Neural Document Segmentation Using Weighted Sliding Windows with Transformer Encoders

Saeed Abbasi

A Thesis Submitted to the Faculty of Graduate Studies

In Partial Fulfillment of the Requirements

for the Degree of Master of Science

Graduate Program in

Electrical Engineering and Computer Science

York University

Toronto, Ontario

August 2024

Abstract

The subdivision of documents into semantically coherent segments is a fundamental challenge in Natural Language Processing (NLP), with notable applications in information retrieval and question answering. Effective text segmentation is crucial for enhancing Retrieval-Augmented Generation (RAG) systems by providing coherent segments that improve the contextual accuracy of responses. We introduce a weighted sliding window framework, WeSWin, that effectively segments arbitrarily long documents using Transformers. WeSWin consists of overlapping document partitioning followed by the weighted aggregation of multiple sentence predictions within the generated windows, ensuring that sentences with larger context visibility contribute more to the ultimate label. Additionally, we propose a multi-task training framework, WeSWin-Ret, which combines text segmentation with an auxiliary sentence retrieval task. This approach injects query-awareness into the embedding space of the shared Transformer, resulting in improved segmentation performance. Extensive experiments demonstrate that our methods outperform state-of-the-art approaches. On the Wiki-727k benchmark, both our WeSWin and WeSWin-Ret models surpass existing works based on BERT, RoBERTa, or LongFormer Transformers. Notably, our RoBERTa baseline often matches LongFormer's performance while being significantly more efficient in training and inference. We validate our model's robustness on domain-specific segmentation benchmarks, including en_city, en_disease, and an industrial automotive dataset, demonstrating generalizability across domains. Lastly, our model proves to be highly effective in enhancing downstream RAG applications by providing cohesive chunks for knowledge retrieval.

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my supervisor, Prof. Aijun An, for all her support, guidance, and encouragement throughout the course of my research. Her profound knowledge and invaluable insights have been essential in shaping this thesis.

I would also like to extend my thanks to my co-supervisor, Prof. Heidar Davoudi, for their continuous support and valuable feedback. Their expertise and thoughtful suggestions have greatly contributed to the improvement and refinement of my work. I am grateful to the members of my thesis committee, Prof. Xiaohui Yu and Prof. Manos Papagelis, for taking time out of their busy schedules to review my research and serve on my examination committee.

I want to acknowledge my family for their love and support. To my parents, thank you for believing in me and supporting my dreams. I am especially thankful to my wife, for her constant support, love, and encouragement throughout this journey. Finally, I want to thank my friends, for their companionship and moral support. Your presence has made this journey more enjoyable.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the evolving landscape of Natural Language Processing (NLP), the subdivision of documents into discrete, semantically coherent segments represents a fundamental challenge with a wide range of applications. Text segmentation aims to dismantle extensive texts into manageable, meaningful blocks, each signifying a relatively distinct topic, subtopic, or theme. Document segmentation directly facilitates various applications such as document summarization, content organization, and, notably, information retrieval and question answering. Despite its widespread application, the challenge lies in the variability of text structures, themes, and topical granularity size which can significantly differ across domains. The evolution of text segmentation techniques reflects the increasing complexity and subtlety required in understanding and parsing various topical regions within text.

1.1 Background and Motivation

Retrieval-Augmented Generation (RAG) is a widely used technique that enhances Large Language Models (LLMs) by incorporating relevant external information into their generation process. This integration results in more accurate, contextually appropriate, and up-to-date responses. A crucial component affecting RAG's effectiveness is text segmentation, which involves dividing documents into coherent segments that can be retrieved and used in generative prompts for LLMs.

Recent works on text/document segmentation have evolved significantly from early rule-based and statistical methods to sophisticated deep learning techniques. Unsupervised models initially focused on lexical overlaps and semantic relatedness to infer segment boundaries. TextTiling (Hearst, 1997) pioneered the approach by analyzing shifts in lexical co-occurrence patterns, while GraphSeg (Glavas et al., 2016) utilized semantic relatedness graphs to identify segments. Enhancements to these models came with the incorporation of embeddings from GloVe and BERT to compute semantic similarities (Song et al., 2016; Xu et al., 2020).

In supervised settings, methods transitioned to treating segmentation as a sequence labeling task, with models like BiLSTM and hierarchical BiLSTM trained on large datasets like Wiki-727K (Koshorek et al., 2018). Modern transformer-based architectures offer dense, context-aware text representations that capture subtle semantic nuances. For instance, cross-segment BERT and hierarchical BERT+BiLSTM provide improved document segmentation using contextual embeddings (Lukasik et al., 2020). Sliding-window document processing methods, such as Glavas et al. (2021), Gong et al. (2020), and Zhang et al. (2021) have been introduced to handle the limitations of fixed-length input constraints in transformer models

like BERT and RoBERTa. Multi-task training approaches have also been explored to improve text segmentation by combining it with related tasks such as coherence modeling or extractive summarization (Glavas and Somasundaran, 2020; Cho et al., 2022; Yu et al., 2023).

Despite their advantages, these techniques face limitations in effectively using context and may suffer from costly training or inference. Our proposed method addresses these challenges by improving the state-of-the-art benchmarks across multiple transformer models.

1.2 Contributions

In this work, we propose an overlapping sliding window framework, denoted as WeSWin, to process long documents in transformer-based text segmentation models. Documents are converted into overlapping sequences that are limited in size according to the transformer's sequence length. In this context, several partitioning methods have been explored, including intersected and strided sentence overlap. Sufficient context is crucial for accurately predicting topic shifts in a document. Our method effectively enlarges the overall context window for individual sentences, removing the need for training costlier models with larger context sizes, which have significantly less efficient inference.

During inference, we aggregate multiple sentence predictions across overlapping windows. For effective aggregation, we propose several weighting functions to adjust the contribution of multiple sentence predictions in the final decision. These weight assignment functions amplify the contribution of sentence positions with access to richer context in a sequence. Boundary sentence positions are penalized due to their lack of right or left context.

Additionally, we develop a multi-task training framework, denoted as WeSWin-Ret, that simultaneously trains text segmentation alongside an auxiliary sentence retrieval task using a shared transformer model. Sentence retrieval involves predicting relevant sentences of a document in response to a given query. The retrieval module can enhance sentence representations for segmentation by forming coherent topic clusters in the shared embedding space.

We conduct comprehensive experiments using publicly available datasets in both general and domain-specific settings. We demonstrate that the proposed method consistently outperforms existing state-of-the-art approaches while using more efficient Transformers with smaller context size. In addition to direct segmentation benchmarks, we further evaluate our segmentation model by assessing its performance in a downstream RAG system through retrieval-based question answering.

Finally, we evaluate our segmentation model on a real-world proprietary dataset, which contains significantly longer documents with a more complex hierarchy of content compared to existing public datasets. We demonstrate that our model can generalize to new domains, especially through fine-tuning, making it a suitable semantic chunker for industrial RAG-based systems.

The main contributions of our work can be summarized as follows:

- We propose an overlapping sliding window framework, denoted as WeSWin, for processing long documents in transformer-based text segmentation models.
- We explore and evaluate four document partitioning methods at different accuracy-runtime trade-offs.
- We develop three weighting functions for effectively aggregating sentence predictions

during inference.

- A multi-task training framework, denoted as WeSWin-Ret, is introduced, combining text segmentation with an auxiliary sentence retrieval task.
- We perform comprehensive experiments demonstrate our method's superiority over state-of-the-art approaches and its efficiency in training and inference.
- We assess our model's performance in a downstream RAG system through retrieval-based question answering.
- We evaluate the model on a real-world dataset, showing its ability to generalize to new domains.

1.3 Overview of the thesis

The rest of the thesis is organized as follows. In Chapter 2, we look at a variety of related work in the literature regarding document segmentation and its applications. Chapter 3 formally defines document segmentation as a sequence labeling task. The specifics of our methodology, including our contributions to the field, are detailed in Chapter 4. An extensive set of experiments that thoroughly evaluate the performance of our proposed models using various datasets is presented in Chapter 5. Chapter 6 demonstrates the practical application of our model in real-world domain settings. Finally, we conclude the thesis and discuss future works in Chapter 7.

Chapter 2

Literature Review

2.1 Text Segmentation

Topic segmentation has evolved from rule-based and statistical methods to sophisticated deep learning techniques.

Unsupervised methods. Early unsupervised models utilize methods such as lexical overlaps (Hearst, 1997) or semantic relatedness graphs (Glavas et al., 2016) to measure the lexical or semantic cohesion between sentences and infer segment boundaries.

One of the leading works in this category, TextTiling (Hearst, 1997) proposed to segment texts into multi-paragraph subtopic passages by calculating coherence scores between consecutive utterance pairs. This process begins with tokenizing the text into equal-sized segments, followed by assessing the lexical similarity between adjacent blocks to generate a

sequence of coherence scores that denote topical relatedness. By analyzing shifts in lexical co-occurrence patterns, TextTiling identifies potential boundaries between subtopics. More specifically, it computes ”depth scores” by considering the highest coherence scores on either side of an interval. Pairs of utterances with depth scores exceeding a statistical threshold are then selected to mark segment boundaries.

As another unsupervised work, Glavas et al. (2016) presents an unsupervised text segmentation algorithm, GraphSeg, which leverages word embeddings and semantic relatedness measures to construct a graph of the document where nodes represent sentences. Semantic coherence is identified through maximal cliques within this graph, determining the text segments. Evaluation on synthetic and real-world datasets showed competitive performance, compared to state-of-the-art topic modeling-based methods at the time.

Later, Song et al. (2016) and Xu et al. (2020) enhanced TextTiling (Hearst, 1997) by respectively utilizing embeddings from GloVe and pretrained BERT, to compute semantic similarity for consecutive text pairs. More specifically, Song et al. (2016) addresses the problem of session segmentation in open-domain dialogue systems by proposing an embedding-enhanced TextTiling approach. The authors enhance the traditional TextTiling technique by incorporating modern embedding-based similarity measures and introduce a tailored method for word embedding learning, combining queries and their corresponding replies into ”virtual sentences” to model interactions more effectively. Furthermore, Xu et al. (2020) focuses on addressing the challenge of topic shifts within multi-turn dialogues in retrieval-based systems. The authors proposed an unsupervised topic-aware segmentation algorithm to identify and segment dialogue into topic-relevant units. They employ a dual cross-attention mechanism to match each topic segment with potential responses, aiming to accurately track and adapt to the natural topic shifts occurring in conversations. They evaluated their model on

three public datasets, with notable improvements on the E-commerce dataset.

Supervised methods. In supervised settings, many works tackle topic segmentation as a sequence labeling task, building neural models to directly predict segment boundaries.

Wang et al. (2016) incorporates a bidirectional long short-term memory (BiLSTM) model alongside a convolutional neural network (CNN) for effective paragraph representation. They introduce an algorithm based on frequent subsequence mining for automatic cue phrase identification, targeting improved topic segmentation accuracy in web documents. Results indicate that the BLSTM-CNN model, augmented with auto-identified cue phrases, outperforms traditional methods such as TextTiling and Conditional Random Fields (CRF).

Later, Koshorek et al. (2018) proposed a supervised hierarchical BiLSTM model for coarse-grained topic segmentation trained over their automatically labeled dataset. They introduce a large dataset, WIKI-727K, derived from English Wikipedia, which they utilize to train a hierarchical neural model. This model consists of a lower-level bidirectional LSTM for sentence representation and a higher-level LSTM for segmenting texts based on sentence representations. Their experiments demonstrate superior performance of their model particularly on the Wikipedia-based datasets, highlighting the importance of large-scale, domain-varied training data.

In Lukasik et al. (2020), the authors propose three transformer-based architectures for document and discourse segmentation, challenging the traditional reliance on RNNs by employing BERT-style contextual embeddings. They proposed a cross-segment BERT model that uses only local context around candidate breaks, as well as two hierarchical models, BERT+BiLSTM and hierarchical BERT, all of which improved the state-of-the-art results.

The experiments underscore the importance of context size and model architecture in segmentation tasks, leading to insights that local context can often suffice for high performance.

Among sliding-window document processing methods, Glavas et al. (2021) proposed HITS, a hierarchical text segmentation approach leveraging RoBERTa for token-level embeddings and an upper transformer for sentence contextualization. Notably, HITS integrates adapter-based fine-tuning to prevent overfitting to the training domain and maintain general linguistic knowledge, improving domain-transfer performance. They utilized sliding window sequences and binary cross-entropy loss for training. Their results demonstrated that HITS achieved state-of-the-art performance on Wikipedia-based datasets.

Gong et al. (2020) presents Recurrent Chunking Mechanisms for Machine Reading Comprehension on long texts, overcoming fixed-length input constraints of models like BERT. It utilizes reinforcement learning for flexible text segmentation and recurrent mechanisms to maintain context across segments. Their approach shows performance improvements on CoQA, QuAC, and TriviaQA datasets compared to standard BERT-based models, particularly in handling longer documents.

Tackling spoken document segmentation, the proposed method in Zhang et al. (2021), utilizes a sentence-level sequence labeling framework, equipped with a self-adaptive sliding window technique, which dynamically adjusts the processing window based on prior segmentation decisions to improve inference efficiency. They achieved state-of-the-art performance on Wiki-727k using BERT and RoBERTa transformers.

In an attempt to predict rhetorical roles in legal documents, Belfathi et al. (2023) used pre-trained language models like BERT augmented with sentence position information. The

authors proposed two architectures for processing longer documents: BERT-SentPos and Hi-BERT. BERT-SentPos integrates sentence position embeddings directly into the BERT input to reflect the sentence's role based on its position in the document. HiBERT builds on this by adding a hierarchical structure that contextualizes sentences within their surrounding text.

Among multi-task training attempts, Glavas and Somasundaran (2020) introduced Coherence-Aware Text Segmentation (CATS), integrating text segmentation with explicit coherence modeling. Utilizing two hierarchically connected Transformer networks, CATS operates on a multi-task learning framework that combines sentence-level segmentation prediction with an auxiliary task that enhances text coherence by distinguishing original text snippets as more coherent from corrupted (shuffled or replaced) sentence sequences.

Yu et al. (2023) formulates topic segmentation as a sentence-level sequence labeling task, which predicts binary labels corresponding to each sentence in a document. The paper introduces two novel modules: Topic-aware Sentence Structure Prediction (TSSP) and Contrastive Semantic Similarity Learning (CSSL), aiming to enhance the model's ability to understand and predict topic changes within documents. The TSSP module helps the model learn inter-sentence coherence, while CSSL focuses on improving semantic similarity between sentences belonging to the same topic. Notably, they achieved state-of-the-art performance on Wiki-727k and WikiSection's en_city and en_disease datasets using a Longformer-based transformer model.

Inan et al. (2022) introduces a method that addresses text segmentation and topic labeling simultaneously through a generation-based approach. The methodology involves training a pre-trained encoder-decoder model such as BART or T5 to predict both segment boundaries and corresponding topic labels in a unified sequence. They conducted experiments on

datasets including Wiki-727K, WikiSection, and QMSum, achieving state-of-the-art performance, particularly in scenarios with limited training data.

Cho et al. (2022) introduces an extractive summarization system named Lodoss, unifying section segmentation and summarization into a single optimization framework based on Longformer model. Evaluated on datasets containing scientific articles and lecture transcripts, their model builds sentence representations by simultaneously performing section segmentation and summarization tasks.

Lee et al. (2023) tackles topic segmentation by proposing a siamese sentence embedding approach, allowing two input sentences to be processed independently. Evaluated on WikiSection dataset, their model employs multi-task objectives, namely Same Topic Prediction (STP), Topic Classification (TC), and Next Sentence Prediction (NSP) to improve segmentation performance.

Xia and Wang (2023) introduces a Sequence-to-Sequence model with Mixed Pointers (Seq2Seq-MP) aimed at improving topic segmentation and labeling. It employs a shared sentence-topic encoder leveraging BERT for contextual embeddings, and a joint decoder for simultaneous generation of segment boundaries and topics, ensuring coherence and logical text flow.

In the following two sections, we review the recent works in Dense Retrieval and Retrieval Augmented Language Modeling as direct applications of text segmentation.

2.2 Dense Retrieval

Dense retrieval has emerged as a pivotal technique in retrieval-based systems, leveraging dense vector representations of text to enable efficient and effective search. Dense retrieval is intimately linked to the prerequisite task of text segmentation, which involves dividing text into meaningful units such as passages or paragraphs. Text segmentation provides the necessary granularity and context for dense retrievers to operate effectively, making it a foundational aspect of dense retrieval systems.

The success of dense text similarity and retrieval is often reliant on access to extensive training datasets. In many domain-specific applications, annotated data is scarce, necessitating the use of existing annotated datasets for initial training, followed by fine-tuning or zero-shot transfer learning to new domains. The process of representation learning, which is central to an effective passage retrieval, shares similarities with Semantic Text Similarity (STS) tasks, albeit with differences in their symmetric (sentence-sentence) and asymmetric (query-passage) natures. Techniques and models developed for STS, may often be used for constructing retrieval embeddings too.

