
<answer>

Candidate Answer:
<predicted answer>

Table 7: GPTScore Prompt