Text Segmentation to Improve Long Document Understanding

Alex Dong
awd275@nyu.edu

Shwetanshu Singh
ss11404@nyu.edu

Andrew Yeh
ay1626@nyu.edu

Abstract

Transformer models are unable to handle long documents because of the attention module's quadratic time complexity with regards to sequence length. Previous adaptations of the transformer to long document problems involve using a sparse attention mechanism that spans the document, or using a multi-level hierarchical model. These works have focused on the model architecture and attention mechanism without regards to the quality of segmentation. We introduce a new segmentation procedure using BERT's next sentence prediction probabilities to create non-overlapping segments (to facilitate well-contained topic areas within each segment) as well as segments that overlap on the sentence level. We then implement a hierarchical transformers model using those segmentation breakpoints to find consistent improvements of over 5% accuracy in downstream classification tasks when compared to the baseline segmentation method.

1 Introduction

Transformer models as first introduced by Vaswani et al. (2017) have improved natural language understanding dramatically. However, classic transformer models like BERT have been limited to relatively short inputs (512 tokens), due to the expensive n^2 computation in the attention module (Devlin et al., 2018). Previous work has attempted to address this limitation in two ways:

1. Work with the token limit, and segment the long document such that each segment has fewer than 512 tokens.

2. Break the token limit by sparsifying the connections in the attention module.

Our proposal aims to extend a method in the first category by intelligently segmenting the text into distinct segments. In particular, we extend the Hierarchical Transformer method proposed by Pappagari et al. (2019).

The Hierarchical Transformer splits the document into segments of length 200 tokens, finds embeddings for each segment, and feeds the embeddings into a higher-level model. We call fixed length window segmentation the "naive segmentation approach" because the segmentation is arbitrary and ignores linguistic organization like sentence breaks.

We see room for improvement over the naive segmentation approach because of the nature of long documents. If a meaningful span of text is split up midway by the naive segmentation approach, both resulting segment embeddings lose potential meaning. Splitting at the wrong place can separate a pronoun from its antecedent, split the assertion from the evidence, and dilute the meaning of an individual segment, making the downstream tasks more difficult.

To bring intelligent segmentation to our handling of long documents, we propose segmenting the document using BERT's Next Sentence Prediction (NSP) capability with the hypothesis that abrupt breaks in the topic of discussion that would cause BERT to not link two sentences together are also natural break points for our segments.

We find that using this NSP segmentation strategy yields 4 – 5% higher accuracy over the baseline and stronger performance than tuned naive segmentation strategies.

2 Related Work

Dua et al. (2019) highlight a weakness in document comprehension systems that helps motivate our area of study. They create an adversarial dataset that requires the model to resolve references over multiple input positions and keep track of long-range connections and find that the best systems only attain a 32.7% F1 score compared to expert human performance of 96.0%. The paper finds that current models struggle with questions whose answers are spread out among the document; this

motivates our study of long documents where resolving references over multiple, distant positions is necessary for comprehension but not addressed through traditional transformers.

Pappagari et al. (2019) demonstrate the use of a set of hierarchical transformers to process long documents with one layer to create an embedding for each segment, and another layer to compile the segment embeddings into a document embedding. They apply this methodology to three different tasks and get then-state of the art results on the Fisher topic classification task of 10-minute long phone calls, the CSAT prediction task on transcripts of customer service calls, and the classification of news articles into one of 20 topics.

Beltagy et al. (2020), Child et al. (2019), and Roy et al. (2020) take the second approach to long documents discussed in Section 1 by building a sparse attention mechanism that scales linearly with document length instead of quadratically. Beltagy et al. (2020) use a sliding window of attention so that each token can only attend to characters within its window of length $w \in \mathbb{R}$ –this decreases the computational complexity from $O(n^2)$ to $O(n \times w) = O(n)$. However, because these windows overlap, the windowed attention mechanism can incorporate long-range relationships across the document without having to use attention across the whole document. Additionally, the authors incorporate global attention for certain points in the document to improve its flexibility, analogous to shortcut, residual connections for RNNs.

Cer et al. (2018) take a different approach to segmentation by providing a method to create embeddings on the sentence level as opposed to the word level. This lowers the computational complexity of attention across the document. They demonstrate that these sentence level embeddings lead to competitive evaluation scores and higher levels of transfer learning compared to word level embeddings. This motivates us to tailor our segmentation strategy on the sentence-level instead of on the word or character level.

Dai et al. (2019), similarly to Pappagari et al. (2019), adapts a transformer model with fixed length segments and a hierarchical model to process the embeddings of each segment. However, instead of processing the embeddings of each layer independently, the authors follow an RNN-like structure whereby the final hidden state of one segment is the initialization of the hidden state of the next segment. Dai et al. (2019) claim that this process facilitates information flow across segments which theoretically helps capture longer-term dependencies. They also use a character-level language model instead of a word-based model. Dieng et al. (2017) provides some evidence that this RNN-like structure can capture long-term dependencies across documents by integrating the RNN capturing local structure with global, latent topics.