In this regard, Sentence-T5 (Ni et al., 2022a) proposes a multi-stage contrastive learning recipe involving fine-tuning first on semi-structured web-mined corpora and then on Natural Language Inference (NLI) datasets. With that said, we particularly discuss a few recent works in STS. Similarly, GTR (Ni et al., 2022b) combines pre-training with generic data and fine-tuning with MS Marco (Campos et al., 2016) under contrastive settings. E5 (Wang et al., 2022) models are contrastively trained on CCPairs, a curated web-scale text pair dataset containing CommunityQA, Common Crawl and Scientific papers. Recently, Instructor (Su et al.,

2023) proposed instruction-based fine-tuning over MEDI, a collection of 330 text embedding datasets, enabling task- and domain-aware embedding generation.

In addition to supervised methods, unsupervised or self-supervised approaches have been employed to approximate retrieval tasks, offering alternatives when annotated data is lacking. Chang et al. (2020) proposes inverse cloze task (ICT), where a sentence span in a passage is randomly chosen and paired with the rest of the passage as a positive sample. Contriever (Izacard et al., 2022a) proposes independent cropping, which involves sampling independently two spans from a document to form a positive pair, in addition to in-batch (following Chen et al. (2017), Karpukhin et al. (2020)) and cross-batch negative sampling. Neelakantan et al. (2022) use neighboring pieces of text on the Internet as positive pairs as well as in-batch negatives to do contrastive pre-training. Moreover, Oguz et al. (2022) performs domain-matched pretraining over synthetic data to improve in-domain results, and SPAR (Chen et al., 2022) trains a dense retriever by a sparse teacher retriever, BM25.

Among works tackling information retrieval using document segmentation, Tiedemann and Mur (2008) compared ways of dividing documents into passages including TextTiling (Hearst, 1997) algorithm and coreference chains for passage boundary detection. According to them, simple window-based techniques outperformed semantically motivated ones, at least in a newspaper corpus, which is contrary to our findings in other domains. According to Wartena (2013), video passage retrieval performs best with fixed length (consecutive or overlapping) chunking or when lexically coherent segments is used. In latter case, segments are derived from a minimum cut model based on sentence similarity of the two sides of a cut. Shtekh et al. (2018) used a TextTiling (Hearst, 1997) based algorithm, equipped with different embedding methods, to show the quality of document-level information retrieval, evaluated on arXiv preprints, was improved using semantic document segmentation.

2.3 Retrieval Augmented Language Modeling

Retrieval Augmented Language Modeling (RALM) represent significant advancements in leveraging external knowledge to enhance language models. These techniques hinge on the ability to retrieve relevant text segments from vast external sources, grounding the language model's generation process in contextualized and accurate information. Text segmentation, therefore, becomes a foundational task, as it dictates how information is chunked and accessed during the retrieval process. By effectively segmenting text into meaningful units, retrieval-augmented models can dynamically incorporate pertinent information, thereby improving their performance in various applications, such as open-domain question answering and generative tasks. The following paragraph delves into the diverse architectures and approaches within RALM.

RALM aims to ground the LM during generation by conditioning on relevant documents retrieved from an external knowledge source. Retrieval-based language models come with different architectures depending on retrieval chunk granularity (e.g., passage chunks, tokens, or entities) and frequency of retrieval (e.g., once per query, every n tokens, or every entity mention), among others. Realm (Guu et al., 2020) augments Masked Language Modeling (MLM) pretraining with a dense retriever to search over and find most relevant document chunks from Wikipedia to a given query, and the end-to-end model is jointly fine-tuned for open-domain QA task. DPR (Karpukhin et al., 2020) proposed pipeline training of a dual-encoder retriever with a following open-domain QA objective. RAG (Lewis et al., 2020) proposed end-to-end finetuning of a pretrained retriever with a pretrained seq2seq model to tackle open-domain generative QA task. Atlas (Izacard et al., 2022b) jointly pretrain a retrieval-augmented seq2seq language model, comprised of a passage-based dense retriever

and an encoder-decoder language model. In-Context RALM (Ram et al., 2023) and REPLUG (Shi et al., 2023) employ tuneable (or frozen) retrievers to augment black-box language models by prepending the retrieved content into the input context, with retrieval frequency of every n tokens. RETRO (Borgeaud et al., 2021) combines a frozen dual-tower BERT retriever and a differentiable encoder with a chunked cross-attention mechanism in intermediate layers to predict tokens based on a massive token dataset. Operating on token level, kNN-LM (Khandelwal et al., 2020) extends a pretrained LM by linearly interpolating its next word distribution with a k-nearest neighbors (kNN) model, based on distance in the embedding space of tokens. To achieve more retrieval efficiency, FLARE (Jiang et al., 2023), He et al. (2021) and Alon et al. (2022) proposed adaptive retrieval of chunks or tokens.

Particularly, for passage or chunk-level retrieval performed per input query, any embedding model tailored for semantic similarity, including Sentence Transformers (Reimers and Gurevych, 2019) and OpenAI Embeddings (OpenAI, 2023), may be used to create an embedding dataset over extracted chunks. An input query is embedded using the same model and its representation is compared against the stored chunks of knowledge through a search algorithm, such as FAISS (Johnson et al., 2019), to derive the top matching chunks. Once retrieval is done, a generative model such as ChatGPT (OpenAI, 2023) may be used to synthesize the answer from the input prompt, consisting of the input query alongside the retrieved context information. As an indirect way of evaluation, we will later use this RAG setup to test our proposed segmentation model.

Chapter 3

Problem Statement

In this work, we tackle *Document Segmentation* by framing it as a supervised sentence-level sequence labeling task.

Given a document $D = \langle s_1, s_2, ..., s_n \rangle$ made of n sentences, text segmentation aims to divide D into semantically cohesive segments or chunks, $C = \langle c_1, c_2, ..., c_{|C|} \rangle$. Segments are non-overlapping, ordered subsequences or blocks of sentences within the document. To this end, we cast the problem into a sentence-level binary classification task to predict topic shift probabilities p_i for each sentence $s_i \in D$. The objective is to predict whether sentence s_i should be labeled as the *end of a segment* if its immediate subsequent sentence s_{i+1} (along with its subsequent sentences) is semantically different enough to start a new segment. A positive prediction for sentence s_i indicates that it is the concluding sentence of a segment, while a negative prediction indicates otherwise.

Table 3.1 provides an overview of a sample document constructed from a Wikipedia arti-

cle sourced from the Wiki-727k dataset (Koshorek et al., 2018), which will be introduced in detail later in Subsection 5.1.1. Each document is formatted as a list of sentences accompanied by their associated binary labels. A positive label for a sentence indicates that it marks the end of the segment it belongs to. This particular article, titled "Downtown Toronto"[1], consists of 52 sentences and 6 segments, each discussing a different topic or subtopic within the document. Although section or segment titles are displayed in this table, they are neither utilized in the training phase nor during inference of our models. In other words, the determination of a topic shift is based solely on the semantics of the sentences (excluding any titles).

[1] https://en.wikipedia.org/wiki/Downtown_Toronto

Section	Index	Label	Sentence
preface.	1	0	Downtown Toronto is the city centre and main central business district of Toronto, Ontario, Canada.
	2	0	Located entirely within the district of Old Toronto, it is approximately bounded by Bloor Street to the north, Lake Ontario to the south, the Don River to the east, and Dufferin Street to the west.
	3	0	It is also the governmental centre of the City of Toronto and the Province of Ontario.
	4	1	The area is made up of the city's largest concentration of skyscrapers and businesses and form its skyline, which has the third most skyscrapers in North America exceeding in height, behind New York City and Chicago.
Neighbourhoods.	5	0	The Financial District, centred on the intersection of Bay Street and King Street is the centre of Canada's financial industry.
	...	...	...
	34	0	The PATH Underground, which is an extensive network of tunnels connecting the buildings of the area, helps take people from off the streets, especially during the winter months.
	35	1	Among the important government headquarters there is the Ontario Legislature, and the Toronto City Hall.
Architecture.	36	0	In the 1970s, Toronto experienced major economic growth and surpassed Montreal to become the largest city in Canada.
	...	...	...
	41	0	The CN Tower, once the tallest free-standing structure in the world, remains the tallest such structure in the Americas, standing at 553.33 metres (1,815 ft., 5 inches).
	42	1	Other notable buildings include Scotia Plaza, TD Centre, Commerce Court, the Royal Bank Plaza, The Bay's flagship store, and the Fairmont Royal York Hotel.
Education.	43	0	The University of Toronto, established in 1827, is the oldest university in the province of Ontario and one of the world's leading public research institutions.
	...	...	...
	46	1	There are many public schools of the Toronto District School Board in downtown Toronto.
Retail.	47	0	Downtown Toronto is home to the flagship department stores of The Bay and Sears Canada (formerly Eaton's).
	48	1	The Toronto Eaton Centre, a large, multilevel enclosed shopping mall and office complex that spans several blocks and houses 330 stores, is the city's top tourist attraction with over one million visitors weekly.
Transportation.	49	0	The most famous and busiest road in Toronto is Yonge Street, which begins at the end of Lake Ontario and runs through downtown, continuing north all the way to the city of Barrie, Ontario.
	...	...	...
	52	1	Nearby airports include Billy Bishop Toronto City Airport, which is adjacent to downtown, and the much larger Toronto Pearson International Airport located 27 km to the northwest.

Table 3.1: Sample document from Wiki-727k dataset titled as "Downtown Toronto".

Chapter 4

Methodology

We approach document segmentation as a sentence-level sequence labeling problem, classifying each sentence as either the end of a semantic segment or not. In this regard, we fine-tune and evaluate several pretrained transformer encoders for document segmentation including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020). For documents exceeding the transformer's sequence length, we partition them into appropriately-sized sequences both for training and inference. More specifically, we propose an end-to-end transformer-based framework for document segmentation equipped with several sliding-window document partitioning and potential sentence aggregation techniques. We explain the *training* and *inference* procedure of our proposed model, WeSWin, in Section 4.1 and Section 4.2 respectively. Finally, to enhance the performance of our proposed framework, we design a multi-task training approach in Section 4.3 to jointly train document segmentation with *sentence retrieval* as an auxiliary task using a shared transformer encoder.

4.1 WeSWin Model Training

Tackling document segmentation as a sentence-level sequence labeling problem involves classifying each sentence to determine whether it signifies a semantic shift in topic. Among the three pre-trained transformer encoders, BERT and RoBERTa have maximum sequence lengths of 512 tokens, while Longformer can process up to 4096 tokens as input. Due to the limited input size of transformer encoder models, longer documents typically cannot fit into a single input sequence after tokenization, especially for the smaller-sized BERT and RoBERTa models. Therefore, documents longer than the model sequence length need to be partitioned into appropriately-sized sequences to form either training or inference examples. At the expense of slower training and inference, Longformer offers larger context visibility, leading to more informed predictions compared to BERT or RoBERTa transformers. Interestingly, higher context visibility does not necessarily translate into much higher segmentation accuracy beyond a certain point in context size. Consequently, it is crucial to efficiently utilize the available sequence length of transformers while exploring the optimal context length required for document segmentation. In the next part, we discuss the document partitioning method used for generating training examples.

Training Sequence Formation. Given a document $D = \langle s_1, s_2, ..., s_n \rangle$ consisting of n sentences, and their respective binary class labels $\langle y_1, y_2, ..., y_n \rangle$, $y_i = 1$ 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. The first sequence is formed by including as many sentences as possible to fill up the input window of the transformer. The second sequence is formed by starting with the last sentence in the previous sequence and

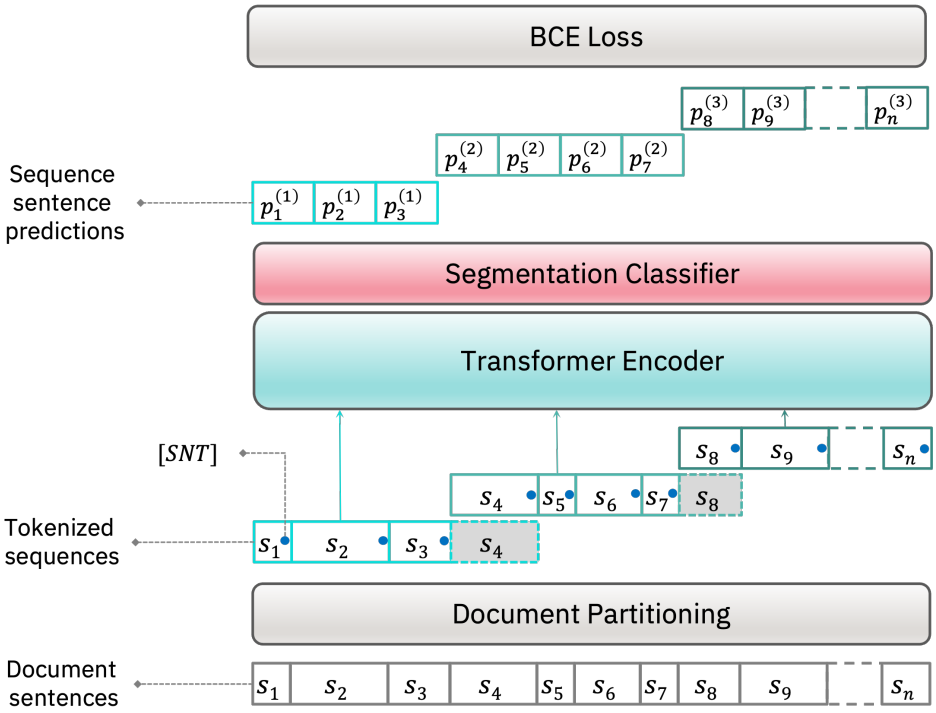

Figure 4.1: Training pipeline of our WeSWin model. Input sequences are formed using CR-1 document partitioning method. Each tokenized sequence is fed to the transformer encoder followed by the segmentation classifier to output topic shift probabilities $p_i^{(j)}$ for individual sentences s_i, derived from contextualized embedding of the corresponding $[SNT]$ tokens. BCE loss is used to train the model end-to-end.

including as many subsequent sentences as possible to maximally utilize the transformer's input size. This process continues until all the sentences in the document are covered. In addition to training, this sequence formation setting is used as an inference baseline, referred to as CR-1 in the experiments section.

Inspired by previous works (Glavas and Somasundaran, 2020; Zhang et al., 2021; Yu et al., 2023), we introduce a special token, $[SNT]$, to append at the end of each sentence to encode the segmentation information. A sequence $Seq_{i_s:i_e} = \{s_i\}_{i \in i_s:i_e}$, spanning document D

from sentence index i_s to i_e, is tokenized as follows,

$$Seq_{i_s:i_e}^{(T)} = \langle [CLS], s_{i_s}, ..., [SNT], s_{i_{s+1}}, ..., [SNT], ..., s_{i_e}, [SNT], [EOS] \rangle \tag{4.1}$$

where s_i is the ith sentence in the sequence and is tokenized as $t_{i,1}, t_{i,2}, ..., t_{i,|s_i|}$ (where $|s_i|$ is the number of tokens in s_i). $[CLS]$ and $[EOS]$ special tokens mark the beginning and end of the sequence respectively. Sequences are subject to a length constraint in terms of number of underlying tokens, i.e., $|Seq_{i_s:i_e}^{(T)}| \leq T$, where T is the maximum sequence length of the transformer. Note that the start and end sentences of the tokenized sequences are dynamically set in the runtime based on the length of the underlying sentences in the document. The only fixed sequence boundaries are the first sentence of the first generated sequence (i.e., s_1) and the last sentence of the last generated sequence (i.e., s_n).

Training Pipeline. Given the training sequences (obtained from CR-1 document partitioning method) along with their sentence labels, we fine-tune a pretrained transformer encoder model with a segmentation classification head. As illustrated in Figure 4.1, the transformer takes a sequence as input and computes the contextual representations of its individual tokens (Equation 4.2). The constructed embedding of each $[SNT]$ token is then fed into the segmentation head, which is a binary Softmax classifier (Equation 4.3) responsible for predicting topic shift probabilities for the corresponding $[SNT]$ positions as sentence representatives.

$$\{h_j\}_{t_j \in Seq_{i_s:i_e}^{(T)}} = Transformer\left(Seq_{i_s:i_e}^{(T)}\right) \tag{4.2}$$

$$\forall h_i \in \{h_i\}_{i_s \leq i \leq i_e} = \{h_j\}_{t_j = [SNT]}, \quad p_i = Classifier(h_i) \tag{4.3}$$

Finally, we use the standard binary cross-entropy loss over all sequence sentences in a training batch to train the model (Equation 4.4).

$$L_{BCE} = -\sum_{Batch}\sum_{i \in Seq_{is:ie}} y_i log(p_i) + (1-y_i) log(1-p_i) \tag{4.4}$$

It is worth noting that in CR-1 document partitioning, the last (i.e., rightmost) sentence of a sequence is not included in loss calculation (except when it is the last sentence in the concluding sequence of the document). The reason is that the last sentence of a sequence may not have necessary right context needed for accurate topic shift detection. In fact, its label is used in the training with the next sequence, and it only appears as Context at the Rightmost position of the current sequence, hence the name CR-1. In the next section, we will introduce more document partitioning methods for inference.

