The NLP Task Effectiveness of Long-Range Transformers

Guanghui Qin **Yukun Feng** **Benjamin Van Durme**
Department of Computer Science, Johns Hopkins University
{gqin2, yfeng55, vandurme}@jhu.edu

Abstract

Transformer models cannot easily scale to long sequences due to their $O(N^2)$ time and space complexity. This has led to Transformer variants seeking to lower computational complexity, such as Longformer and Performer. While such models have theoretically greater efficiency, their effectiveness on real NLP tasks has not been well studied. We benchmark 7 variants of Transformer models on 5 difficult NLP tasks and 7 datasets. We design experiments to isolate the effect of pretraining and hyperparameter settings, to focus on their capacity for long-range attention. Moreover, we present various methods to investigate attention behaviors to illuminate model details beyond metric scores. We find that the modified attention in long-range transformers has advantages on content selection and query-guided decoding, but they come with previously unrecognized drawbacks such as insufficient attention to distant tokens and accumulated approximation error.

1 Introduction

Transformer-based models (Vaswani et al., 2017) have advanced the state of the art in natural language processing. However, their quadratic time and space complexity hinder their application on long texts. Various proposals have been made to address these concerns (Tay et al., 2020c), with mathematical guarantees on improved time or space. These models have been evaluated primarily via perplexity (Dai et al., 2019) and non-NLP benchmarks (Tay et al., 2020b). These metrics may not be ideal (Sun et al., 2021) and may not reflect performance on complex NLP tasks (Arutiunian et al., 2020; Thorne et al., 2021). We argue these metrics have not been sufficient for the development of efficient Transformers and their practical application on long texts, and that existing benchmarks are insufficient guides for architecture selection.

It is not straightforward to have a fair and side-by-side comparison among those models due to the differences between their pretraining and hyperparameter settings (Tay et al., 2020c), and the metrics alone cannot convey detailed information about the self-attention blocks (Sun et al., 2021). We wish to fairly validate the effectiveness of long-range attention techniques, and to uncover the underlying factors behind model behaviors. We critique the reliance on perplexity evaluations in previous work, experimenting with five difficult, long-text NLP tasks. These tasks cover typical NLP modeling scenarios: token or span-level prediction, sequence-level classification, and sequence-to-sequence generation. To our knowledge, this is the first work to evaluate long-range transformers on such a wide spectrum of representative NLP tasks.

To verify the key features of long-range transformers, we *ablate* distant attention to measure what they gain from long-range mechanisms. For models without pretrained checkpoints, we migrate parameters from their prototype models for fairness. We cover 3 main kinds of long-range transformers, including pattern-, recurrence-, and kernel-based methods. To our knowledge, we are the first to adopt all these methods to probe transformers. To investigate the relationship between performance and document lengths we break down the metric with a customized algorithm (Bagga and Baldwin, 1998). Also, we use entropy and attribution analysis (Li et al., 2017) to test the effectiveness of cached memories in recurrent transformers and the global tokens for query-based problems.

We find that long-range context brings performance gains to transformers in some cases, which we attribute to more selective attention, especially for query-based tasks like QA. Surprisingly we observe that some long-range models do not effectively utilize distant information, and the accumulated error of approximation is unacceptable. We hope this analysis helps practitioners better understand the current state of the art of long-range attentions and suggests paths for future research.

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3774–3790
May 2-6, 2023 ©2023 Association for Computational Linguistics

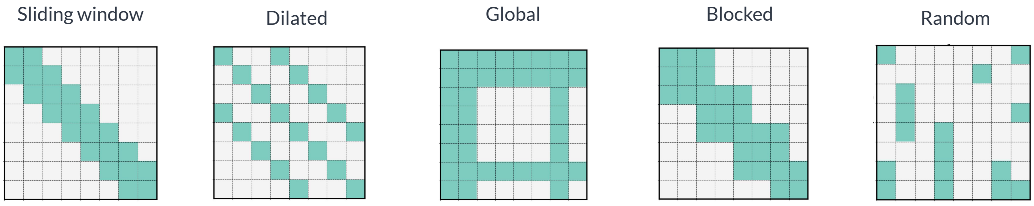

Figure 1: Illustration of 5 patterns used by long-range transformers, from Beltagy et al. (2021) with permission.

2 Background

2.1 Long-Range Transformers

Researchers have proposed a number of Transformer variants (Tay et al., 2020c). Most of these models support decoding (causal masking) (Peng et al., 2021a), while only a few of them have pretrained checkpoints (Beltagy et al., 2020, inter alia). We cluster these approaches into 3 main categories.

Sparsified Patterns Pattern-based methods try to make self-attention sparse. Some apply prespecified attention patterns. Specifically, Longformer (Beltagy et al., 2020) applies 3 patterns: *Sliding window* requires that each token can only attend to the tokens in a local window, *dilated pattern* lets each token only attend at fixed intervals, while the *global pattern* requires a few tokens as globally attended and lets them to attend all tokens in the sequence. In addition to the global pattern, BigBird (Zaheer et al., 2020) applies a *blocked pattern*, which splits the sequence into fixed-length blocks, and *random patterns*, by which tokens can attend to any other tokens randomly. An illustration is shown in fig. 1. Although the attention of each layer is not full, the receptive field can be increased as multiple layers are stacked. The selected or appended "global" tokens can be task-specific (Beltagy et al., 2020), allowing for direct distant information exchange. Instead of pre-defined attention patterns, some use content-based patterns so they become learnable, with techniques including locality sensitive hashing (Kitaev et al., 2020), the differentiable Sinkhorn algorithm (Tay et al., 2020a), or the learnable routing algorithm (Roy et al., 2020).

Recurrence & Compressed Memory These methods use segment-level recurrence to reuse the cached hidden states of previous steps. Transformer-XL (Dai et al., 2019) and XLNet (Yang et al., 2019) connects different chunks with cross-attention, where the tokens in a block attend to the hidden states of the previous blocks in addition to their self-attention. Note that the gradients remain in the same segment and are not propagated to previous segments (fig. 2). To reduce the number of history hidden states, Rae et al. (2020) compress them as memories for efficient re-use with pooling or convolutions. From a different perspective, Izacard and Grave (2021) use retrieval-based methods to collect evidences from external knowledge, resulting in a more targeted context information.