As discussed above, none of the related literature for processing long documents addresses or utilizes the idea of intelligent segmentation for long documents. This presents a gap in the literature which we attempt to address.

3 Methodology

We first provide an overview of the Recurrence over BERT method (RoBERT) in Pappagari et al. (2019). We finetune this method, and then introduce two new segmentation methods, which can directly replace the segmentation method in RoBERT.

3.1 Recurrence over BERT

The RoBERT method proposed in Pappagari et al. (2019) is designed to solve a multi-class long document classification problem. First, the long document input is tokenized. Then, the document is split into segments of 200 tokens with a shift of 50 tokens, such that two consecutive segments have a 150 token overlap. Each segment is embedded by passing the segment through BERT-base, and taking the final hidden state of the [CLS] token. This sequence of embeddings is then passed into a 128-dimensional uni-directional LSTM. Finally, the final hidden state of the LSTM is passed through two fully connected layers with ReLU (64-dimensional) and softmax (dimensions are number of classes).

We keep all design decisions made in the RoBERT model besides the segmentation scheme to more closely compare our results with the baseline implemented by Pappagari et al. (2019) and to isolate the effect of segmentation on performance.

3.2 Segmentation Schemes

3.2.1 Naive Overlap

For the naive overlap segmentation scheme, we use the overlap scheme described in Section 3.1, but finetune the approach further by exploring various sequence and shift lengths. We also define our baseline segmentation scheme to be a naive overlap

with sequence length of 200 and shift length of 50, as in Pappagari et al. (2019)

3.2.2 NSP-only

Our goal with the segmentation task is to split the document into mutually exclusive segments such that each segment addresses a different topic.

We perform the NSP-only segmentation as follows. We first split our input text into sentences using NLTK's Punkt Tokenizer (Kiss and Strunk, 2006). This forms an n-length sequence of sentences. We then slide the BERT NSP model over each consecutive pair of sentences, obtaining an $n-1$ length sequence of NSP-probabilities. We also keep track of how many tokens are in our current segment. For a predefined probability threshold p, whenever the probability in the NSP-probability sequence is less than or equal to p, that index becomes a *split index*. If adding the sentence to the current segment would go over BERT's 512 token limit, then we automatically split. Then, for two split indices i_1, i_2, we form a segment from the (i_1+1)th to i_2th sentences.

3.2.3 NSP with Overlap

We perform the NSP with overlap segmentation as follows. We determine the segments as in the NSP-only segmentation method in Section 3.2.2. Each segment then gains an additional sentence, which is the sentence that follows the last sentence in the segment. The last segment in the document does not gain an additional sentence. As before, we check to make sure adding a sentence to a segment will not increase the number of tokens in the current segment past 512. For example, in a document with three segments, suppose segment A contains sentences 1 and 2; segment B contains sentences 3 and 4; and segment C contains 5, 6, and 7 under NSP segmentation. Then, under NSP with overlap, segment A would contain sentences 1, 2, 3; segment B would contain sentences 3, 4, 5; and segment C would contain sentences 5, 6, 7.

4 Experiments

We provide an implementation on Github [1]. We reference an implementation of the RoBERT model discussed in Section 3.1.[2]

[1] https://github.com/alexwdong/NLUProject

[2] https://github.com/helmy-elrais/RoBERT_Recurrence_over_BERT

4.1 Implementation

For the naive overlap scheme, we vary the sequence length between 100 and 300 in intervals of 50, and the shift length between 0% and 100% (as a proportion of sequence length).

For the NSP-only and NSP with overlap segmentation schemes, we experiment with various probability cutoffs for determining segment breakpoints using the BERT NSP probabilities. We find that varying the threshold between 0.5 and 0.95 does not result in very different segment breakpoints (on average, it would result in 0.17 additional splits across the training set) because the vast majority of NSP probabilities are between 0.95 and 0.99 or greater than 0.99. Therefore, we experiment with a threshold at 0.95 and between 0.99 and 1. A threshold of 1 means splitting the document at every sentence. For the hierarchical model, we use RoBERT and vary the segmentation scheme between baseline, NSP, and NSP with overlap as described in Sections 3.2.1, 3.2.2, and 3.2.3. We use the Adam optimizer with learning rate of 5e-5 and cross-entropy loss (Kingma and Ba, 2017). We trained each model for 500 epochs (note that we are not fine-tuning BERT).

We also experimented with other pretrained BERT models and didn't notice significantly different or improved results over BERT Base Uncased.

4.2 Dataset

20 newsgroups (20news): The 20 newsgroups dataset contains 18,846 documents split approximately evenly between 20 topics as is shown in Figure 1.[3] The prediction problem is multi-class topic classification. The dataset has a pre-defined train-test split, and within the training set, we further split 90-10 to create the train-validation sets.