4.2 WeSWin Model Inference

Transformers are often constrained by a maximum context length, which poses a challenge when processing long documents that typically exceed this limit. A simple exclusive sequence generation will suffer from lack of context when predicting for boundary sentences in sequences due to abrupt truncation of context. Therefore, we have explored several sliding-window strategies for partitioning documents into overlapping sequences to alleviate the context limitation with a controllable degree of predictive or context-only repetition. More specifically, we introduce *context-only* and *multi-prediction* document partitioning methods to construct sentence-overlapped sequences subject to the transformer sequence length.

4.2.1 Context-only Document Partitioning

In this approach, selected sentences at the beginning and/or end of sequences are reserved exclusively to provide contextual support, while not actively participating in prediction task. This strategy ensures contextual support for near-boundary sentences within sequences while still producing single (as opposed to multiple) predictions for individual sentences. Two variations of this method are explored in this work, as elaborated below.

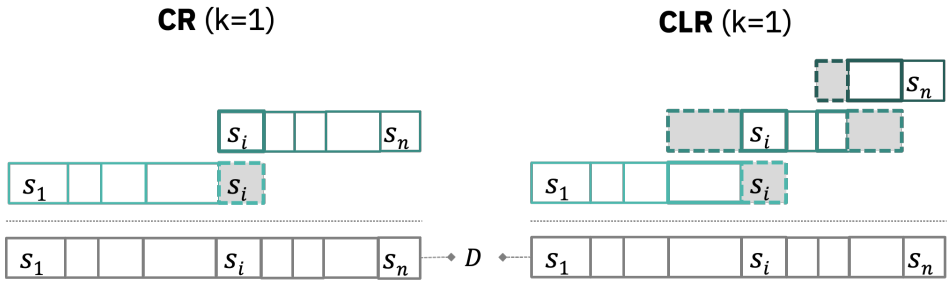

Figure 4.2: Sample context-only document partitioning methods: CR-1 and CLR-1. The sole purpose of sentences with gray background in the sequence is to provide additional context to the model when predicting the segmentation label of active sentences, shown with white background.

Context-only Overlap of k Right-most Sentences (CR-k) In this configuration, the last (i.e., rightmost) k sentences of a sequence are designated solely as context, excluded from prediction and loss calculation. If k exceeds the capacity of the sequence to provide right context, only the leftmost sentence remains active; the remaining sentences are reserved as context. This approach ensures that each sentence in the document is active for prediction only and exactly once across all derived sequences. Subsequent sequences initiate immediately following the final active sentence from the preceding sequence. Notably, the concluding sequence of a document consists entirely of active sentences, due to the absence of trailing context. From this category, CR-1 partitioning is used in the training phase of our models, as introduced in Section 4.1. CR-1 is also used in the inference as our lightest par-

titioning method in terms of runtime efficiency. Sequence generation with CR-1 partitioning method is illustrated in Figure 4.2.

Context-only Overlap of k Left-most and k Right-most Sentences (CLR-k) In this setting, we reserve the first and last k sentences of each sequence exclusively for supporting context, enabling prediction solely for the central, active sentences. If the total disabled span, $2k$, exceeds the sequence's capacity to provide context on both sides, only one central sentence is kept active, with an equal or nearly equal number of support sentences on each side. In any case, this strategy ensures that every sentence in the document is active for prediction only and exactly once. Sequence generation with CLR-1 document partitioning method is illustrated in Figure 4.2.

By adjusting the number of context-designated sentences, these partitioning strategies have been shown to enhance segmentation accuracy with a reasonable trade-off in efficiency.

4.2.2 Multi-Prediction Document Partitioning

To maximize the use of context in inference, we propose sliding the sequence window over documents in a non-exclusive manner, allowing for a desired degree of context overlap between consecutive sequences. This method allows multiple predictions for individual document sentences, and at the same time effectively increases the overall length of the context considered for making an aggregated decision for each sentence. When combined with a multi-sentence aggregation module, we refer to our proposed method as *Weighted Sliding Window Segmentation (WeSWin)*.

Given a document to be segmented, we create overlapping sequences of consecutive sentences to populate the transformer's sequence window. In this regard, we have explored sliding windows with k-Sentence Intersection (SI-k) and k-Sentence Stride (SS-k). In either case, the first sequence starts with the document's first sentence and includes more sentences up to the maximum sequence length T. The starting sentence of the next sequences (if any) is determined by the choice of specific partitioning strategy, as described below.

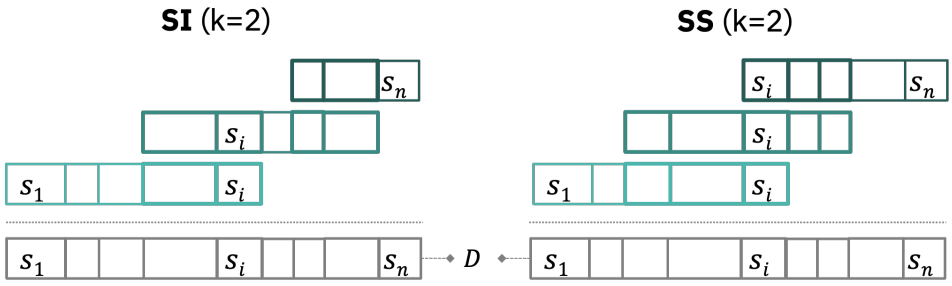

Figure 4.3: Sample multi-prediction document partitioning methods: SI-2 and SS-2. All sentences actively take part in the prediction task, resulting in multiple predictions for some or many sentences depending on the degree of overlap.

Sentence Stride (SS-k). As introduced earlier, each sequence, denoted as $Seq_{i_s:i_e} = \{s_i\}_{i \in i_s:i_e}$, spans from sentence index i_s to i_e of document D. Using a sentence stride of $k > 0$, the next sequence $Seq_{i'_s:i'_e}$ will begin from the $(k+1)$-th sentence of the previous sequence $Seq_{i_s:i_e}$ if applicable:

$$i'_s = min(i_s + k, i_e)$$

In case k is large enough to exhaust the region of previous sequence, minimum overlap is used for the next sequence by starting from the last sentence of the previous sequence (i.e., $i_e + 1$). Overall, the lower the stride (k) the higher the overlap between consecutive sequences. Particularly, SS-1 is the most extensively-overlapped and the least efficient partitioning method among multi-prediction settings. Sequence generation with SS-2 document partitioning is

illustrated in Figure 4.3.

Sentence Intersection (SI-k). With a sentence-level intersection of $k > 0$, the next generated sequence $Seq_{i'_s:i'_e}$ will initiate from the k-th last sentence of the previous sequence $Seq_{i_s:i_e}$ if applicable:

$$i'_s = \max(i_e + 1 - k, i_s + 1)$$

Again, if k is large enough to exhaust the region of the previous sequence, maximum overlap is used for the next sequence by starting from the second sentence of the previous sequence (i.e., $i_s + 1$). This time, the overlap between consecutive sequences is increased at higher ks. Particularly, SI-1 is the least overlapped and lightest partitioning method among multi-prediction settings.

Note that due to variable sentence-length of generated sequences, SS-k and SI-k partitioning methods, although inherently similar, cannot be generally reduced to (i.e., obtained from) one another using any two respective ks. However, in extreme cases, SS-∞ is equivalent to SI-1 and SI-∞ is equivalent to SS-1 according to their definition. Sequence generation with SI-2 document partitioning is illustrated in Figure 4.3.

In multi-prediction document partitioning methods, as name suggests, sentences receive multiple predicted labels from their participation in multiple sequences. Therefore it is necessary to reduce them into single predictions for individual sentences. The subsequent section will outline several weighting functions designed to effectively consolidate those multiple sentence predictions. In the experiments section, we will assess the impact of the proposed document partitioning strategies on model accuracy and runtime efficiency.

4.2.3 Weighted Aggregation of Multiple Sentence Predictions

By introducing overlap between consecutive sequences as in SS-k and SI-k partitioning methods, multiple predictions arise for sentences as they participate in different sequence spans. For example, as illusterated in Figure 4.4, s_7 appears in sequence 2, 3 and 4 and thus receives three respective predictions, $p_7^{(2)}$, $p_7^{(3)}$, $p_7^{(4)}$. To ensemble such multiple predictions, we aggregate output probabilities of the same sentence across different sequence occurrences. Ensemble prediction on overlapped sequences is naturally expected to result in more reliable predictions due to voting mechanism across multiple sequence observations.

The process involves combining the initial output probabilities associated with multiple occurrences of individual sentences into single decisions. More specifically, sequence sentence probabilities (i.e., softmax-normalized logits) are compiled and averaged out to compute a reduced sentence probability as shown in Equation 4.5. For each sentence, the reduced probability directly informs the final binary decision based on a validation-tuned decision threshold.

$$p_i = \sum_{j:\, s_i \in Seq_j} w_i^{(j)} p_i^{(j)} \tag{4.5}$$

Given that sentences near the boundaries of a sequence typically have limited contextual information, their predictions may not be as robust as those located centrally. Therefore, we propose several position-aware weighting mechanisms to aggregate estimated probabilities. These weights adjust the influence of each sentence's prediction on the final aggregated result according to its position within the sequence: sentences closer to sequence boundaries receive lower weights in the aggregation process. With that said, weight of sentence i in sequence j, denoted as $w_i^{(j)}$ in Equation 4.5, can be derived from a position-aware weighting function

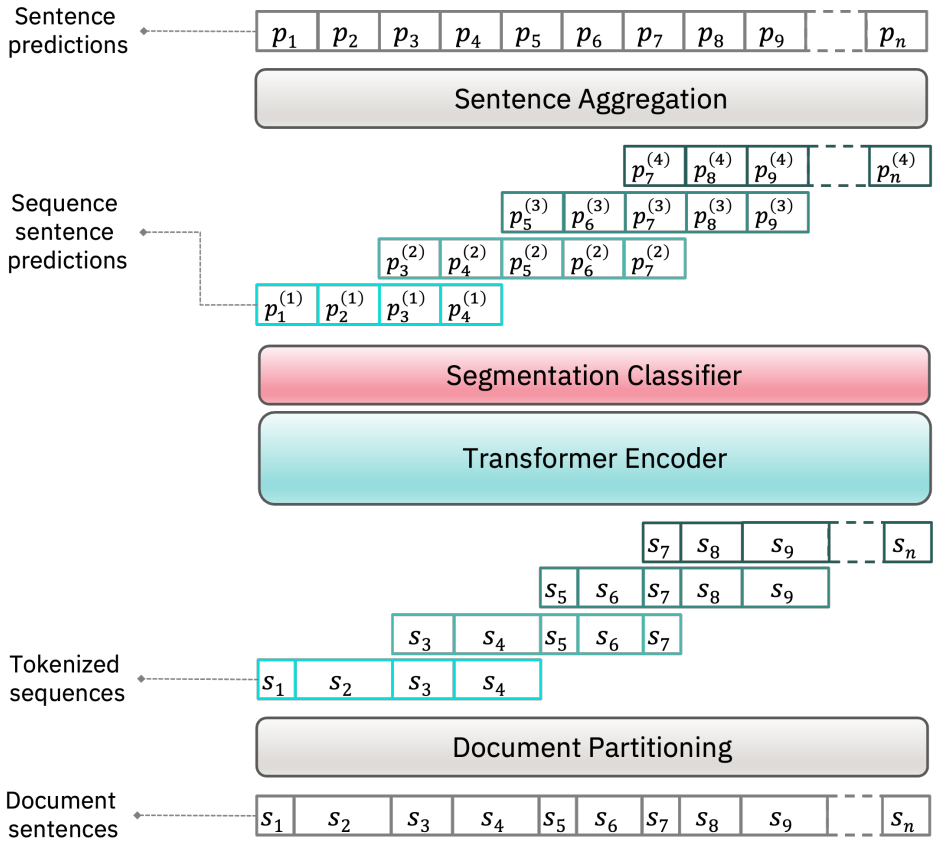

Figure 4.4: Inference pipeline of our WeSWin model. Input sequences are formed using SS-2 as a sample document partitioning method here. Tokenized sequences are fed to the transformer encoder followed by a segmentation classifier to derive sequence sentence predictions. For multi-prediction partitioning methods, sentence aggregation is applied on top of multiple sentence predictions ($p_i^{(j)}$) to derive final sentence predictions (p_i) for individual sentences (s_i) in the document.

of form $wfn(i_j, m_j)$, where i_j is the local index or position of sentence i within sequence j and m_j is the total number of sentences in sequence j. To concretely define $wfn(.)$, we propose and examine three alternative weighting functions, including *linear*, *polynomial*, and *loss-based* designed to modulate the contribution of individual predictions to the aggregated decision as follows. [1]

[1]The subscript j is dropped in the following definitions for simplicity.

Linear Positional Weights. Linear weight assignment function is defined as,

$$\text{lin}(i, m \mid k, \varepsilon) = \varepsilon + (1 - \varepsilon) \cdot \left(\frac{\min(d_{i,m}, k)}{k} \right) \tag{4.6}$$

where $d_{i,m} = \min(i-1, m-i)$ is the (symmetric) positional distance of sentence i to its closer boundary of the sequence with m sentences. The parameter ε is the minimum weight assigned to the boundary sentences ($0 < \varepsilon < 1$). The parameter k is a positive integer that controls the rate of increase in weight as we move from a boundary position to the center. In other words, with linear weighting, as we move from side positions to the center of a sequence of m sentences, position weights are linearly increased from ε to an upper bound (which is not more than 1) in k steps. (See the blue curves in Figure 4.5)

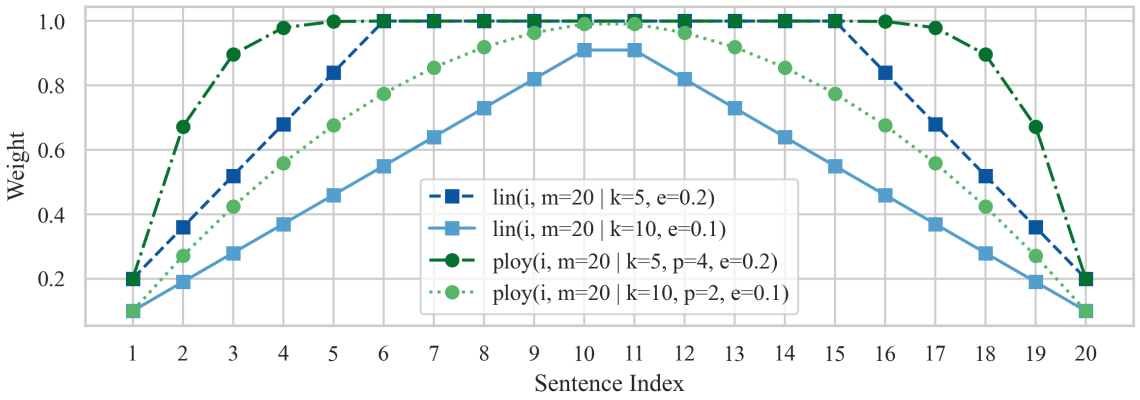

Figure 4.5: Sample linear and polynomial weight assignment functions, mapping sequence-wide sentence positions into their respective weights. The assigned weight to each sentence observation will determine its contribution in the final sentence probability.

Polynomial Positional Weights. Polynomial weight assignment function is defined as,

$$\text{poly}(i, m \mid k, p, \varepsilon) = \varepsilon + (1 - \varepsilon) \cdot \left(1 - \left(1 - \frac{\min(d_{i,m}, k)}{k} \right)^{p} \right) \tag{4.7}$$

where i, m, $d_{i,m}$ and parameter ε and k are defined similarly as in the linear function. Here both k and p control how fast the weight changes when moving from a boundary position to the center. Set to a non-zero even integer, parameter p is the degree of the polynomial function. In this case, as we move from a boundary position to the center, the weight increases from ε to the upper bound (which is not more than 1) in k steps following a polynomial function of degree p. (See the green curves in Figure 4.5)

Loss-based Positional Weights. In this setting, we propose to use a weight function based on the model performance on individual sentence positions of the validation set. More specifically, we take the average loss for individual sentence positions identified by their relative proximity to the closer boundary of sequence. Position-weight mass is derived by complementing the position-loss distribution, followed by non-decreasing smoothing when moving from the boundary positions to the center.