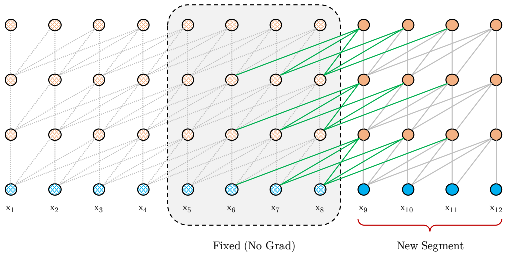

Figure 2: Recurrent transformers. "No Grad" means that the gradients do not back-propagate to this block. Obtained from Dai et al. (2019) with permission.

Low-Rank & Kernels These methods approximate the self-attention with low-rank approximation (Wang et al., 2020) or kernelization without explictly computing the matrix production. Among them, Choromanski et al. (2021); Peng et al. (2021b) use random features. Katharopoulos et al. (2020) reduce the time complexity to linear and space complexity to constant by replacing softmax with linear kernel features.

2.2 Benchmarks and Analysis

There is not an agreed-upon standard benchmark for long-range transformers. Researchers have considered various tasks and domains, including language (Dai et al., 2019), protein sequences (Choromanski et al., 2021), and images (Katharopoulos et al., 2020). Few conduct experiments on long sequence NLP tasks, including question answering (Beltagy et al., 2020) and summarization (Zaheer et al., 2020). Tay et al. (2020b) propose Long Range Arena comprised of six non-NLP tasks to exclude the factor of pretraining. Concurrent to our efforts, Shaham et al. (2022) propose a suite of text-to-text tasks as a long sequence NLP benchmark.

Researchers are also interested in the utility of context in transformers. Rae and Razavi (2020) find that Transformer-XL does not necessarily need long and deep contexts. Sun et al. (2021) reveal that Longformer and Routing transformers can only reduce the perplexity of LMs on a small set of tokens. More related to our work, Lai et al. (2020) show that BERT can make use of a larger scope of context than a BiLSTM.

3 Setup

3.1 Settings

It is non-trivial to compare distinct transformer models, due to differences between their pretraining and hyper-parameter settings. Our goal is to minimize these confounding factors to allow a focus on the long range attention ability of each model on different tasks. We therefore propose two sets of experimental conditions.

Restricted Attention Range To evaluate the performance gain from long-range attention, we evaluate models in both their default *context-aware* settings and a *context-agnostic* setting that restricts the receptive range of the self-attention blocks. For pattern-based transformers, we achieve the restriction by segmenting the input sequence into chunks and running the transformers on segments independently. For recurrent models, we ablate the recurrence to eliminate the dependencies between segments. In practice, we segment the texts into chunks with length L,[1] which ranges from 128 to 1536. L=∞ indicates no segmentation is used.

Parameter Migration Kernel-based models usually do not come with checkpoints for general tasks, but they may have similar structures to other pretrained models like BERT and may be designed to approximate the original results. Therefore, it is feasible to migrate parameters from pretrained prototypes to their "efficient version" to observe if the performance could be preserved. This type of method can be suitable for models without additional parameters, such as Performer.

3.2 Transformers and Tasks

Transformers We consider three approaches: 1) *pattern-based*: Longformer (Beltagy et al., 2020), and BigBird (Zaheer et al., 2020); 2) *recurrent*: XLNet (Yang et al., 2019); and 3) *kernel-based*: Performer (Choromanski et al., 2021). We also include the results of RoBERTa (Liu et al., 2019) and SpanBERT (Joshi et al., 2020), where some of our approaches are initialized from those two non-long-range models. Due to memory requirements, we use the base version for all models.[2] You may refer to appendix B for more details.

Dataset	Task	#tokens	#docs
Ontonotes	Coref.	467	3493
TriviaQA	eQA	2895	95k
DocNLI	NLI	399	1.44m
SummFD	Summ.	5.6k	4.3k
GovRep	Summ.	7.9k	19k
Qasper	aQA	3.7k	5.7k
QuALITY	aQA	4.2k	6.7k

Table 1: Task and dataset overview. The #tokens is the number of tokens per doc on average.

Tasks We cover five tasks of three types, including 1) *span-level* predictions: coreference resolution (Coref.) (Weischedel et al., 2011) and extractive question answering (eQA) (Joshi et al., 2017); 2) *sequence classification*: natural language inference (NLI) (Yin et al., 2021);[3] and 3) *seq2seq*: summarization (Summ.) (Chen et al., 2021; Huang et al., 2021), abstractive QA (aQA) (Dasigi et al., 2021; Pang et al., 2022). We pick seven datasets that involve long texts, whose statistics are shown in table 1. For more details about the data preprocessing, please refer to appendix A.[4]

4 Experiments

4.1 Coreference Resolution

Coreference resolution (coref.) is the task of identifying mention spans and clustering them into entities. We consider multiple coreference strategies: 1) the widely used Coarse2Fine (C2F) method[5] (Lee et al., 2018) which relies on span representations and 2) the current state-of-the-art method called Start2End (S2E) (Kirstain et al., 2021) that works on token representations. The

[1] We use wordpieces instead of words in this paper.

[2] We used the codebase of Katharopoulos et al. (2020) for Performer and Huggingface (Wolf et al., 2020) for the rest.

[3] DocNLI is modified from ANLI (Nie et al., 2020), SQuAD (Rajpurkar et al., 2016), DUC2001, DailyMail (Nallapati et al., 2016), and Curation (Curation, 2020)

[4] Our codebase is available on `https://github.com/hiaoxui/long-range-transformers`.

[5] We used the re-implementation by Gardner et al. (2018).