4.3 Results and Analysis

Table 1 shows the test accuracies for naive segmentation across a range of window and overlap lengths. In general, an overlap percentage of 75% and a window length of 100 tokens performed the best in the 20news dataset. The performance gains from overlap motivated experimenting with overlap in the NSP segmentation strategy.

Table 2 shows the results for NSP-only and NSP with overlap segmentation schemes. Both valida-

[3] https://huggingface.co/datasets/newsgroup

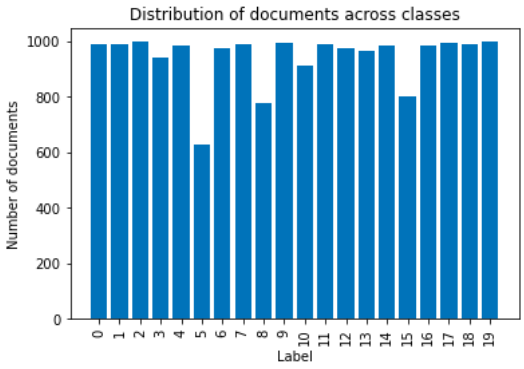

Figure 1: Label distribution of the data in the dataset.

		Overlap			
		0%	50%	75%	88%
Window	100	0.59	0.59	**0.63**	0.60
	150	0.56	0.56	0.59	0.61
	200	0.55	0.57	0.58	0.58
	250	0.53	0.53	0.57	0.55
	300	0.51	0.51	0.52	0.52

Table 1: Test accuracy of all naive overlap configurations. Pappagari et al. (2019) baseline of 200-token window length with shift of 50 tokens is underlined. Best configuration is bolded

	NSP-only		NSP w/ Overlap	
$1-p$	Validation	Test	Validation	Test
0.05	0.70	0.63	0.69	0.62
1e-2	**0.72**	**0.63**	0.70	0.59
1e-3	0.70	0.64	0.70	0.62
1e-4	0.71	0.63	0.70	0.63
1e-5	0.66	0.60	0.70	0.63
0	0.67	0.60	**0.74**	**0.64**

Table 2: Validation and test accuracy figures for NSP-only and NSP with overlap. p is the probability threshold used to segment documents. The best performing results for NSP and NSP with overlap are bolded.

Segmentation	Validation	Test
Baseline	67%	58%
Best Naive Overlap	71%	63%
Best NSP-only	72%	63%
Best NSP w/ Overlap	74%	64%

Table 3: Validation and test accuracies for hierarchical models under a variety of segmentation strategies. For each strategy, the model with the highest validation accuracy is displayed.

tion and test accuracies are robust across a wide range of probability thresholds.

Table 3 shows a summary of the results for each segmentation strategy with the results for the model with highest validation accuracy displayed. We were able to achieve a higher validation and test accuracy over Pappagari et al. (2019) by tuning the naive overlap method with a grid-search across window length and overlap percentage. Furthermore, we were able to achieve superior performance on both the validation and test sets with both the NSP-only and NSP with overlap segmentation schemes.

We can also see that reliance on a naive segmentation strategy makes the accuracy of the model susceptible to how the hyperparameters are tuned, and potentially the training dataset chosen. Table 1 shows test accuracy ranging from 51% to 63% for common window length and overlap thresholds while Table 2 show much less variance–between 60% and 63% for NSP and between 59% and 64% for NSP with overlap.

5 Discussion

We investigated using a smarter segmentation scheme in the Hierarchical Transformers model and found that using BERT's NSP capabilities improves performance. Attempting to combine NSP segmentation with overlap by overlapping each segment by one sentence did not achieve significantly better results over non-overlapping NSP segmentation. In some sense, this is expected because overlapping segments does not actually give the model any extra information – it only gives the model extra chances to extract the same information. Since NSP-only performed equally as well as NSP with overlap, using a smarter segmentation scheme may reduce the need for overlap.

Our results show that the formal linguistic organization that authors have encoded within documents, e.g. stanza breaks within poems, act breaks within plays, or chapter breaks within books, rather than being discarded in the pre-processing stage can instead be leveraged to give both human and machine readers a stronger framework for understanding language. In this paper, we designed our hierarchical model to use the information encoded in sentence breaks–the simplest and most universal form of linguistic organization–and achieve consistent and superior performance as a result.

Collaboration Statement: Alex wrote the first pass of the hierarchical model pipeline. Andrew checked the implementation and extended the NSP segmentation strategy to the NSP with sentence overlap method. Shwetanshu scaled the pipeline on the high performance cluster and extended the experiments to other BERT variants. All members were involved in brainstorming, running experiments, discussing results, and writing the report.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Adji B. Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2017. Topicrnn: A recurrent neural network with long-range semantic dependency.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *CoRR*, abs/1903.00161.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Tibor Kiss and Jan Strunk. 2006. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics*, 32(4):485–525.

R. Pappagari, P. Zelasko, J. Villalba, Y. Carmiel, and N. Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.