As the positional loss distribution, $l(.)$ maps sequence sentence positions to their respective average loss values (See the upper part of Figure 4.6). Given that $c(.)$ is the normalized complement of the positional loss distribution $l(.)$, the loss-based weight for the ith sentence in a sequence with m sentences is defined as:

$$\text{lob}(i,m \mid k,\varepsilon) = \varepsilon + (1-\varepsilon) \cdot \begin{cases} \max\limits_{1 \le i' \le i} c(i',m) & \text{if } i \le k, \\ \max\limits_{i \le i' \le m} c(i',m) & \text{if } i > m-k, \\ \max\limits_{k < i' \le m-k} c(i',m) & \text{otherwise.} \end{cases} \tag{4.8}$$

where $0 < \varepsilon < 1$ sets the minimum weight for boundary positions, and the positive integer k (which is maxed at $\lfloor \frac{m+1}{2} \rfloor$) controls the (asymmetric) growth rate from each boundary posi-

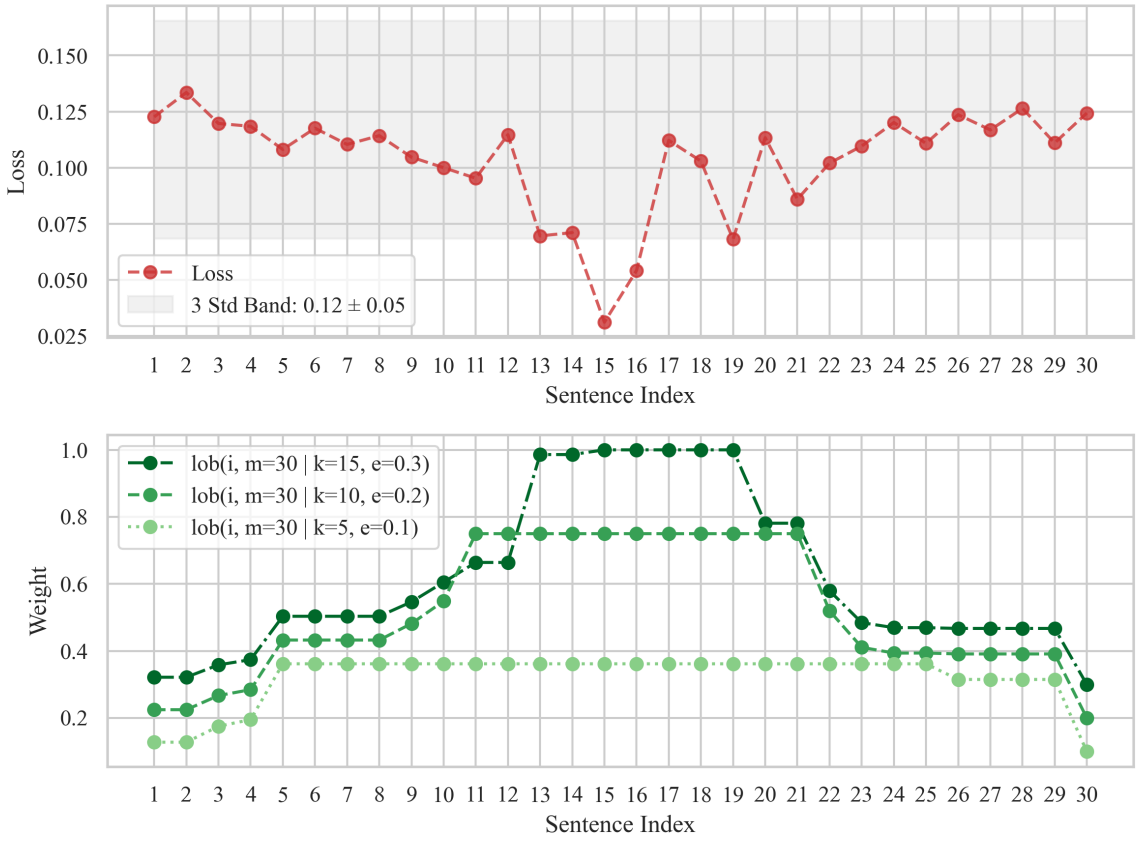

Figure 4.6: a) Positional loss distribution $l(.)$ over 1k random documents from the validation set of Wiki-727K dataset. Sequences are derived using SS-4 document partitioning method. b) Sample loss-based weight functions mapping sequence-wide sentence positions into their respective weights. The assigned weight to each sentence observation will determine its contribution in the final sentence probability.

tion to the center of a sequence of length m. The *max* function smooths out the fluctuations in c(.) to ensure the weights are changing in non-decreasing order from sides to center. In $lob(.)$ weight distribution, the weight starts from the ε value in boundary positions and increases in proportion to the normalized complement of positional loss function $l(.)$, denoted as $c(.)$, up to a saturation point (which is no more than 1) at the center, the growth rate of which is controlled by k. A larger k value leads to a higher maximum weight in a narrower middle range of the sequence. Figure 4.6 illustrates the positional loss mass at the top and a few smoothed

loss-based weight functions at the bottom.

In the experiments section, we explore several configurations of individual weight functions to adjust the contribution of multiple sentence observations, and the best one is picked for inference based on validation results.

4.3 Multi-task Training of Segmentation and Retrieval

We aim to enhance segmentation performance through multi-task training with sentence retrieval as an auxiliary task. Therefore, we propose to jointly fine-tune the two segmentation and retrieval tasks using a shared transformer with two classification heads. The goal is to learn more robust sentence representations to improve the model's ultimate generalization mainly for the task of document segmentation. Through simultaneous training over segmentation and retrieval data, the model can pick up query-awareness for document sentence representations, implicitly guiding the segmentation to identify cohesive regions more effectively. In the following, we introduce sentence retrieval task first, and then dive into the specifics of our multi-task model, referred to as WeSWin-Ret.

4.3.1 Sentence Retrieval

Sentence retrieval aims to identify relevant sentences from a document to an input query. In this work, sentence retrieval is mostly used as an auxiliary task to improve document segmentation by introducing query awareness to sentence representations in a shared embedding space. Given a document $D = \{s_1, s_2, ..., s_n\}$ made of n sentences and a query q, sentence re-

trieval aims to find the relevant sentences from the document D that can collectively answer the query q. Same as for segmentation, sentence retrieval is defined as a sentence-level sequence labeling task responsible for predicting the relevance probability of context sentences, denoted as $p'_k \in \{p'_1, ..., p'_n\}$, with respect to an input query.

Training Sequence Formation. For sentence retrieval, we use a similar sequence tokenization as introduced in Section 4.1, with the difference that sequences of sentences from the input document D (i.e., $Seq_{i_s:i_e} = \{s_i\}_{i \in i_s:i_e}$) are augmented by the input query q:

$$QSeq^{(T)}_{i_s:i_e} = \langle [CLS], s_{i_s}, [SNT], s_{i_{s+1}}, [SNT], ..., s_{i_e}, [SNT], [SEP], q, [EOS] \rangle \tag{4.9}$$

where each sentence s_i in the sequence is represented (and followed) by the special $[SNT]$ token. The query tokens are separated from the sequence's sentence tokens using the standard $[SEP]$ token. After making room for the query tokens at the end of the sequence, we fill up the available context window by squeezing as many sentences as possible in the sequence. In cases where the document exceeds the transformer's input size, CR-1 document partitioning is utilized to generate sequences for the training batch, same as in the segmentation task (Section 4.1).

For each sequence in the training batch, we use the updated $[SNT]$ token representations as input to a retrieval classification head to label each sentence as *relevant* or *irrelevant* categories. Finally, the binary cross-entropy loss is used to train the model end-to-end.

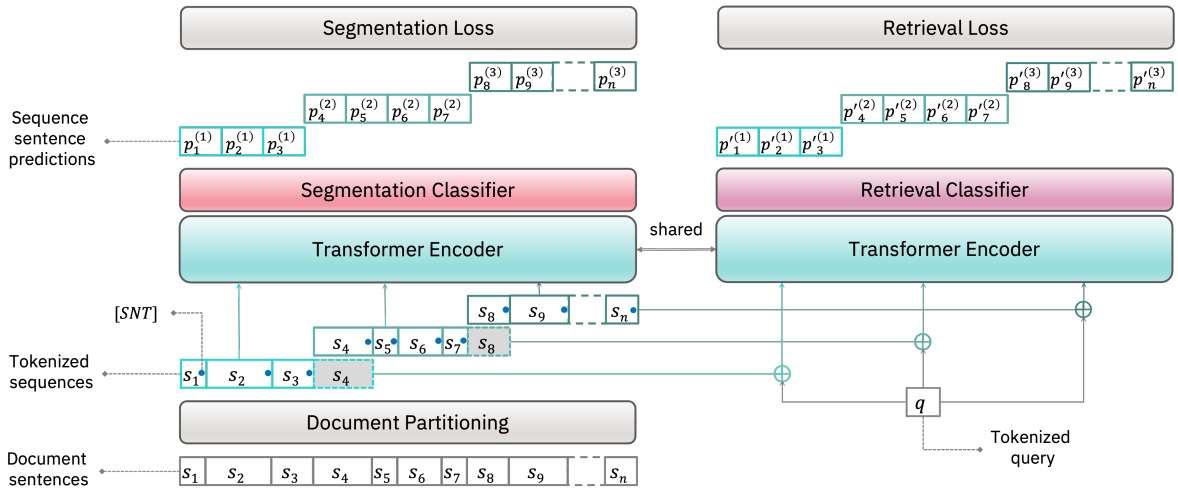

Figure 4.7: Training pipeline of our WeSWin-Ret model. Input sequences from either task are formed using CR-1 document partitioning method. While equipped with two task-specific classifiers, the transformer itself is shared between segmentation and retrieval tasks. Alternating between the two tasks, BCE loss function is used to update the model parameters end-to-end.

4.3.2 Multi-Task Training.

In WeSWin-Ret, we propose to formulate both document segmentation and sentence retrieval as sentence-level sequence labeling tasks within a shared transformer encoder equipped with two task-specific classification heads. The shared transformer encoder is responsible for learning multi-task sentence representations by taking task-specific input sequences coming from a potentially partitioned input document (and a corresponding query, in case of retrieval).

Figure 4.7 illustrates an overview of our multi-task training framework consisting of a shared transformer encoder with two task-specific classification heads. First, an input document is partitioned into properly-sized sequences that are respectively tokenized according to Equation 4.1 for segmentation and Equation 4.9 for retrieval. The resulting samples are combined to form task-alternating training batches, serving as input to the transformer encoder.

Through multi-task training, contextualized sentence representations within batch sequences are derived and passed to their respective task-specific classification head to output topic-shift or query-relevancy labels. A standard binary cross-entropy loss is used to train the model.

Chapter 5

Experimental Results

5.1 Datasets

We use three benchmark datasets, namely Wiki-727k (Koshorek et al., 2018) and en_city and en_disease subsets of WikiSection (Arnold et al., 2019), to evaluate document segmentation. We utilize NLQuAD dataset (Soleimani et al., 2021) formulated as a sentence retrieval task for multi-task training with segmentation.

5.1.1 Wiki-727k

The Wiki-727k dataset, as introduced by Koshorek et al. (2018), is a comprehensive collection of 727,746 Wikipedia documents segmented according to their table of contents. As one of the crucial datasets in the field of text segmentation, Wiki-727k provides a vast and diverse

corpus for training and evaluating text segmentation models. It is designed to overcome the limitations of previous datasets, which were either too small or not representative of natural text distributions, by offering a large, natural, and open-domain collection of documents with well-defined segmentation. Each document within this dataset has been pre-processed to ensure the removal of non-text elements such as images and tables, while also ensuring that each segment is properly divided into sentences using the Punkt tokenizer (Bird et al., 2009).

Dataset	#Train	#Validation	#Test	#Segment/Doc	#Sentence/Doc	#Word/Doc
Wiki-727k	582,160	72,354	73,232	6.18	52.65	1,117.32
en_city	13,679	1,953	3,907	6.84	53.24	1058.3
en_disease	2,513	359	718	7.75	54.57	1122.76

Table 5.1: Cardinality and length statistics of text segmentation datasets: Wiki-727k and WikiSection's en_city and en_disease subsets.

5.1.2 WikiSection

The WikiSection dataset (Arnold et al., 2019) consist of segmented Wikipedia articles in domain settings. Two notable subsets within this dataset are "en_city" and "en_disease". The en_city dataset contains 19.5k articles about diverse city-related topics. The en_disease dataset consists of 3.6k medical and health-related documents with scientific details from Wikipedia. We use en_city and en_disease datasets to further evaluate our baselines in domain-specific settings.

5.1.3 NLQuAD

The NLQuAD dataset, introduced by Soleimani et al. (2021), is designed for the task of non-factoid long question answering, which necessitates document-level language understand-

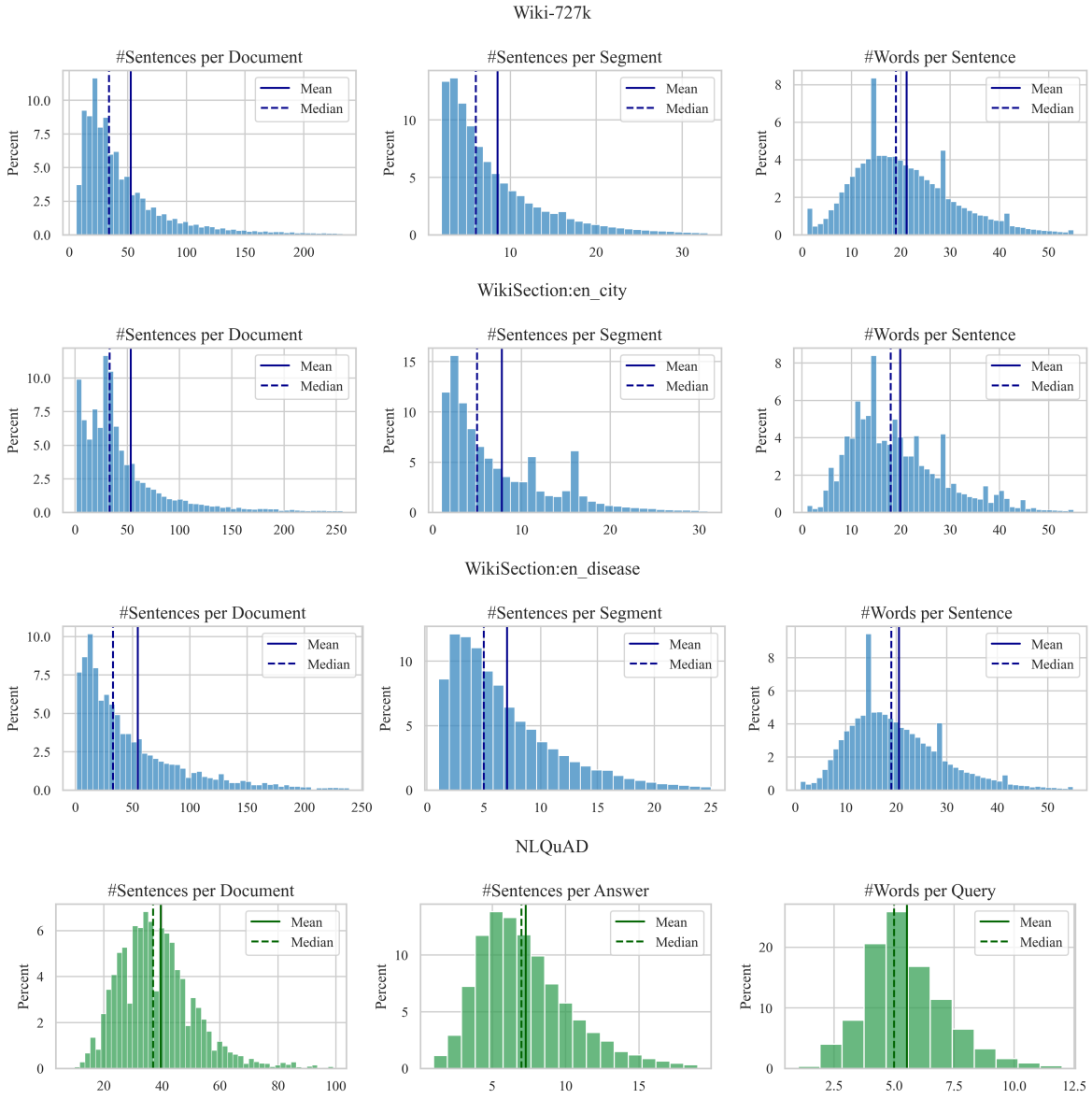

Figure 5.1: Length distribution of a) Wiki-727k, b) WikiSection:en_city, c) WikiSection:en_disease, and d) NLQuAD datasets. Samples with extreme values are excluded past three standard deviation from the average.

ing. Unlike conventional QA datasets focused on relatively short span detection, NLQuAD comprises non-factoid questions that demand descriptive, multi-sentence answers or opinions, thereby challenging models with its requirement for extensive text comprehension. The

dataset consists of 31k non-factoid questions derived from 13k BBC news articles. The questions and answers are extracted from article sub-headings and the subsequent paragraphs, respectively. Since the answer portion spans multiple sentences, we can use this dataset for a sentence-level retrieval task to further test our model's ability to learn document-aware sentence representation required for identifying relevant sentences to a given query.

Dataset	**#Train**	**#Dev**	**#Test**	**#Sentence/Doc**	**#Word/Doc**	**#Word/Answer**	**#Word/Query**
NLQuAD	24,541	3,024	3,017	39.65	824.95	152.74	5.53

Table 5.2: Cardinality and length statistics of NLQuAD as a sentence retrieval dataset

According to Table 5.1 and Table 5.2, the average document length in the presented datasets significantly exceeds the typical 512-token input limit of transformer-based models, highlighting the need for proper document partitioning techniques ensuring the optimal use of context. Detailed length distribution plots are shown in Figure 5.1 for individual datasets.