	Encoder	L=128	L=256	L=512
Model: Coarse2Fine	Longformer	75.74	76.72	77.36
	Longformer^G	75.68	76.25	77.23
	BigBird	75.95	76.78	77.64
	XLNet	74.57	74.48	74.33
	XLNet^m	74.73	75.76	76.29
	RoBERTa	74.64	76.45	76.83
	RoBERTa^p	51.58	51.71	50.39
	SpanBERT	75.04	75.84	76.59
	SpanBERT^p	52.46	52.06	50.51
	Longformer	76.77 (1024)	76.32 (∞)	
	BigBird	77.31 (1024)	77.57 (∞)	
Model: Start2End	Longformer	74.77	76.27	77.73
	Longformer^G	74.15	76.19	77.41
	BigBird	73.68	75.57	77.40
	XLNet	45.89	60.05	68.23
	XLNet^m	52.61	56.37	66.91
	RoBERTa	71.96	76.27	77.78
	RoBERTa^p	40.06	42.35	41.69
	SpanBERT	68.70	74.27	75.32
	SpanBERT^p	38.69	41.93	42.10
	Longformer	77.54 (1024)	77.57 (∞)	
	BigBird	77.43 (1024)	77.66 (∞)	

Table 2: Coref. results with the C2F and the S2E models. Numbers are averaged F1 (MUC, B^3, and CEAF$_{\phi_4}$). Longformer with ^{G} uses global tokens. XLNet with ^{m} uses recurrence memory. Encoders with ^{p} have their self-attention replaced with a Performer kernel.

dataset we use is **Ontonotes 5.0**. Transformers considered include **Longformer**, **XLNet**, and **Performer**. We migrate the parameters of **SpanBERT** and **RoBERTa** to Performer and include results on these models as well. We segment the input tokens into chunks with lengths of L . For models with global tokens, we lack a natural choice so we consider all tokens to be global. For XLNet, we keep the memory of the same length of the segments (e.g. we keep a memory length of 256 for a model with segment length 256).[6] The results are shown in table 2. Refer to appendix F.1 for complete results.

Some observations are consistent across two coref. models. 1) Though further pretrained upon RoBERTa, pattern-based methods do not show improvement over RoBERTa, even with longer attention range. 2) Models gain advantage when the segments get longer, but it is saturated when the segment length reaches 512. Distant contexts might not be exploited. 3) Performer-based models under-perform their corresponding non-kernelized models by a huge gap. 4) XLNet performs better with cached memory, but the performance gain is less observable when for shorter segments. [7]

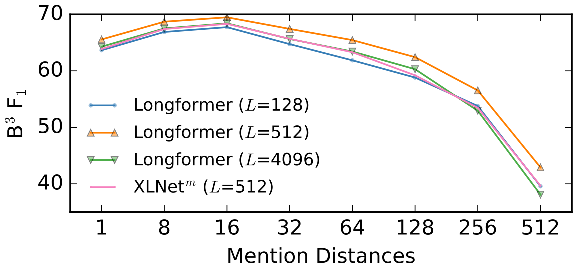

Figure 3: B^3 breakdown scores of 4 models for mention pairs with ranges from $[1, 8)$ to $[512, \infty)$.

To further show the performance of those models on documents with different lengths, we conduct metric breakdown with a few typical configurations. Instead of simply clustering the document according to the lengths, we propose a breakdown-version B^3 metric. Given a mention distance range $[L_1, L_2]$, we calculate its corresponding B^3 value by only considering the mention pairs whose distances fall into this range. The breakdown metrics are shown in fig. 3. We can see that the performance of all models follows the same trend and peaks at the $[16, 32)$ bucket. Also, the graph shows that Longformer with longer context encoding does NOT benefit on distant mentions (p value < 0.01). [8] On the contrary, they suffer more from distance than shorter-context models, showing that long-range attention fails to capture long-distance information.

4.2 Natural Language Inference

NLI is a classification task concerning a premise and hypothesis with variable lengths. The **DocNLI** dataset uses document-length inputs with binary labels (`entailment` and `not entailment`). We adopt the model proposed by Yin et al. (2021), and consider **Longformer**, **BigBird**, **Performer**, and **XLNet**. We make the prediction on the `CLS` token, which is at the beginning for Longformer and the end for XLNet. Since we are only interested in the encoding of `CLS`, we adopt the strategy of Yin et al. (2021) which truncates the sequence to L while preserving the hypothesis for Longformer and RoBERTa. The results are shown in table 3.

[6] We adopt the same strategies with segmentation and memories in the remainder of the paper.

[7] The observations on XLNet and Performer are consistent across all the tasks in this paper.

[8] We conduct significance test for the comparison between curves. Please refer to appendix D for more details.

Encoder	L=128	L=256	L=512
XLNet	29.95	40.39	24.31
XLNet^m	32.94	45.97	30.42
RoBERTa	48.96	47.78	46.04
RoBERTa^p	17.83	24.91	23.65
Longformer	29.11	25.73	45.28
BigBird	28.95	24.71	31.72
Longformer	45.96 (1024)	44.42 (∞)	
BigBird	33.58 (1024)	18.08 (∞)	

Table 3: F1 scores on the test set for DocNLI.

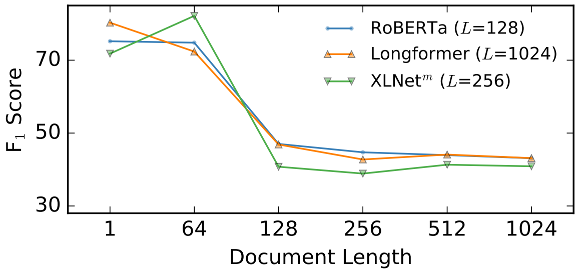

Figure 4: The breakdown analysis of DocNLI. We pick the best configuration for each model for brevity.

Observing table 3, surprisingly, the best performance is achieved with short segments, though the optimal lengths vary from model to model. Also, the performance of Longformer and BigBird is much lower than their baseline RoBERTa (c.f. Yin et al. (2021)). Observing the breakdown analysis in fig. 4, the performance of 3 best models follow the same trend w.r.t. the document length.[9] We speculate that all models are unable to comprehend the relationship between long documents, and long-range attention does not bring any advantage.