5.2 Evaluation Metrics

In this section, we mainly use F1 score, alongside with Precision and Recall, to evaluate segmentation predictions in sentence-level binary classification.

5.2.1 F1 Score

F1 score is widely used in text segmentation literature to reflect how well the segments identified by the algorithm match the true segments in the text in sentence-level granularity. F1 score is derived from *Precision* and *Recall* measures. Precision is the number of correct positive predictions over the total number of positive predictions, denoted as $\frac{TP}{TP+FP}$, while Recall is the number of correct positive predictions over the total number of positive labels denoted as $\frac{TP}{TP+FN}$, where True Positive (TP) reflects correctly identified segment boundaries, False Positive (FP) reflects incorrectly identified boundaries, and False Negative (FN) reflects missed boundaries. The F1 score is defined as the harmonic mean of Precision and Recall, providing a balance between them:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

While there are other evaluation metrics, such as P_k (Beeferman et al., 1997) and WD (Pevzner and Hearst, 2002), used in previous works to evaluate text segmentation models, we only report the superior F1 score as it is more strict in pinpointing misalignments between predicted and reference segment boundaries.

5.3 Experimental Results

5.3.1 Comparison with SOTA Methods

We compare our proposed methods with several baseline methods on three benchmark datasets. We trained both our single-task (WeSWin) and multi-task (WeSWin-Ret) frameworks based on three different transformer models, namely BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020).

Base Model	**Model**	**Wiki-727k**		
		F1	**Prec**	**Rec**
RNN	Bi-LSTM (Koshorek et al., 2018)	57.7	69.3	49.5
BERT	CSBERT : CE weights (Ghosh et al., 2023)	60.9	-	-
	CSBERT : Focal Loss (Ghosh et al., 2023)	61.4	-	-
	Cross-segment BERT (Lukasik et al., 2020)	66.0	69.1	63.2
	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
	WeSWin-Ret:BERT (**ours**)	75.11	75.61	74.62
RoBERTa	SeqModel:RoBERTa-Base (Zhang et al., 2021)	70.2	66.2	74.7
	WeSWin:RoBERTa (**ours**)	77.23	77.37	77.08
	WeSWin-Ret:RoBERTa (**ours**)	**77.79**	80.22	75.51
Longformer	Longformer-Base (2048) (Yu et al., 2023)	76.27	-	-
	Longformer-Base+TSSP+CSSL (2048) (Yu et al., 2023)	77.16	-	-
	WeSWin:Longformer-1024 (**ours**)	77.38	77.36	77.39
	WeSWin-Ret:Longformer-1024 (**ours**)	**77.4**	76.28	78.57
	WeSWin:Longformer-2048 (**ours**)	77.55	78.11	77.01
	WeSWin-Ret:Longformer-2048 (**ours**)	**77.92**	81.34	74.77

Table 5.3: Comparison of our WeSWin and WeSWin-Ret models with SoTA segmentation methods on Wiki-727k dataset

Table 5.3 shows the comparison results on the Wiki-727k dataset, where the models are grouped by the base transformer model used. Both BERT and RoBERTa have an input size of 512 tokens while our Longformer baseline is employed with 1024 and 2048 tokens. With BERT as the base model, both of our WeSWin and WeSWin-Ret models significantly outperform the baselines. The same happens with the RoBERTa baselines. With Longformer,

the SOTA baselines use an input size of 2048 tokens. We trained our models with both 1024 and 2048 input size, respectively denoted as Longformer-1024 and Longformer-2048. We observe that both of our Longformer-1024 and 2048 baselines outperform the SOTA method with the 2048-token input size. [1]

Between our two settings, as shown in Table 5.3, WeSWin-Ret considerably outperforms WeSWin across RoBERTa and Longformer-2048 while remaining competitive for the other two transformers, BERT and Longformer-1024. More detailed comparisons between single and multi-task settings are given in Subsection 5.3.5.

Model	**en_city**			**en_disease**		
	F1	**Prec**	**Rec**	**F1**	**Prec**	**Rec**
SEC 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 (Yu et al., 2023)	85.14	-	-	<u>77.33</u>	-	-
WeSWin:RoBERTa (**ours**)	**86.85**	89.13	84.68	**77.39**	76.76	78.03
WeSWin:Longformer-2048 (**ours**)	<u>86.77</u>	89.64	84.08	76.68	75.65	77.73

Table 5.4: Comparison of our proposed WeSWin model with SoTA segmentation results on en_city and en_disease

In domain settings, Table 5.4 compares our WeSWin:RoBERTa and WeSWin:Longformer-2048 models with the Longformer-based SOTA baseline on the en_city and en_disease datasets. All these models are pre-trained on Wiki-727k and fine-tuned on the domain-specific training data sets. We observe that our RoBERTa-based model (with 512 input token limit) outperform the Longformer-based SOTA method with 2048-token limit on both en_city and en_disease. This again illustrates the high effectiveness of our proposed methods.

In the following sections, we first compare the proposed partitioning methods (Subsection 5.3.2) and sentence aggregation functions (Subsection 5.3.3). Next, we take a closer look on the performance of three explored transformer models, trained and tested on Wiki-727k

[1] Results from baselines are taken from respective papers.

dataset (Subsection 5.3.4). We further evaluate our model's robustness on domain-specific en_city and en_disease benchmarks (Subsection 5.3.6) as well as an industrial automotive dataset (Chapter 6). We also evaluate our method in a downstream RAG task in Subsection 5.3.7.

5.3.2 Comparison of Document Partitioning Methods

Introducing overlap in sequences generated from input documents, either as context-only support (Subsection 4.2.1) or as multi-prediction active overlap (Subsection 4.2.2), has significantly contributed to the model performance. As previously illustrated in Figure 4.6, the transformer model has proven to be less reliable in predicting boundary sentences within a document sub-sequence simply because it cannot see sufficient right or left context from the rest of the document. This motivates us to either designate the boundary regions for context only or to aggregate multiple overlapped sentence predictions, ensuring each sentence is reasoned over in a non-boundary position at least once or most of the time.

By introducing some repetition in document partitioning, we may significantly improve the model's performance at the expense of relatively higher inference runtime. More specifically, in context-only partitioning, the model has the opportunity to rely on additional support sentences in the left and/or right boundaries of a sequence when predicting the labels of active sentences. In multi-prediction partitioning on the other hand, all sentence observations receive predictions. Therefore, multiple sentence probabilities, coming from multiple sequences, are further aggregated into final sentence decisions.

In this experiment, we use our WeSWin:RoBERTa baseline to evaluate each partitioning

method category. The model is fine-tuned on Wiki-727k and tested on a random split of 1k documents sampled from the test set with a fixed seed for all experiments. Following the literature, prediction for the last sentence of a document is always excluded from the evaluation results throughout this work.

5.3.2.1 Context-only Document Partitioning

In this section, we compare several context-only document partitioning strategies, using our WeSWin:RoBERTa baseline, trained and tested on the Wiki-727k dataset. Figure 5.2 illustrates the performance of several CR-k and CLR-k partitioning methods, highlighting the trend of F1 score against inference speed for multiple k values. Inference speed is measured in terms of number of documents processed per minute (Doc/Min).

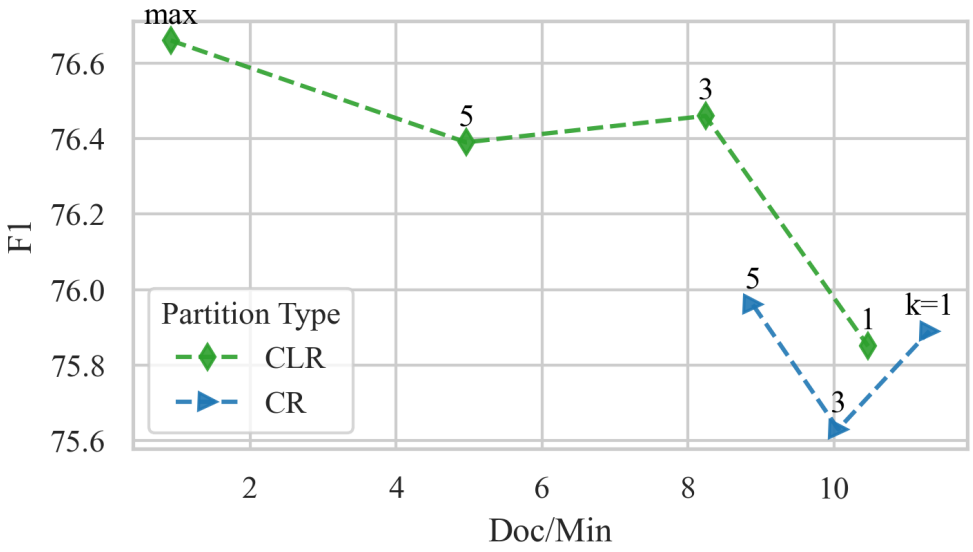

Figure 5.2: Comparison of several context-only document partitioning methods in terms of accuracy (F1) and runtime speed (Doc/Min) using WeSWin:RoBERTa model, trained and tested on Wiki-727k. The annotation on top of each CR-k or CLR-k partitioning baseline shows the value of its k parameter.

Our lightest partitioning strategy, denoted as CR-1, achieves an F1 score of 75.89 with the runtime speed of 11.31 Doc/Min. As our lightest partitioning baseline, CR-1 serves as a reference point for evaluating other context-only partitioning strategies. [2] Increasing overlap up to 5 sentences, denoted as CR-5, only results in a slightly improved F1 of of 75.96 with a trade-off in processing speed, sitting at 8.89 Doc/Min. Therefore, more than one-sentence context-only overlap in the right-most positions of sequences may not be justified. However, higher overlap from both left and right-most positions may be beneficial as seen with the CLR-k method.

We may achieve higher accuracy when designating context-only sentences on both sides of the sequences as in CLR-k partitioning method. In this regard, CLR-1 stands on par with CR-1 method in terms of accuracy with slightly higher runtime. Increasing left and right overlap of consecutive sequences to 3 sentences, CLR-3 achieves a F1 score of 76.46, showing a significant improvement over the CR-1 and CLR-1 baselines. However, this comes at the cost of slightly reduced performance at 8.24 Doc/Min. With the maximum possible overlap, CLR-max only considers the central sentence as active within a sequence while designating all other sentences as context-only on each side. Owing to access to a large context, CLR-max achieves the highest accuracy with an F1 score of 76.66, but with significantly reduced efficiency at only 0.91 Doc/Min.

In conclusion, the analysis reveals a clear trade-off between accuracy and efficiency among alternative context-only document partitioning methods. CR-1 and CLR-3 methods achieve competitive accuracy while maintaining reasonable efficiency.

[2] Being the lightest, CR-1 is the partitioning method used in the training of our models.

5.3.2.2 Multi-prediction Document Partitioning

In this section, we provide a comprehensive evaluation of multi-prediction document partitioning methods. More specifically, we compare SI-k and SS-k partitioning methods, evaluated on the Wiki-727k dataset using our WeSWin:RoBERTa baseline. In this experiment, the F1 score is studied as the primary evaluation metric in relation to the number of documents processed per minute (Doc/min) as the efficiency metric. Figure 5.3 illustrates the trade-off between accuracy and performance when increasing the partitioning overlap for each of SI-k and SS-k methods though parameter k. For reduction of multiple sentence predictions, uniform (as opposed to weighted) mean aggregation is used in this experiment.

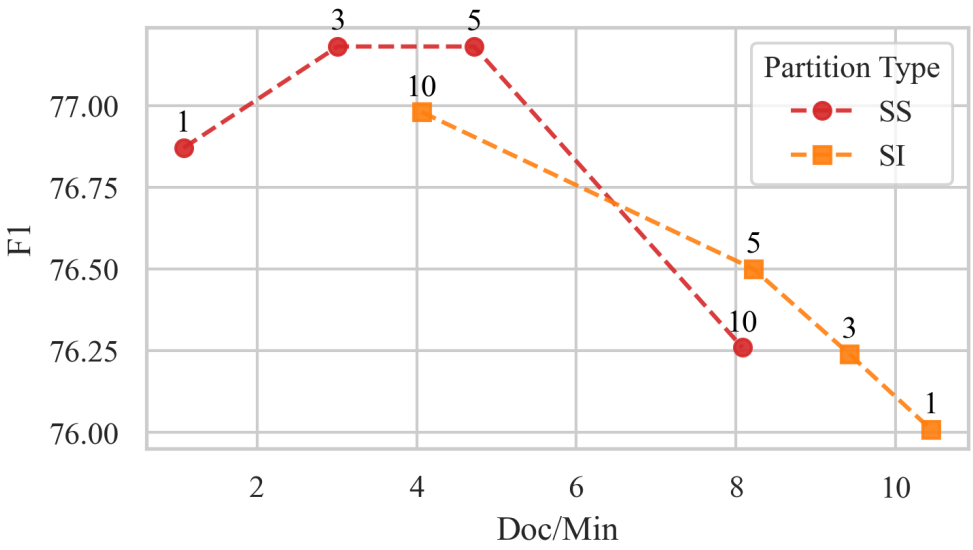

Figure 5.3: Comparison of several multi-prediction document partitioning methods in terms of accuracy (F1) and runtime speed (Doc/Min) using WeSWin:RoBERTa model, trained and tested on Wiki-727k. The annotation on top of each CR-k or CLR-k partitioning baseline shows the value of its k parameter.

In the SI-k partitioning baselines, sequences are generated with an intersection of k sentences. As the intersection overlap increases from 1 sentence in SI-1 to 10 sentences in SI-10,

the F1 score consistently improves. Starting with least overlap, SI-1 achieves an F1 score of 76.01 while processing 10.44 Doc/Min, showing a slight improvement over the single-decision CR-1 baseline. Increasing the intersection to 3, 5, and 10 sentences consistently improves the accuracy at higher inference costs. Particularly, SI-5 and SI-10 respectively achieve 76.5 and 76.98 F1 scores with reduced processing speed of 8.22 and 4.06 Doc/Min. Model accuracy can be further improved by introducing even more overlap, which is obtained through the alternative SS-k partitioning method.

When using a sentence stride of k between consecutive sequences, denoted by SS-k, increasing overlap by using smaller ks will generally improve the F1 score. Starting with SS-10 partitioning, the F1 scores stands at 76.26 with inference speed of 8.08 Doc/Min. Increasing overlap to SS-5 results in a significant boost in F1, reaching 77.18, justifying the increased runtime of 4.72 Doc/Min. Beyond a point, more overlap does not necessarily improve the F1 score. For instance, SS-3 and SS-1 (with 78.18 and 76.87 F1 socres), hardly match the performance of SS-5 while being more extensively overlapped and thus more costly (sitting at respective runtime speeds of 3.01 and 1.08 Doc/Min). It is worth mentioning that SS-5 outperforms the SI-10 method with a better runtime trade-off, suggesting overall superiority of stride (SS-k) over intersection (SI-k) in well-overlapped settings.

The comparison highlights a clear trade-off between the F1 score and runtime efficiency, measured in the number of documents processed per minute, across different document partitioning methods. Methods with higher overlap, such as SI-10 and SS-5, stand out in terms of superior F1 scores at higher but feasible runtime trade-offs. The best partitioning method, along with the choice of k, can be determined using hyper-parameter tuning over the validation set. The degree of overlap in the partitioning method can also be adjusted based on the time-sensitivity of the chunking application. In any case, the document segmentation task

can often tolerate higher inference runtime for the sake of improved accuracy, as it is often an offline prerequisite for other downstream applications such as information retrieval and question answering.

5.3.3 Exploring Weight Assignment Functions

As introduced in Subsection 4.2.3, multi-prediction document partitioning method, i.e., SI-k or SS-k, result in multiple predictions per individual sentences that need to be further aggregated. In this regard, we proposed several weighting functions to aggregate multiple sentence predictions coming from overlapped sequences.

Particularly, we explore three weighting mechanisms—*linear*, *polynomial*, and *loss-based*—and compare them with the *uniform* mean aggregation in terms of the ultimate F1 score. In each method, multiple sentence predictions, with different positions within multiple sequences, are weighted proportionally to their position index in the participating sequences. Sentence positions closer to the sequence boundaries are assigned lower weights to prioritize central positions that have access to a larger bi-directional context, leading to more reliable predictions.

Figure 5.4 compares F1 scores from different weighting functions (including uniform) on the Wiki-727k dataset using WeSWin:BERT and WeSWin:Longformer-1024 models. SS-k document partitioning method is used in this experiment with five stride values. Sequence overlap increases at lower stride values.

For the BERT model, the loss-based weighting performs better than linear or polynomial methods. Particularly, in SS-5 partitioning, loss-based weighting improves upon the uniform

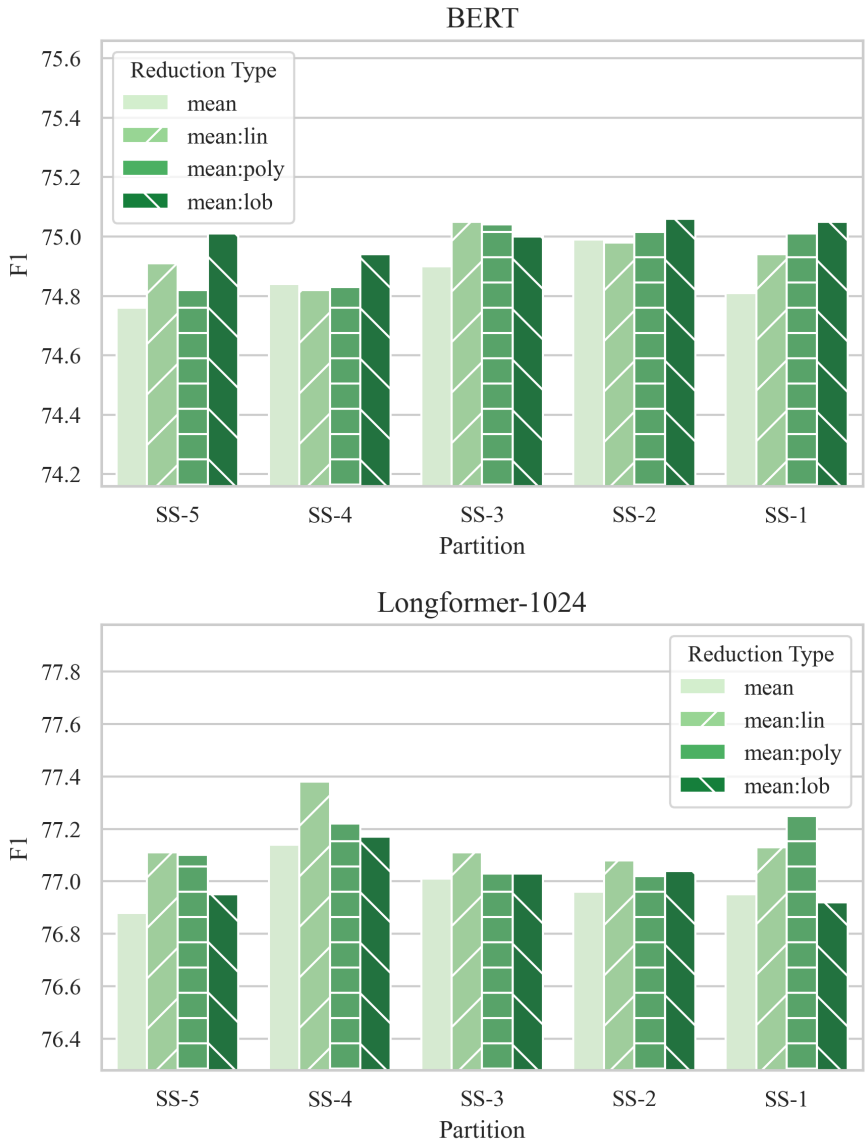

Figure 5.4: Effect of different weighting functions in sentence aggregation of overlapping sequences using a) WeSWin:BERT and b) WeSWin:Longformer-1024 model, trained and tested on Wiki-727k.

weighting by increasing the F1 score from 74.76 to 75.01. However, for the Longformer-1024 model, the linear and polynomial weighting functions perform superior to the loss-based method in average. In SS-5 partitioning, for instance, linear and ploynomial methods achieve F1 scores of 77.11 and 77.10, improving both the uniform and loss-based methods

with respective F1 scores of 76.88 and 76.95.

Results indicate that all proposed weighting functions improve the model's accuracy compared to the uniform mean aggregation. However, since no single function stands out as the best, we consider all of them as potential methods and select a function based on the corresponding validation performance for the specific transformer model and partitioning method used. Once we identify the best weighting function through hyperparameter tuning, we utilize it in evaluating the test set.

5.3.4 Comparison of Transformer Encoders and Context Size

In this section, we compare the performance of commonly-used transformer encoders, including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020). Starting with reviewing the specifics of each transformer encoder, we proceed to fine-tune and evaluate our WeSWin model once per choice of transformer to compare their performance on Wiki-727k as the main segmentation benchmark.

BERT model (Devlin et al., 2019) revolutionized pre-training deep language representations by utilizing a bidirectional training methodology on the Transformer architecture. BERT processes text by considering both the left and right contexts simultaneously across all layers, using two primary tasks during pre-training: masked language modeling (MLM) and next sentence prediction (NSP). This enables the model to develop a richer understanding of language context, facilitating enhanced performance across a variety of NLP tasks such as question answering and language inference by fine-tuning with just one additional output layer.

Building on BERT's foundations, RoBERTa (Liu et al., 2019) modifies several of BERT's original training parameters and tasks to improve performance. RoBERTa notably discards the next sentence prediction task, focusing solely on masked language modeling, and optimizes training by adjusting factors like batch size, learning rate, and the amount of training data. These modifications help RoBERTa not only outperform BERT but also suggest that the extensive training with more data and longer sequences can further enhance model capabilities, particularly in understanding complex language patterns and improving on benchmarks like GLUE and SQuAD.

Longformer (Beltagy et al., 2020), on the other hand, focuses on efficiently managing longer documents. Unlike BERT and RoBERTa, which are optimized for shorter texts, Longformer introduces a novel attention mechanism combining a sliding window local attention with selective global attention to handle longer sequences effectively. Global attention particularly enables the model to focus on key tokens such as the beginning of a document or special tokens. This design specifically supports extended tasks such as document classification and long-form question answering. The model's attention mechanism is tailored to reduce the quadratic dependency on sequence length in computation, making it particularly suitable for applications involving large-scale documents.

In this section, we designed an experiment to evaluate the performance of aforementioned transformer models with varying context lengths for the task of document segmentation. This comparison aims to identify the appropriate context length while illustrating the trade-offs between the choice of transformer and computational costs categorized by several partitioning strategies.

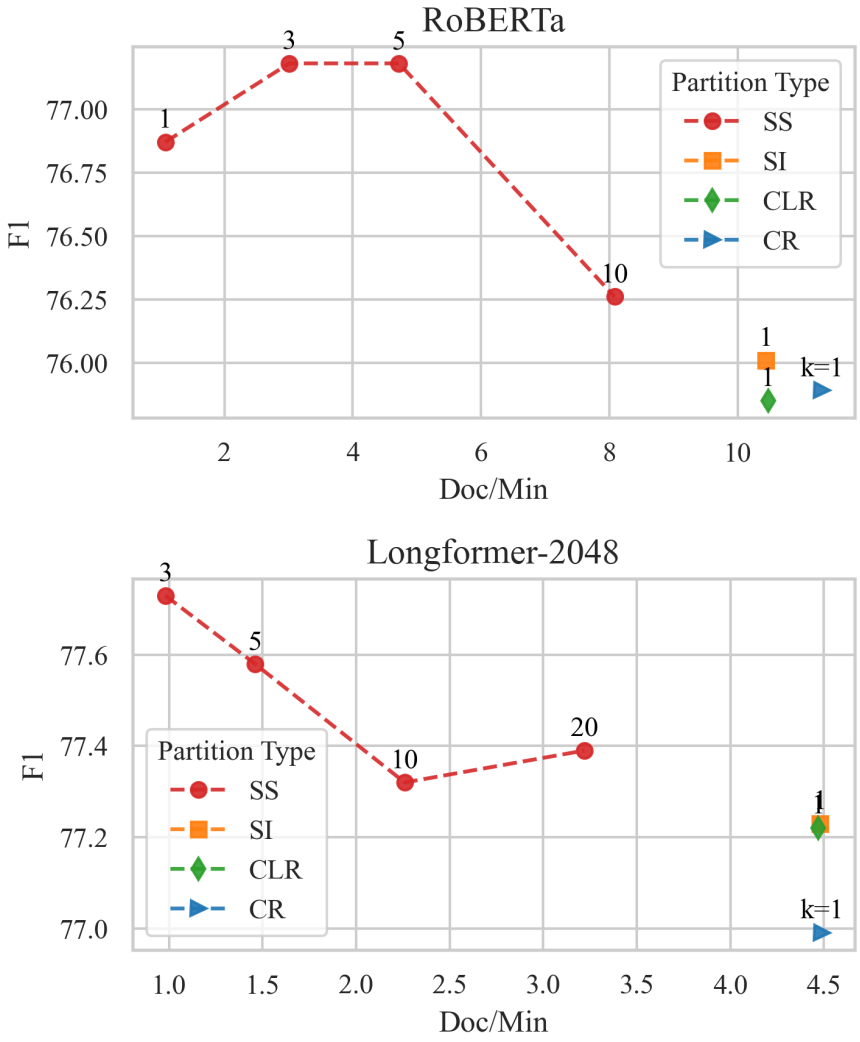

Figure 5.5: Comparison of WeSWin:RoBERTa and WeSWin:Longformer-2048 models (trained and tested on Wiki-727k) in terms of runtime efficiency (Doc/Min) across different document partitioning methods.

Runtime Efficiency of Transformers. Particularly, Figure 5.5 illustrates the F1 accuracy against inference runtime, comparing the performance of the single-task WeSWin:RoBERTa and WeSWin:Longformer-2048 transformer models. The RoBERTa model has a smaller sequence window of 512 tokens, whereas the Longformer-2048 model has a larger input size fixed at 2048 tokens. The longer input size of Longformer-2048 allows for attention over a

larger context, albeit at the expense of slower inference.

In both models, a greater overlap in document partitioning generally results in higher accuracy. When comparing the lightest partitioning baseline for each model, denoted as CR-1, RoBERTa is approximately 2.5 times more efficient than Longformer-2048, with respective inference speeds of 11.31 and 4.49 documents per minute (Doc/Min).

In a multi-prediction SS-k partitioning setting, the difference in runtime becomes more pronounced, with RoBERTa achieving an inference speed of 4.72 Doc/Min while Longformer-2048 achieves 1.46 Doc/Min. This demonstrates that although Longformer-2048 achieves higher F1 accuracy compared to the smaller-context RoBERTa, it is 3 times slower in inference when using comparably-overlapped partitioning methods.

It is worth noting that in this experiment, uniform (as opposed to weighted) aggregation was used for the aggregation of multi-prediction sentences in SI-k and SS-k partitioning methods. Otherwise, RoBERTa would become even more competitive with Longformer-2048 in terms of F1 accuracy when appropriate positional weighting functions are utilized in multi-prediction sentence aggregation, as will be studied next.

Performance of Transformers. In this part, we compare all four transformer encoders—BERT, RoBERTa, Longformer-1024, and Longformer-2048—trained and tested with the single-task WeSWin framework on Wiki-727k. Figure 5.6 illustrates the F1 accuracy trend across these transformer models and multiple partitioning methods. This time, SS-k methods are equipped with the weighted aggregation functions. First of all, the familiar trend of higher accuracy for higher partitioning overlap is visibile across all transformer models. More specifically, the longer-context Longformer-2048 model consistently achieves the

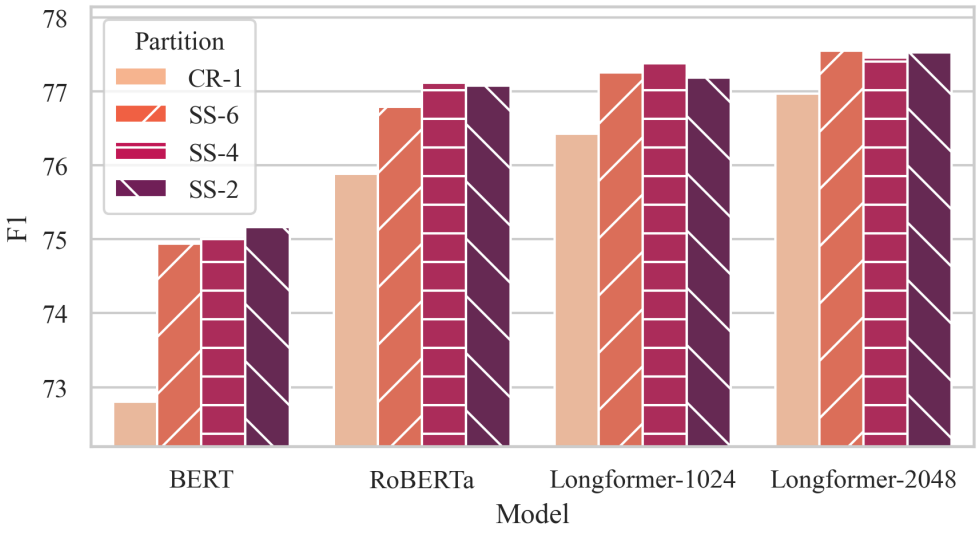

Figure 5.6: Comparison of BERT, RoBERTa, and Longformer transformer models trained and tested on Wiki-727k using our (single-task) WeSWin framework. CR-1 and a few SS-k methods are used for document partitioning.

highest F1 scores across all partitioning strategies, albeit at a proportionally higher inference cost (as discussed in the previous experiment). Similar trends are observed for the smaller-context Longformer-1024, RoBERTa, and BERT models, following relatively closely in performance. In summary, all models benefit from more extensively overlapped partitioning strategies, with the most significant improvements seen when using the SS-k partitioning baselines.

Examining the effect of different partitioning types on individual models, the two smaller context models, BERT and RoBERTa, seem to benefit the most from overlapping partitioning. As the transformer context availability increases, as seen with Longformer-1024 and Longformer-2048, the gains from increased overlap become less pronounced. The smaller gains observed in longer context models are due to the fact that as the sequence length increases, more documents tend to fit entirely within the predefined sequence length. This reduces the need for partitioning, regardless of the partitioning method used.

It is worth noting that RoBERTa starts at a much lower F1 score in the CR-1 baseline partitioning setting compared to the Longformer models. However, with sufficient overlap in sequence generation, especially in the SS-k case, RoBERTa not only performs nearly competitive with the Longformer models but also achieves the best F1 to runtime ratio as discussed earlier. This makes it a suitable choice when faster inference is crucial, particularly given that Longformer models require significantly longer training times as well.

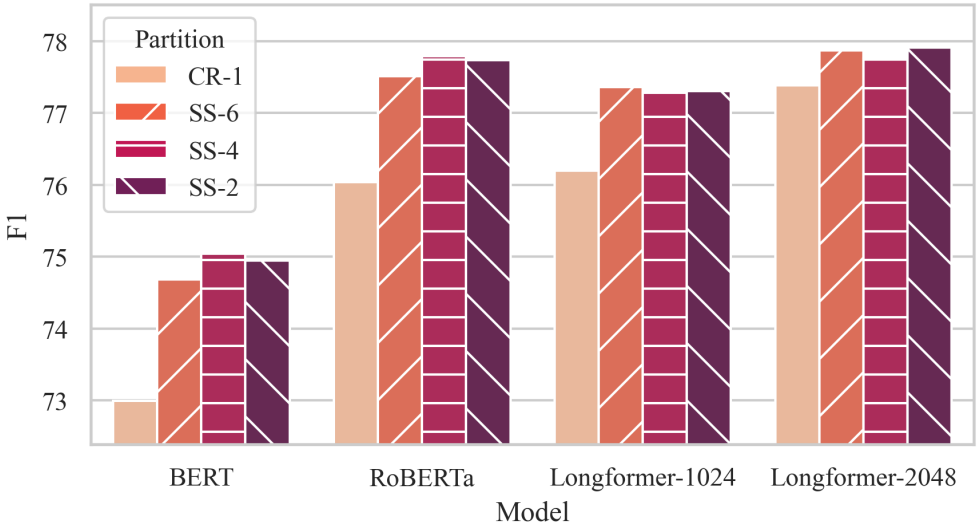

Figure 5.7: Comparison of BERT, RoBERTa, and Longformer transformer models trained and tested on Wiki-727k using our multi-task WeSWin-Ret framework. CR-1 and a few SS-k methods are used for document partitioning.

Interestingly, when training the transformer models in a multi-task setting, referred to as WeSWin-Ret, the lighter RoBERTa model stands even closer to the costlier Longformer-2048 model in terms of F1 accuracy. As illustrated in Figure 5.7, RoBERTa model outperforms the 1024-token version of Longformer and performs super competitively to the 2048-token Longformer. In other words, WeSWin-Ret:RoBERTa may achieve almost the same accuracy as WeSWin-Ret:Longformer-2048 while being three times more efficient in inference. In the next section, we compare the two single and multi-task frameworks in more details.

5.3.5 Multi-task Training of Transformer Models

In Section 4.3, we introduced our multi-task training framework, called WeSWin-Ret, that co-trains document segmentation with auxiliary sentence retrieval task using a shared transformer architecture. Multiple pre-trained transformer encoders, including BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and Longformer (Beltagy et al., 2020), are jointly fine-tuned on Wiki-727k as the segmentation task and NLQuAD as the auxiliary sentence retrieval task. The fine-tuned WeSWin-Ret model is evaluated on 1k random documents from the test split of Wiki-727k. Notably, all previously proposed techniques, such as different document partitioning methods and weighting functions are used in our multi-task framework as well.