Encoder	L=128	L=256	L=512
Longformer	54.26	58.83	63.88
RoBERTa	55.81	60.29	63.45
RoBERTa^p	23.17	21.87	21.11
BigBird	55.28	59.39	63.51
XLNet	51.46	56.26	60.05
XLNet^m	52.71	57.96	62.85
Longformer	63.91 (1024)	63.66 (∞)	
Longformer^G	- (1024)	72.96 (∞)	
BigBird	66.50 (1024)	71.78 (∞)	

Table 4: Results for TriviaQA. Longformer^G indicates that the Longformer sets question as the global tokens.

	Chunk	512	1024	1536	∞
Qasper	BART	24.70	26.30	-	-
	LED	8.40	15.80	17.86	18.79
	LED^G	-	-	-	28.64
QLTY	BART	26.80	26.00	-	-
	LED	30.73	31.35	31.78	31.21
	LED^G	-	-	-	29.87

Table 5: Performance Qasper and QuALITY. Qasper is evaluated with F1 and QuALITY with accuracy. The results on BART are from Shaham et al. (2022).

4.3 Question Answering

For question answering we experiment with eQA (**TriviaQA**) and aQA (**Qasper** and **QuALITY**). **Performer**, **Longformer**, **BigBird**, and **XLNet** are tested for encoder-only tasks; **BART** and Longformer-Encoder-Decoder (**LED**) is used for seq2seq tasks. For Longformer^G and BigBird, we set the question text (and candidate answers for QuALITY) as global tokens.[10] **TriviaQA** is a question answering dataset that involves extracting answer spans from reference documents. We adopt the method and codebase of Joshi et al. (2017). F1 is used as evaluation metrics. **Qasper** addresses the QA task in the domain of academic papers and involves various answer types: extractive, abstractive, boolean, and unanswerable. We unify such tasks as an abstractive QA task (cf. Shaham et al. (2022)) and implement an LED-based decoder to generate answers. F1 score is used for the evaluation of Qasper. **QuALITY** is a multiple-choice QA task. Given a question and a passage, the task is to select the correct answer from several candidates. We regard it as a seq2seq problem, with the objective to predict the correct answer conditioned on the concatenation of query, candidate answers, and passage. During inference, the answer with the least perplexity is selected. All results are shown in tables 4 and 5, and the full results with more metrics are shown in appendix F.2.

Across all results, we find that larger receptive fields lead to better performance in most cases. More importantly, setting queries as global tokens greatly benefits the performance on both eQA and aQA. For QuALITY, it slightly hurt the perfor-

[9] The graph looks less smooth than fig. 3, possibly because DocNLI is made up of examples pulled from different datasets, which may have examples of different average lengths (cf. tab 1 in Yin et al. (2021)). Therefore the length of an example in DocNLI may correlate with different domains making up the dataset, which would interfere with our analysis. Future work will consider other datasets without this confounding concern.

[10] The global tokens of BigBird are fixed in the first 2 blocks, so we place the query at the beginning of the sequence.

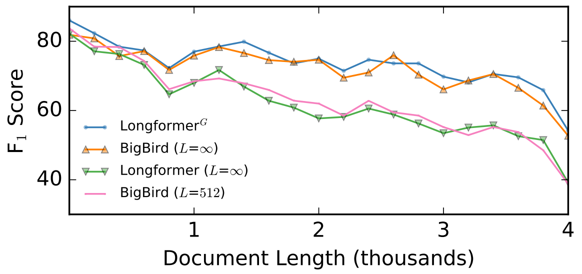

Figure 5: The performance of Longformer and BigBird on different lengths of TriviaQA documents. Note that Longformer^G and BigBird (L=∞) have global tokens.

	Chunk	512	1024	1536	∞
SF	BART	26.30	27.20	-	-
	LED	32.81	33.07	33.22	33.57
GR	BART	45.60	47.90	-	-
	LED	53.86	54.13	54.83	56.60

Table 6: Results for Summarization on SummFD and GovRep. ROUGE unigram is used as the metric. Results on BART are from Shaham et al. (2022).

mance to set both query and candidate as global tokens. We reckon too many global tokens might introduce more noise, similar to the case of coref.

We speculate that the performance gain mostly comes from enhanced attention to the query, which if further verified by the metric breakdown that is shown in fig. 5. All models perform well on short texts, while models with global tokens obtain an observably greater advantage over the baseline models for longer documents (p value < 0.01). We think that the global token mechanism could help the model be less distracted on long texts via more attention on the queries (section 5.3), which consequently improves the performance.

4.4 Summarization

As a typical seq2seq problem, we adopt **LED** to perform the summarization task. We chunk the source sequence into segments (no segmentation for $L = \infty$) to restrict the receptive range. Intuitively, the summary may be benefited from the contextual representation with a broader view of the document. Two datasets are used: **SumScreen** addresses the domain of TV shows. Following Shaham et al. (2022), we use the subset of ForeverDreaming (SummFD) consisting of 88 different shows. The goal is to summarize the transcript of an episode, for which the recap is used as the ground truth summary. **GovReport** is a long-document summarization dataset in the domain of government policies with human-written summaries. ROUGE (Lin, 2004) is used as the evaluation metric. Results are shown in Table 6, and the full results are shown in appendix F.3.

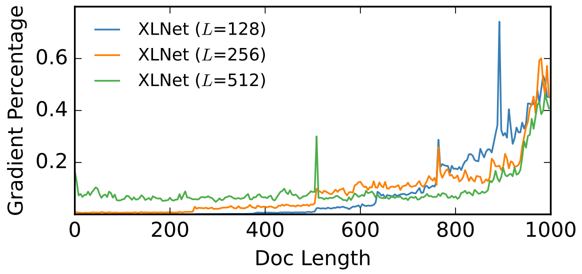

Figure 6: The gradient distribution over tokens for XLNet^m on DocNLI documents.

From table 6, we observe slight superiority of the context-aware models. Note that cross attention will attend to all source tokens whether we segment it or not, but the intuition of summarization is to *skim over* the document, so we speculate that the performance improvement may be related to the *selectivity* of the encoder-side attention, which is further analyzed in section 5.2. We do not conduct breakdown analysis for summarization because the sequence length directly contributes to the metric.

5 Analysis

5.1 The Attribution of Recurrence Memories

Even if we know that distant contexts can help or hurt performance, it's still unclear *how much* they contribute to the predictions. One way to quantify this is *attribution analysis* (Simonyan et al., 2014; Li et al., 2017). Suppose ℓ is our loss function and $\mathbf{e}_i \in \mathbb{R}^d$ is the word embedding of the i-th token. We use α_i to measure the attribution of the i-th token to the final prediction where

$$\alpha_i = \left\| \frac{\partial \ell}{\partial \mathbf{e}_i} \right\|_l . \tag{1}$$

We set $l = 1$ to take the L_1 norm in practice, and the ground truth labels in the test set are used to calculate the gradient. Intuitively, tokens with higher contribution have greater gradient norms. Also, recurrent models like XLNet stop the gradient from being propagated back to the cached memory, and we temporarily turn off this feature for analysis.

We apply this method to XLNet on DocNLI, where we pick 128 documents of lengths between 1,000 and 1,024. We normalize the α_i over all tokens for each document, and take an average on

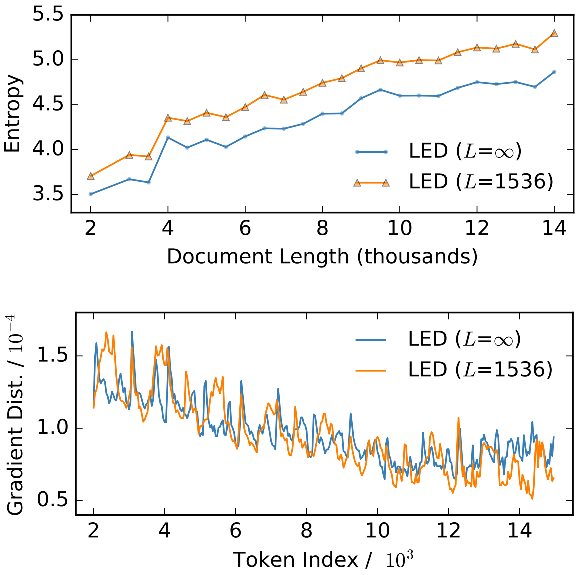

Figure 7: Attention distribution entropy (above) for documents with different lengths and attribution analysis (below) over source tokens for LED on SummFD.

each token index across all documents. The results are shown in fig. 6. Note that prediction is made in the last segment, and previous segments contribute only through the memory. We see the attribution of tokens is stratified according to their segment lengths, with a minor peak at the `CLS` token of each segment. As we have more and more segments, the attribution of distant tokens to the final prediction becomes negligible. For example, in the case of L=128, last segment made 53.4% of attribution, while the first segment made less than 0.01%.

5.2 Content Selection in Cross Attention

The cross attention of LED attends to the whole document no matter if we segment the inputs or not, thus we suppose source tokens should have similar attribution to decoding, which can be verified by *attribution analysis* in fig. 7. However, the crucial problem for summarization is whether the attention is *selective*, given that only a portion of the document is helpful. Therefore, we inspect the *entropy* [11] of the cross attention distribution over the source tokens fig. 7.

Reading the entropy curve, we find that the entropy of models without segmentation (L=∞) is consistently lower (p value < 0.01), which can be translated to higher selectivity of cross attention and explains the superiority of LED (L=∞) in table 6. Reading the gradient curve, we find that both models have a relatively uniform attribution over tokens with a slight slope on the left side due to the existence of short sequences. This is reasonable because summarization requires the decoder to skim over the whole document, thus cross attention should not have a locality preference.

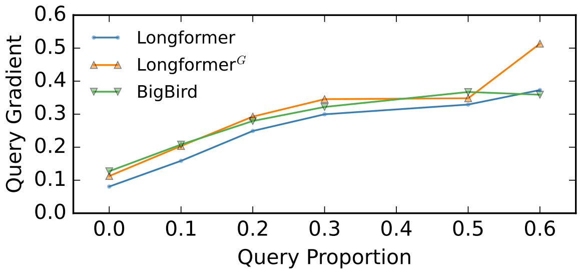

Figure 8: The accumulated gradient on query vs. the proportion of query, i.e. (query length / doc length) with Longformer and BigBird (L=∞) on TriviaQA.

5.3 Query-Guided Extraction and Decoding

Different from other problems, the queries in QA can be treated as a "guidance" on how to read the documents. To see if the encoders can exploit this structure, we conduct attribution analysis on TriviaQA with Longformer and BigBird. The results are shown on fig. 8, where we plot the relationship between the proportion of accumulated gradients on queries and the proportion of query tokens in the document. It is clear that models with global tokens pay more attention to queries (p value < 0.01 except for one exception), which is consistent with the purpose of their design.

For scenarios where seq2seq decoding meets queries, intuition suggests queries could instruct the cross-attention to attend to specific tokens, making the decoding more *selective*. From fig. 9, the entropy of attention distribution on source sequence against the doc length, we see Longformer with global token unquestionably has lower entropy (p value < 0.01), implying more targeted decoding.

5.4 Error Accumulation of Kernel Methods

We find that Performer could not match the results of its prototypes (tables 2 to 4). We suspect that the error incurred by the kernel approximation may not be acceptable for span-level tasks like coref. We conducted another set of experiments: Instead of training the performer model from the checkpoints of SpanBERT, we directly replace some layers from a fine-tuned SpanBERT model with Performer layers with their parameters preserved. Experiments

[11] The distribution entropy is averaged over decoding tokens, attention heads, and transformer layers.