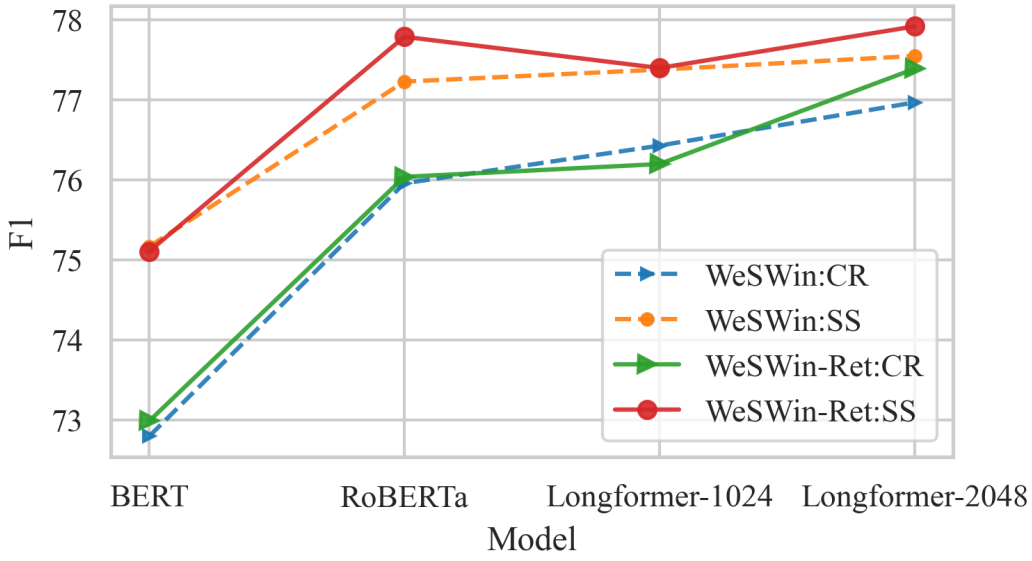

Figure 5.8: Comparison of WeSWin and WeSWin-Ret transformer baselines trained and tested on Wiki-727k. Results of both CR-1 and (the best) SS-k partitioning methods are shown for each model.

For a head-to-head comparison of our single (WeSWin) versus multi-task (WeSWin-Ret) models, Figure 5.8 illustrates the performance for individual transformer baselines for ei-

ther framework. Out of four transformer baselines, RoBERTa and Longformer-2048 significantly benefited from multi-task training with sentence retrieval, with the highest gain seen in RoBERTa. Multi-task BERT and Longformer-1024, however, performed competitively with their single-task counterparts.

Model	**Inference Config**			**Wiki-727k**		
	Partition	**Aggregation**	τ	F1	Prec	Rec
WeSWin:BERT	SS-2	mean:lob_k12_e0.1	0.42	75.17	75.35	74.99
WeSWin-Ret:BERT	SS-1	mean:poly_k8_p2_e0.1	0.39	75.11	75.61	74.62
WeSWin:RoBERTa	SS-5	mean:poly_k10_p2_e0.1	0.45	77.23	77.37	77.08
WeSWin-Ret:RoBERTa	SS-4	mean:lob_k10_e0.1	0.45	77.79	80.22	75.51
WeSWin:Longformer-1024	SS-4	mean:lin_k5_e0.1	0.37	77.38	77.36	77.39
WeSWin-Ret:Longformer-1024	SS-1	mean:poly_k8_p2_e0.1	0.44	77.4	76.28	78.57
WeSWin:Longformer-2048	SS-6	mean:lob_k10_e0.1	0.41	77.55	78.11	77.01
WeSWin-Ret:Longformer-2048	SS-3	mean:lob_k5_e0.1	0.47	77.92	81.34	74.77

Table 5.5: Comparison of underlying transformer architectures used as backbones once for either WeSWin and WeSWin-Ret frameworks, trained and tested on Wiki-727k. Partition, Aggregation, and τ columns respectively represent document partitioning method, sentence aggregation method, and decision threshold, collectively denoted as the validation-tuned inference configuration.

More specifically, as shown in Table 5.5, multi-task training can significantly improve the F1 accuracy of RoBERTa from 77.23 to 77.79 and that of Longformer-2048 from 77.55 to 77.92. However, for the other two transformer settings, BERT and Longformer-1024, single-task and multi-task frameworks remain competitive in terms of F1 score. For each transformer, the best configuration of hyper-parameters, i.e., the optimal document partitioning method, weighting function, and the final decision threshold (τ), is determined through hyper-parameter search using 1k random samples from the validation set. For instance, the best aggregation function for WeSWin-Ret:RoBERTa with SS-4 partitioning method is the loss-based weighting function with $k = 10$ and $\varepsilon = 0.1$, denoted as $lob(.|k = 10, \varepsilon = 0.1)$ as introduced in Section 4.2.3. The hyper-parameter settings and specifics of the search space for each hyper-parameter is further explained in Appendix A.

5.3.6 Transfer Learning to Domain Settings

In this section, we explore the transfer learning capabilities of our proposed models by evaluating them on WikiSection dataset (Arnold et al., 2019). The WikiSection dataset contains segmented Wikipedia articles in domain settings. Two significant subsets are "en_city" with approximately 13.7k training articles on city-related topics, and "en_disease" with around 2.5k medical and health-related training documents. Due to the small size of these datasets, pre-training on the larger Wiki-727k dataset, which contains 582k training documents, may be beneficial. We utilize the pre-trained Wiki-727k checkpoints for RoBERTa and Longformer-2048 transformers to perform fine-tuning on the en_city and en_disease datasets independently. Finally, we use the fine-tuned checkpoints to evaluate the test sets. WeSWin framework with overlapping document partitioning and sentence aggregation methods is used in inference.

Model	**en_city**			**en_disease**		
	F1	**Prec**	**Rec**	**F1**	**Prec**	**Rec**
WeSWin:RoBERTa	**86.85**	89.13	84.68	**77.39**	76.76	78.03
WeSWin:Longformer-2048	86.77	89.64	84.08	76.68	75.65	77.73

Table 5.6: WeSWin results on en_city and en_disease datasets. Checkpoints are first pre-trained on Wiki-727k and then fine-tuned on domain datasets.

On en_city dataset, as shown in Table 5.6, our WeSWin:RoBERTa model achieves F1 score of 86.85, slightly outperforming the costlier WeSWin:Longformer-2048 model with F1 score of 86.77. On en_disease, WeSWin:RoBERTa and WeSWin:Longformer-2048 models achieve F1 scores of 77.39 and 76.68, this time, showing a clear superiority of RoBERTa despite being the lighter baseline. As previously shown in Table 5.4, our baselines outperform the SOTA methods on en_city and en_disease datasets, proving the effectiveness of our proposed contributions in domain settings as well.

In the next chapter, we further test our proposed segmentation model when incorporated in a downstream RAG application.

5.3.7 RAG with Semantic Chunking

A segmentation model can be a direct application for Information Retrieval or Retrieval-augmented Question Answering (RAG) tasks. In this section, we evaluate the effectiveness of our segmentation model in the downstream task of Long-form Question Answering through the RAG framework. Specifically, we employ our semantic chunker to perform Retrieval-augmented Question Answering on the NLQuAD dataset. The NLQuAD dataset consists of sufficiently long documents annotated with question and long-form answer pairs.

The retrieval process involves segmenting the knowledge documents into manageable chunks of text, which are stored in a knowledge database. An embedding vectore for each chunk is constructed using OpenAI Embeddings[3] (OpenAI, 2023) and stored alongside the corresponding chunk in the database. Given an input query or question at inference time, we take its embedding and perform a FAISS search (Johnson et al., 2019) over the embedding database to find the top-k chunks with the highest cosine similarity scores. Once the matching chunks are obtained through the retrieval phase, we form a generative prompt using the input query and the retrieved chunks as context, asking for the answer. In this regard, we use ChatGPT[4] (OpenAI, 2023) as a generative model to extract the span of text from the context chunks that best answers the given question. Finally, the predicted answer is compared against the ground-truth answer in NLQuAD using ChatGPT as judge, providing matching

[3]text-embedding-3-large (`https://platform.openai.com/docs/models/embeddings`)
[4]GPT-4o (`https://platform.openai.com/docs/models/gpt-4o`)

scores out of 10.

More specifically, we create a segment/chunk database from 1k documents from NLQuAD and use 50 question-answer pairs for QA evaluation. FAISS with OpenAI Embeddings is utilized to obtain the top k chunks for individual questions. A QA prompt is then built from the concatenation of these k chunks, referred to as context, and the query. The QA prompt is given to ChatGPT to extract the answer. Finally, we compare the candidate answers to the ground truth using ChatGPT as the judge, providing a matching score out of 10.

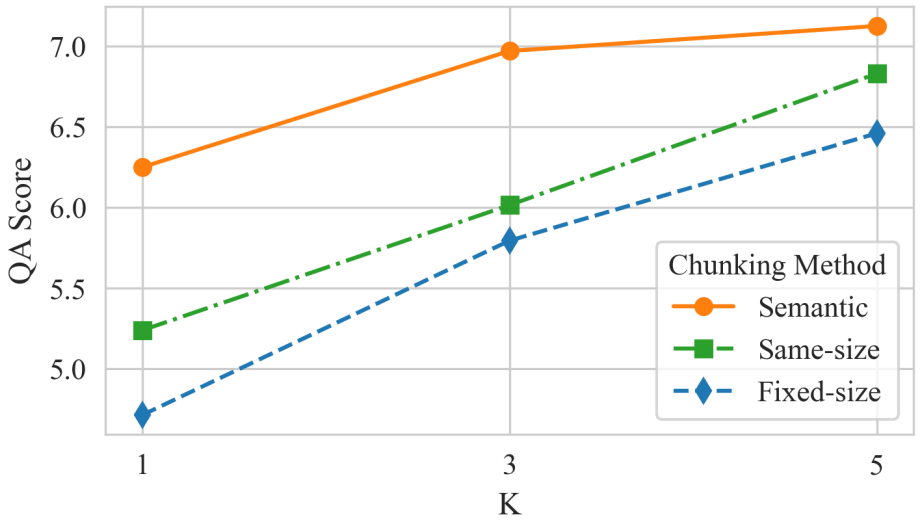

Figure 5.9: Comparison of our Semantic chunker against two Fixed and Same-size chunking baselines, indirectly evaluated as part of a RAG framework evaluated on NLQuAD's QA pairs.

In Figure 5.9, we evaluate our semantic chunker against three retrieval settings. We use WeSWin:RoBERTa as the semantic chunker in this experiment to segment NLQuAD documents into chunks. Notably, this model is trained only on Wiki-727k for the single task of document segmentation and has not seen any data from NLQuAD (as opposed to our WeSWin-Ret checkpoints). We analyze the impact of varying the number of retrieved chunks (k) on the quality of the generated answers, providing insights into the optimal configuration

for the RAG framework. With larger k values, the generator has access to more related chunks for a more informed decision, which is expected to result in an improved QA score.

As baselines, we designed two fixed-length chunking methods. In the *fixed-size* chunking method, documents are segmented into chunks with an upper bound of 100 words. In the *same-size* chunking method, we set the upper bound to the average segment size of our semantic chunker (in terms of words). It is worth noting that the same-size chunker is a fairer baseline for our semantic chunker since statistically longer chunks provide more context to the generator when a fixed number of retrieved chunks (k) is used.

The results demonstrate that our chunker outperforms the baselines across different values of k used in retrieval. Particularly, with only 1 retrieved chunk, our chunker significantly outperforms the fixed-size and same-size chunkers by achieving a QA score of 6.25, while they achieve QA scores of 5.24 and 4.71, respectively. As k is increased to 3 and 5, the QA score improves for all chunking methods. With 5 retrieved chunks, the same-size chunker gets closer but still lags behind our semantic chunker, with respective QA scores of 6.83 and 7.12.

In conclusion, our semantic chunker particularly stands out in small-context settings with only 1 retrieved chunk. This makes the RAG model more efficient by providing a concise yet coherent context, capable of delivering the necessary information for question answering without overwhelming the generator. By focusing on the most relevant chunk, our model may reduce the computational load while maintaining superior accuracy for the generated answers. This efficiency is crucial for applications where quick and precise responses are needed, such as real-time question-answering systems or situations with limited computational resources. Furthermore, the ability to perform well with minimal context underscores

the robustness and effectiveness of our semantic chunker, making it a valuable tool for a wide range of information retrieval tasks.

Chapter 6

Application

In this chapter, we evaluate our segmentation model on a real-world domain dataset. We utilize our segmentation model to chunk an automotive user manual called AutoManual, obtained from iNAGO, into manageable pieces. Our semantic chunker will be a direct application for downstream tasks of Knowledge Retrieval and Retrieval-Augmented Generation in their conversational pipeline.

iNAGO[1] is a company that specializes in developing intelligent conversational assistant platforms designed to enhance human-computer interaction. They focus on creating AI-driven conversational assistants for a variety of applications. These assistants can be integrated into automotive systems, consumer electronics, and smart home devices. Notably, their technology is employed by major automotive and industrial corporations.

The role of the chunker is crucial in such knowledge extensive systems, as it breaks down

[1] https://www.inago.com

extensive and complex knowledge documents into smaller, manageable segments. These segments are essential for efficient Knowledge Retrieval, enabling the system to provide accurate and relevant information swiftly. Additionally, in Retrieval-Augmented Generation, the chunker ensures that the generated responses are contextually appropriate and precise by providing well-defined chunks of information to the generative model.

6.1 Dataset

User manuals consist of multiple chapters, each introducing different aspects of the product. An automotive car manual typically consists of chapters about vehicle operation, maintenance, safety features, technical specifications, and troubleshooting. The manual dataset, referred to as AutoManual, consists of a hierarchy of chapters, sections, sub-sections, paragraphs, and sub-elements such as text, list items, tables, and figures. The dataset is already labeled by a human with segmentation information indicating where the segment boundaries are. We parse the user manual into multiple documents. Input documents are derived from the flattened *chapter* elements from the manual, and they are processed independently.

For this dataset, we include section and sub-section titles among sentences as they provide crucial information regarding the flow of topic shift. Note that ground-truth segment boundaries are not necessarily aligned with section or sub-section titles. The labels are provided by a human expert and may span across more than one section or sub-section depending on the semantics of the content. Therefore, it is safe to include title information in the training and inference phases.

According to Table 6.1, AutoManual consists of 11 chapters, 8 of which are used for

Dataset	#Train	#Validation	#Test	#Sentence/Doc	#Sentence/Segment	#Word/Sentence
AutoManual	12	1	2	1035.12	8.92	12.75

Table 6.1: Cardinality and length statistics of AutoManual as a text segmentation dataset.

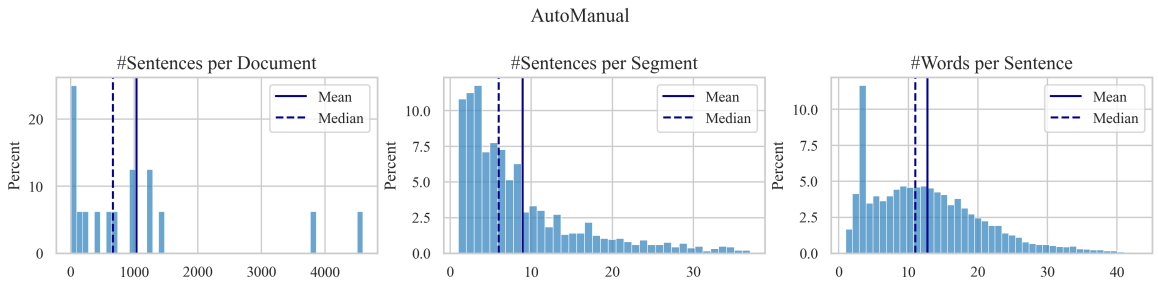

Figure 6.1: Length distribution of AutoManual dataset.

training, 1 for validation, and 2 chapters for testing. AutoManual documents are significantly larger compared to Wikipedia-based articles discussed earlier. Documents come with an average sentence size of 1035. Each segment has an average of 8.92 sentences, and each sentence has an average of 12.75 words. Figure 6.1 shows histogram plots on length distribution of AutoManual.

Using our proposed sliding window document partitioning and sentence aggregation techniques, we are able to efficiently process these long documents while achieving competitive segmentation scores. In the next section we provide more details on the segmentation model used for this task.

6.2 Experiment

We propose to fine-tune and evaluate our WeSWin:RoBERTa and WeSWin:Longformer-2048 models on AutoManual data. Training from scratch on such a small dataset dataset may not

be feasible or as effective as using a pre-trained segmentation checkpoint. Therefore, we fine-tune over our RoBERTa and Longformer-2048 checkpoints, pre-trained on 582k training documents from Wiki-727k.