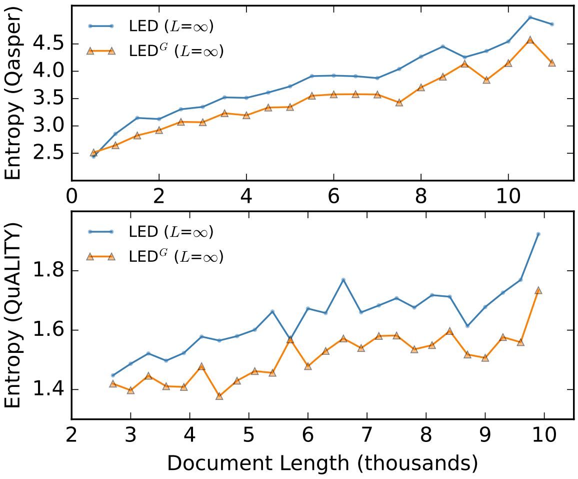

Figure 9: Entropy of source-side attention distribution vs. document length on Qasper and QuALITY.

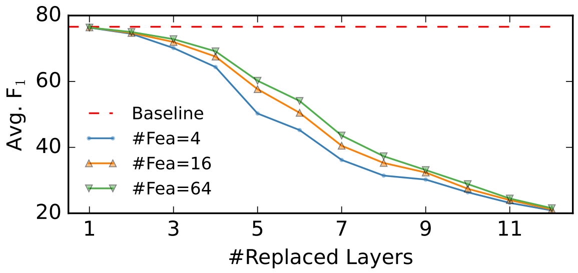

Figure 10: Results of SpanBERT (L=512) with layers replaced with Performer layers with "#Fea" features. The baseline is the performance of the original model.

are conducted with C2F on Ontonotes 5.0.

In fig. 10, we try to replace P layers of SpanBERT in a top-down manner, where $P = 1, 2, \ldots, 12$. We also try using different feature dimensions to exclude the possibility of insufficient features. We find that although large dimension of random features brings marginal advantages to the performance, Performer is not very sensitive to this factor. Instead, the performance drops dramatically as we replace more layers, from the baseline F1 78 to ~20 with all layers replaced. Based on our findings, we conclude that Performer is a good approximation for shallow transformers, even with very low-dimensional random features. However, as we stack more transformer layers, the accumulated errors can be unacceptable, which leads to a failure in the performance.

6 Experiment Confounders

Pretraining and Adaptation The purpose of this paper is to evaluate the effectiveness of different long-range attention approaches by reducing the confounder of pre-training, but it is still unclear how the results would change if we pre-train those models with the same configuration from scratch. Unfortunately, it requires much more resources and introduces more confounders of training settings, and we adopt the most straightforward way to ablate the long-range attention or migrate parameters. For each model-task pair, we make the most natural choice, e.g. setting queries as global tokens, trying to minimize human biases of model adaptations.

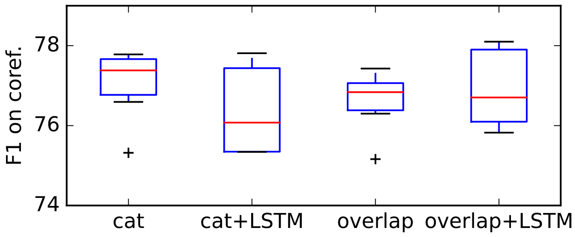

Figure 11: Results of different pooling strategies.

Pooling Strategies Concatenating the representation of segments, while being natural and commonly used, is not the only pooling strategy. As a comparison, we incorporate the results of other 3 solutions: 1) Split the documents into segments of L tokens with $L/2$ overlapped between adjacent segments (Joshi et al., 2019); 2) Stack an LSTM layer over the Transformer representation of the segment ; 3) A combination of above methods. We conduct experiments on the coreference resolution task with both C2F and S2E solution and many variants of transformers as our encoder. We leave the experiment details and discussions in appendix E and show a brief result in fig. 11. We have similar observation with Joshi et al. (2019) that overlapping does not bring performance improvement. Moreover, though introducing more parameters, a stacked LSTM even hurts the performance. We conclude that pooling strategies do not affect our analysis in section 5.

7 Discussion

Researchers have proposed many innovative methods for efficient self-attention over long sequences. The key ideas work as desired in certain cases, though we demonstrate several drawbacks.

Surprisingly, **pattern-based methods**, as the most popular approach, are not necessarily benefited from long-range attention in the general case. Larger sliding windows are helpful, but the benefit would quickly saturate or become negative (table 2).

However, when a small portion of *guidance text* (e.g. query in QA) exists, setting it as *global tokens* can make it more attended and significantly boost the performance (section 5.3). When such text doesn't exist, setting all tokens as global would hurt the performance (table 2). Moreover, we find that long-range attention and global tokens are correlated with the *selectivity* of *seq2seq* problems, which consequently helps the decoding.

The memory of **recurrence models** generally improves performance, proving historical hidden states are beneficial for transformers in various tasks. However, XLNet does not fully exploit the history tokens, with distant information less attended (section 5.1). We speculate that it is because XLNet is pretrained to predict masked tokens, which does not frequently need the participation of long-range context (Sun et al., 2021). Also, the stop-gradient trick may hinder the model from effectively attending to memories.

The approximation of **kernel-based methods** works very well for shallow networks, but faces serious error accumulation problems when transformer layers are deeply stacked, which cannot be remedied by having high-dimensional random features (section 5.4). The resulting performance drop is not acceptable even for the "base" version of transformer encoders with 12 layers (table 2).

8 Conclusion

We conduct experiments with various long-range transformers on NLP tasks that involve long sequences, trying to fairly evaluate their long-range attention ability. While some methods are validated on certain tasks, we also find some previously unrecognized drawbacks. We further analyze the attention behaviors of these transformers with metric breakdown, attribution analysis, and entropy analysis, revealing the performance of those models might be correlated with the attribution of distant tokens, selectivity of attentions, or the approximation errors. We hope our work would shed light on the future development of long-range transformers.

Model Selection Takeaways Based on our findings, we have the following suggestions. For common tasks, such as sequence classification or token-level prediction, it is still competitive to chunk the inputs and apply short-range transformers. When explicit guiding text, such as queries, exists, pattern-based models with global token mechanism is preferred. For seq2seq problems, long-range transformers with pretrained checkpoints deliver better performance.

Acknowledgement

We appreciate the proofreading done by Patrick Xia, Marc Marone, Nils Holzenberger, Elias Stengel-Eskin, Yunmo Chen, and Zhengping Jiang . Thanks to the anonymous reviewers for their valuable feedback.

This work was supported in part by IARPA BETTER (#2019-19051600005). The views and conclusions contained in this work are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, or endorsements of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes.

9 Limitations

Energy Cost Our experiments involve a massive amount of training with many transformers on various tasks. Although we don't conduct any pretraining, the energy cost is still non-negligible. However, our hope is our findings here would enable others to more efficiently select a particular architecture for their task, rather than reproducing the work done here.

Experimental Bias Due to the lack of pretrained checkpoints for general purposes, we focus on representative instead of each type of transformer variant. It is possible that these observations are particular to specific artifacts and implementations considered here. Our goal is foremost to provide a roadmap for continued study on questions raised in this article, with new architectures being evaluated in the future by model developers themselves. For similar reasons, existing artifacts are biased towards English, as are many of the datasets employed in this study. We do not believe our findings are specific to English, but it remains for future work in long range transformer evaluation to extend our analysis into multilingual conditions.

Language Bias For similar reasons, existing artifacts are biased towards English, as are many of the datasets employed in this study. We do not believe our findings are specific to English, but it remains for future work in long-range transformer evaluation to extend our analysis into multilingual conditions.

References

Artashes Arutiunian, Morgan McGuire, Hallvar Gisnås, Sheik Mohamed Imran, Dean Pleban, Priyank Negi, and David Arnoldo Ortiz Lozano. 2020. Reproducibility Challenge: Reformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, page 10.

Amit Bagga and Breck Baldwin. 1998. Algorithms for Scoring Coreference Chains.

Iz Beltagy, Arman Cohan, Hanna Hajishirzi, Sewon Min, and Matthew Peter. 2021. Beyond Paragraphs: NLP for Long Sequences.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer.

Mingda Chen, Zewei Chu, Sam Wiseman, and Kevin Gimpel. 2021. SummScreen: A Dataset for Abstractive Screenplay Summarization.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking Attention with Performers. In *International Conference on Learning Representations (ICLR)*.

Curation. 2020. Curation Corpus Base.

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. In *Association for Computational Linguistics (ACL)*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. In *North American Association for Computational Linguistics (NAACL)*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Association for Computational Linguistics (ACL)*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient Attentions for Long Document Summarization. In *North American Association for Computational Linguistics (NAACL)*, pages 1419–1436. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *European Association for Computational Linguistics (EACL)*.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics (TACL)*, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Association for Computational Linguistics (ACL)*.

Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. BERT for Coreference Resolution: Baselines and Analysis. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *International Conference on Machine Learning (ICML)*.

Yuval Kirstain, Ori Ram, and Omer Levy. 2021. Coreference Resolution without Span Representations. In *Association for Computational Linguistics (ACL)*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations (ICLR)*.

Yi-An Lai, Garima Lalwani, and Yi Zhang. 2020. Context Analysis for Pre-trained Masked Language Models. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3789–3804. Association for Computational Linguistics.

Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. Higher-order Coreference Resolution with Coarse-to-fine Inference. In *North American Association for Computational Linguistics (NAACL)*.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2017. Visualizing and Understanding Neural Models in NLP. In *Association for Computational Linguistics (ACL)*.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, page 8.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. S. Zettlemoyer, and V. Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Xiaoqiang Luo. 2005. On Coreference Resolution Performance Metrics. In *Empirical Methods in Natural Language Processing (EMNLP)*, page 8.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond. In *Computational Natural Language Learning (CoNLL)*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A New Benchmark for Natural Language Understanding. In *ACl*.

Richard Yuanzhe Pang, A. Parrish, Nitish Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He, and S. R. Bowman. 2022. QuALITY: Question Answering with Long Input Texts, Yes! In *North American Association for Computational Linguistics (NAACL)*.

Hao Peng, Jungo Kasai, Nikolaos Pappas, Dani Yogatama, Zhaofeng Wu, Lingpeng Kong, Roy Schwartz, and Noah A. Smith. 2021a. ABC: Attention with Bounded-memory Control.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021b. Random Feature Attention. In *International Conference on Learning Representations (ICLR)*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2020. Compressive Transformers for Long-Range Sequence Modelling. In *International Conference on Learning Representations (ICLR)*.

Jack W. Rae and Ali Razavi. 2020. Do Transformers Need Deep Long-Range Memory. In *Association for Computational Linguistics (ACL)*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics (TACL)*.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison Over Long Language Sequences.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do Long-Range Language Models Actually Use Long-Range Context? In *Empirical Methods in Natural Language Processing (EMNLP)*, page 16.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020a. Sparse Sinkhorn Attention. In *International Conference on Machine Learning (ICML)*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020b. Long Range Arena: A Benchmark for Efficient Transformers.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020c. Efficient Transformers: A Survey.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. Database Reasoning Over Text. In *Association for Computational Linguistics (ACL)*.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Conference on Message Understanding*, page 45. Association for Computational Linguistics.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity.

R. Weischedel, E. Hovy, M. Marcus, M. Palmer, R. Belvin, S. Pradhan, L. Ramshaw, and N. Xue. 2011. OntoNotes: A large training corpus for enhanced processing. In *Handbook of Natural Language Processing and Machine Translation. Springer*. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 38–45. Association for Computational Linguistics.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wenpeng Yin, Dragomir Radev, and Caiming Xiong. 2021. DocNLI: A Large-scale Dataset for Document-level Natural Language Inference. In *Association for Computational Linguistics (ACL)*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

A Data Preprocessing

The coref task usually consumes a large amount of GPU memory. For the experiments on Ontonotes, we truncate the sequence longer than 2000 tokens (for c2f model) or 1400 tokens (for s2e model) for the memory concern. The truncation was applied only to the training set.

We didn't do any pre-processing steps for DocNLI (Yin et al., 2021), except for the truncation that we discussed in section 4. Notably, DocNLI is an aggregated dataset constructed from ANLI (Nie et al., 2020), SQuAD (Rajpurkar et al., 2016), DUC2001, CNN/DailyMail (Nallapati et al., 2016), and Curation (Curation, 2020). Documents from different sources may be distinguishable from their lengths (refer to tab 1 in Yin et al. (2021).