Normally, we use CR-1 document partitioning for sequence generation in training phase. However, due to lack of training samples in this setting, we propose to perform data augmentation in training by doing SS-4 partitioning instead of CR-1. In SS-k, a sequence window is slid over a document with sentence stride of 4 until the whole document is covered. In CR-1, however, consecutive sequences have a sentence overlap of 1, which is only active in the following sequence. To see the impact of data augmentation in low-resource settings, we train two checkpoints for each transformer model, one trained with CR-1 and the other trained with SS-4 partitioning method. In either case, we use SS-k document partitioning in inference to fully utilize the context.

Inference is done over two held-out chapters of AutoManual. Best document partitioning and sentence aggregation methods for inference are decided through hyper-parameter tuning on the validation set. We measure the F1 score of sentence predictions compared against the ground-truth segmentation labels in sentence-level.

6.3 Results

We evaluated our RoBERTa and Longformer-2048 checkpoints, fine-tuned on AutoManual data, as shown in Table 6.2. Each transformer was trained once with the CR-1 partitioning method and once with the SS-4 partitioning method. To further highlight the effect of fine-tuning, we also evaluated our pre-trained RoBERTa and Longformer-2048 checkpoints

without any fine-tuning on AutoManual.

Moreover, we developed multiple unsupervised baselines and evaluated them on AutoManual data for further comparison. For unsupervised baselines, we used Sentence Transformers (Reimers and Gurevych, 2019) and OpenAI Embeddings (OpenAI, 2023) models to capture sentence representations within each document. For each of the $n-1$ potential cut points in a document with n sentences, we measured topic shift probabilities using cosine similarity over the embedding vectors of the surrounding context. Specifically, we took a window of k sentences on each side of each candidate point and calculated the cosine similarity over crossing pairs of sentences' embedding vectors. We reduced the similarity pairs into a scalar using the best of "mean," "max," or "min" pooling, and reversed it to get an aggregate probability of semantic shift for each candidate. Finally, a threshold τ was used to derive binary labels for individual candidates. All hyper-parameters, including k, τ, and pooling functions, were tuned over the validation set of AutoManual.

Setting	**Model**	**AutoManual**		
		F1	**Prec**	**Rec**
Unsupervised	SentenceTransformers: all-MiniLM-L6-v2	36.13	28.79	48.48
	SentenceTransformers: all-mpnet-base-v2	37.23	28.88	52.38
	OpenAIEmbeddings: text-embedding-3-small	37.67	29.15	53.25
	OpenAIEmbeddings: text-embedding-3-large	37.90	33.44	43.72
Pre-trained on Wiki-727k	WeSWin:RoBERTa [CR-1]	41.72	29.11	73.5
	WeSWin:Longformer-2048 [CR-1]	41.55	30.1	67.1
Pre-trained on Wiki-727k, Fine-tuned on AutoManual	WeSWin:RoBERTa [CR-1]	81.21	87.5	75.76
	WeSWin:RoBERTa [SS-4]	**83.45**	90.4	77.49
	WeSWin:Longformer-2048 [CR-1]	80.49	93.68	70.56
	WeSWin:Longformer-2048 [SS-4]	80.58	91.71	71.86

Table 6.2: Comparison of our (pre-trained and) fine-tuned WeSWin model results against unsupervised methods on AutoManual dataset.

In fine-tuned settings, WeSWin:RoBERTa achieved an F1 score of 81.21 when trained with CR-1. When trained with SS-4 partitioning, WeSWin:RoBERTa outperformed the CR-1 baseline by more than 2.7%, reaching an F1 score of 83.45. However, for WeSWin:Longformer-

2048, SS-4 baseline only slightly improved over CR-1, with F1 scores of 80.58 and 80.49, respectively. Interestingly, WeSWin:RoBERTa trained with SS-4 partitioning stood out as our most competitive baseline, while being three times more efficient than its Longformer-2048 counterpart.

Our WeSWin:RoBERTa and WeSWin:Longformer-2048 models in pre-trained settings, with respective F1 scores of 41.72 and 41.55, were vastly outperformed by our fine-tuned models due to the significant differences between the AutoManual data distribution and the Wikipedia data they were trained on. However, they still outperformed the unsupervised baselines. Among the unsupervised methods, two notable Sentence Transformers baselines, ”all-MiniLM-L6-v2” and ”all-mpnet-base-v2,” achieved F1 scores of 36.13 and 37.23, respectively. Slightly improved, the ”text-embedding-3-small” and ”text-embedding-3-large” baselines from OpenAI Embeddings scored 37.67 and 37.90, respectively.

Unsupervised models did not come close in accuracy compared to our supervised baselines, highlighting the importance of supervised token-level interactions in document segmentation. Additionally, the substantial gap between our fine-tuned and non-fine-tuned baselines suggests the importance of fine-tuning on new domains, particularly those with significantly different syntactic and semantic data distributions. In conclusion, our segmentation models have proven competitive in new domains with minimal fine-tuning.

Finally, our semantic chunker model was successfully deployed in iNAGO’s pipeline, serving as the basis for multiple downstream tasks. As demonstrated in Subsection 5.3.7, identifying cohesive segments within a document using a semantic chunker can improve the quality and efficiency of downstream RAG tasks.

Chapter 7

Conclusion

In this work, we have introduced several key contributions to the field of neural text segmentation. We presented two text segmentation frameworks, denoted as WeSWin and WeSWin-Ret, each trained and tested using BERT, RoBERTa, and Longformer transformers. Our methods significantly outperformed existing baselines across all models with different sequence sizes on multiple segmentation benchmarks.

On the Wiki-727k dataset, our approach significantly outperformed existing BERT and RoBERTa-based baselines by more than 10%. Our Longformer-2048 baseline achieved the highest F1 score on Wiki0727k, outperforming the existing baseline by about 1%. Our RoBERTa baseline is almost as competitive as our Longformer-2048 baseline while being at least 3 times more efficient in both training and inference. This suggest smaller-context models may be as competitive as longer ones in text segmentation settings using proper sliding window techniques.

On the en_city and en_disease datasets, our RoBERTa-based model surpassed the Longformer baseline on en_city by 1.8%, while matching its performance on en_disease. This improvement was despite the fact our RoBERTa model is using only a quarter of the input sequence size of its Longformer competitor. This highlights the efficiency of our methods in optimizing context usage with smaller models.

Our multi-task trained model outperformed our single-task one in most comparisons, showing the effectiveness of using sentence retrieval as an auxiliary task for document segmentation. Particlarly, on Wiki-727k benchmark, our multi-task RoBERTa performed 0.7% better than its single-task counterpart.

We further tested our segmentation model in a downstream RAG task. We employed our semantic chunker to perform retrieval-based question answering on NLQuAD dataset in a RAG framework, outperforming the fixed-length chunking baselines by 19%. Finally, We employed and tested our segmentation model on an industrial dataset in automotive domain. Achieving high segmentation performance, our model is readily transferable to new domains with minimal fine-tuning.

7.1 Future Works

Future work may focus on further optimizing the model and exploring its application to other natural language processing tasks.

In the multi-task training setting, incorporating a larger sentence retrieval dataset as the auxiliary dataset may further improve segmentation results. The NLQuAD dataset, currently

used for the auxiliary retrieval task, contains only a fraction of the documents found in the Wiki-727k dataset.

Additionally, the framework can be extended by training over multiple document-level sentence-oriented tasks, enabling it to cover a broader range of tasks. With enough data in multi-task pre-training, our sliding window framework can be readily applicable to new sentence-oriented tasks with minimal fine-tuning. This is especially beneficial when the labelled data is limited in the new domain.

Given the small size of existing segmentation models, in terms of the number of parameters, they may struggle to perform well in zero-shot settings across different domains. For instance, the content or the desired labels within different domains may have varying granularity in terms of topic shift definition. Although this issue may be addressed in some cases by using a tuned decision threshold, there is a compelling need to improve the zero-shot capabilities of such models in various settings.

Furthermore, the optimal weight assignment is currently achieved through hyper-parameter search over the validation set, which may be time-consuming. Therefore, as a future step, making the aggregation process more end-to-end could streamline this step and improve overall efficiency.

Bibliography

Uri Alon, Frank Xu, Junxian He, Sudipta Sengupta, Dan Roth, and Graham Neubig. Neuro-symbolic language modeling with automaton-augmented retrieval. In *International Conference on Machine Learning*, pages 468–485. PMLR, 2022.

Sebastian Arnold, Rudolf Schneider, Philippe Cudré-Mauroux, Felix A. Gers, and Alexander Löser. Sector: A neural model for coherent topic segmentation and classification. *Transactions of the Association for Computational Linguistics*, 7:169–184, 2019. doi: 10.1162/tacl_a_00261.

Doug Beeferman, Adam L. Berger, and John D. Lafferty. Text segmentation using exponential models. *ArXiv*, cmp-lg/9706016, 1997. URL https://api.semanticscholar.org/CorpusID:8620787.

Anas Belfathi, Nicolas Hernandez, and Laura Monceaux. Enhancing pre-trained language models with sentence position embeddings for rhetorical roles recognition in legal opinions. In *ASAIL@ICAIL*, 2023. URL https://api.semanticscholar.org/CorpusID:259938751.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document trans-

former. *ArXiv*, abs/2004.05150, 2020. URL https://api.semanticscholar.org/CorpusID:215737171.

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

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, T. W. Hennigan, Saffron Huang, Lorenzo Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:244954723.

Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, Li Deng, and Bhaskar Mitra. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkg-mA4FDr.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancou-

ver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL https://aclanthology.org/P17-1171.

Xilun Chen, Kushal Lakhotia, Barlas Oguz, Anchit Gupta, Patrick Lewis, Stan Peshterliev, Yashar Mehdad, Sonal Gupta, and Wen-tau Yih. Salient phrase aware dense retrieval: Can a dense retriever imitate a sparse one? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 250–262, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.19. URL https://aclanthology.org/2022.findings-emnlp.19.

Sangwoo Cho, Kaiqiang Song, Xiaoyang Wang, Fei Liu, and Dong Yu. Toward unifying text segmentation and long document summarization. *ArXiv*, abs/2210.16422, 2022. URL https://api.semanticscholar.org/CorpusID:253237148.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Reshmi Ghosh, Harjeet Singh Kajal, Sharanya Kamath, Dhuri Shrivastava, Samyadeep Basu, Hansi Zeng, and Soundararajan Srinivasan. Topic segmentation of semi-structured and unstructured conversational datasets using language models. *ArXiv*, abs/2310.17120, 2023. URL https://api.semanticscholar.org/CorpusID:264490422.

Goran Glavas and Swapna Somasundaran. Two-level transformer and auxiliary coherence

modeling for improved text segmentation. *CoRR*, abs/2001.00891, 2020. URL http://arxiv.org/abs/2001.00891.

Goran Glavas, Federico Nanni, and Simone Paolo Ponzetto. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 125–130, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-2016. URL https://aclanthology.org/S16-2016.

Goran Glavas, Ananya Ganesh, and Swapna Somasundaran. Training and domain adaptation for supervised text segmentation. In *Workshop on Innovative Use of NLP for Building Educational Applications*, 2021. URL https://api.semanticscholar.org/CorpusID:233365337.

Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. Recurrent chunking mechanisms for long-text machine reading comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://api.semanticscholar.org/CorpusID:218674216.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *Conference on Empirical Methods in Natural Language Processing*, 2021. URL https://api.semanticscholar.org/CorpusID:237452184.

Marti A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, mar 1997. ISSN 0891-2017.

Hakan Inan, Rashi Rungta, and Yashar Mehdad. Structured summarization: Unified text segmentation and segment labeling as a generation task. *ArXiv*, abs/2209.13759, 2022. URL https://api.semanticscholar.org/CorpusID:252567766.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022a. ISSN 2835-8856. URL https://openreview.net/forum?id=jKN1pXi7b0.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot Learning with Retrieval Augmented Language Models. *arXiv*, 2022b. URL http://arxiv.org/abs/2208.03299.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *arXiv*, 2023.

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

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HklBjCEKvH.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 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, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2075. URL https://aclanthology.org/N18-2075.

Jeonghwan Lee, Jiyeong Han, Sunghoon Baek, and Min Jae Song. Topic segmentation model focusing on local context. *ArXiv*, abs/2301.01935, 2023. URL https://api.semanticscholar.org/CorpusID:255440640.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL https://api.semanticscholar.org/CorpusID:198953378.

Michal Lukasik, Boris Dadachev, Kishore Papineni, and Gonccalo Simões. Text segmen-

tation by cross segment attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4707–4716, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.380. URL https://aclanthology.org/2020.emnlp-main.380.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. Text and code embeddings by contrastive pre-training, 2022.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.146. URL https://aclanthology.org/2022.findings-acl.146.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855, Abu Dhabi, United Arab Emirates, December 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.669. URL https://aclanthology.org/2022.emnlp-main.669.

Barlas Oguz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksan-

dra Piktus, Xilun Chen, Sebastian Riedel, Scott Yih, Sonal Gupta, and Yashar Mehdad. Domain-matched pre-training tasks for dense retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1524–1534, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.114. URL https://aclanthology.org/2022.findings-naacl.114.

OpenAI. Gpt-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774. arXiv preprint arXiv:2303.08774.

Lev Pevzner and Marti A. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002. doi: 10.1162/089120102317341756. URL https://aclanthology.org/J02-1002.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 2023. URL https://arxiv.org/abs/2302.00083.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen tau Yih. Replug: Retrieval-augmented black-box language models. *ArXiv*, abs/2301.12652, 2023. URL https://api.semanticscholar.org/CorpusID:256389797.

Gennady Shtekh, Polina Kazakova, Nikita Nikitinsky, and Nikolay Skachkov. Applying topic segmentation to document-level information retrieval. In *Central and Eastern European Software Engineering Conference in Russia*, 2018. URL https://api.semanticscholar.org/CorpusID:54444448.

Amir Soleimani, Christof Monz, and Marcel Worring. NLQuAD: A non-factoid long question answering data set. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1245–1255, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.106. URL https://aclanthology.org/2021.eacl-main.106.

Yiping Song, Lili Mou, Rui Yan, Li Yi, Zinan Zhu, Xiaohua Hu, and Ming Zhang. Dialogue session segmentation by embedding-enhanced texttiling. *CoRR*, abs/1610.03955, 2016. URL http://arxiv.org/abs/1610.03955.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. *arXiv preprint arXiv:2212.09741*, 2023.

Jörg Tiedemann and Jori Mur. Simple is best: Experiments with different document segmentation strategies for passage retrieval. In *Coling 2008: Proceedings of the 2nd Workshop on Information Retrieval for Question Answering*, IRQA '08, page 17–25, USA, 2008. Association for Computational Linguistics. ISBN 9781905593552.

Liang Wang, Sujian Li, Xinyan Xiao, and Yajuan Lyu. Topic segmentation of web documents with automatic cue phrase identification and blstm-cnn. In *NLPCC/ICCPOL*, 2016. URL https://api.semanticscholar.org/CorpusID:2286766.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Christian Wartena. Segmentation strategies for passage retrieval from internet video using speech transcripts. *J. Digit. Inf. Manag.*, 11(6):400–408, 2013. URL http://dline.info/fpaper/jdim/v11i6/3.pdf.

Jinxiong Xia and Houfeng Wang. A sequence-to-sequence approach with mixed pointers to topic segmentation and segment labeling. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023. URL https://api.semanticscholar.org/CorpusID:260499852.

Yi Xu, Hai Zhao, and Zhuosheng Zhang. Topic-aware multi-turn dialogue modeling. In *AAAI Conference on Artificial Intelligence*, 2020. URL https://api.semanticscholar.org/CorpusID:221971391.

Hai Yu, Chong Deng, Qinglin Zhang, Jiaqing Liu, Qian Chen, and Wen Wang. Improving long document topic segmentation models with enhanced coherence modeling. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://api.semanticscholar.org/CorpusID:264289093.

Qinglin Zhang, Qian Chen, Yali Li, Jiaqing Liu, and Wen Wang. Sequence model with self-adaptive sliding window for efficient spoken document segmentation. *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 411–418, 2021. URL https://api.semanticscholar.org/CorpusID:236134477.

Appendices

A Hyperparameter Setting

We train our transformer baselines on Wiki-727k with a learning rate of 1e-5 for at most 3 epochs, utilizing early stopping. For en_city, en_disease, and AutoManual, we use a learning rate of 5e-6 for 5 epochs, also with early stopping. The batch size is set to 8 for BERT and RoBERTa models, and 4 for Longformer baselines. BERT and RoBERTa models are trained using an NVIDIA GTX 1080 Ti GPU, while Longformer baselines are trained using an NVIDIA RTX A6000 GPU.

We set inference hyperparameters according to performance over the validation set. To choose the best document partitioning method, we test and compare CR-1 and SS-k methods by trying multiple stride values (k), including 6, 4, or 2. When compiling multiple predictions coming from overlapping sequences, we search for the best weighting function as a hyper-parameter by comparing uniform, linear, polynomial, and loss-based weighting functions. In this regard, ε is set to 0.1, k is set to either 5, 8, 10, or 12, and p is set to 2 (in the case of the polynomial function). Finally, we explore the decision threshold in the range of [0.3, 0.7] to derive binary sentence labels from ultimate output probabilities.