For TriviaQA, we use the scripts of Longformer (Beltagy et al., 2020) to pre-process the data. There is no further modifications on the data.

We adopt the dataset of QuALITY and GovReport from Shaham et al. (2022), which picked long sequences from those datasets.

For Qasper and SummScreen, we simply adopt the original dataset and scripts for preprocessing.

B Implementation Details of Transformers

In this section, we discuss the details of modification we did to the transformers.

Recurrence-Based Methods We tested XLNet for recurrence-based methods. We adopt the codebase of Huggingface as the base model, and followed the common strategy to stop gradient from being propagated back into the cached memory. For each segment of the recurrence, we appended two special tokens `SEP` and `CLS` to the sequence, making it like an ordinary input sequence to the XLNet model except for the possible existence of memory states. We concatenated the token representations after recurrences and remove the special tokens from all but the last segment. Empirically, we found that having special tokens can significantly boost the performance.

Pattern-Based Methods For the pattern-based methods, when we chunked the input sequence into segments, we appended the `SEP` as we did for the XLNet, and prepended the `CLS` token as a convention of other transformers. After concatenation, we removed all the special tokens except for the first `CLS` the last `SEP`, which made it structurally similar to the outputs of non-segmented transformers. What's more, for sliding window mechanism, we might reduce the window size to segment length if needed to save compute and memory.

Kernel-Based Methods We tested Performer (Choromanski et al., 2021) as a representative of kernel-based methods. Given that training from random initialization would lead to suboptimal results, we migrate the parameters from base models (e.g. BERT, RoBERTa) to kernel methods. In detail, we replace the self-attention layers of the base models with kernels. Given that Performer does not require any additional parameters, except for the orthogonal random feature vectors in the FAVOR+ mechanism. Following the default implementation of the fast transformer codebase[12], we set the feature dimension as the query dimension by default for main experiments, though we found the performance isn't sensitive to those features in section 5.4.

C Experiments Details

In this section, we introduce the details of our experiments in the main paper, including hyperparameters, training strategies, data split and loading, and other configurations that are necessary to replicate our results.

Computational Resources All of our experiments were done with NVIDIA RTX 6000 GPU with 24GB of memory. We did most of the experiments with single cards, except for TriviaQA, for which we used multi-GPU training with 4 cards.

Coreference Resolution We used two models: coarse2fine (C2F) (Lee et al., 2018) and start2end (S2E) (Kirstain et al., 2021). For C2F model, we used the codebase reimplemented by AllenNLP (Gardner et al., 2018) for its flexibility on encoder exchange. We used the official codebase of S2E model for other experiments. We didn't change any hypermeters except for the difference on the encoders. We adopted the same training strategy of these repos without any modifications, i.e. we train the model with certain epochs (40 for C2F and 129 for S2E) or until convergence and pick the model performed best on the dev set for evaluation. The typical training time of coref models was 10h for C2F and 24h for S2E.

[12] `https://github.com/idiap/fast-transformers`

Natural Language Inference Due to the size of DocNLI dataset (942k training and 234k dev examples), it's infeasible to adopt the common training strategies. Instead, we train the model with mini-batch gradient descent with a batch size of 4 for only one epoch. Because the training and dev set are too large to fit into CPU memory, we split the training set into smaller datasets consisting of 32768 examples, and iterate over training and dev set during training. We pick the model with best performance on the dev set (not the whole set but one iteration) for test. We use the whole test set consisting of 267k examples for the final evaluation. We adopt the same architecture as the model used in Yin et al. (2021) and reimplement it with AllenNLP Gardner et al. (2018). The typical training time is around 2 days and the test time is around 4 hours.

Question Answering For TriviaQA, we adopted the training scripts and hyperparameters used by Beltagy et al. (2020) except for that we set the training batch size as 4 and number of epochs as 8. The performance is evaluated on the dev set and we pick the best checkpoint with patience of 3.

For Qasper, we follow the training scripts and hyper-parameters in its official repository. [13] We disable the evidence setting, and extend the training to a maximum epoch of 20. The performance is evaluated on the dev set with patience of 5.

For QuALITY, we adopt the LED model with "allenai/led-base-16384" configuration from Huggingface. [14] We concatenate the question, candidate answers, and passages as the encoder input, and the correct answer as the decoder input for training. During inference, we feed each candidate answer as the decoder input, and consider the one with the lowest perplexity as the predicted answer. We use a warmup steps of 1000 and learning rate of 5×10^{-5} for training. Evaluation is performed on the dev set with a patience of 5.

Summarization For both the SummScreen and GovReport datasets, we use the LED model with "allenai/led-base-16384" configuration. We use a warmup steps of 1000 and learning rate of 5×10^{-5} for training and patience of 5 for testing. GovReport is evaluated on the dev set and SummScreen is evaluated on its official test set.

D Significance Test

For the comparison between curves in the section 5, we conduct significance test using bootstrapping methods to verify our conclusions. Let D be the test set for a task. For the performance comparison between two configurations, we sample a new D^* from D with replacement and we keep $|D^*| = |D|$. We treat the event "configuration A performs better than B" as a Bernoulli random variable P, and compute the probability of the null hypothesis $P < 0.5$ as the p value. We sample $B = 1024$ test sets for each comparison. If multiple significance tests are conducted, we only report the larges value that we obtain. [15]

For example, in fig. 3, we claim that the performance of Longformer (L=512) is better than any other encoders regardless of the mention distances. To verify it, we conduct significance test between Longformer (L=512) and other 3 encoders for every mention distance. The greatest p value among 24 p tests is smaller than 0.01, so our claim is secured by our significance test.

E More Pooling Strategies

We conduct experiments with 4 transformers, including 2 short-range transformers (RoBERTa and SpanBERT) and 2 long-range transformers (Longformer and BigBird) on the coreference resolution task. We set L=512, which is the maximum acceptable length for short-range transformers.

The full results are shown in table 7, and a box plot can be found in fig. 11. In overall, we have similar observations as Joshi et al. (2019) that overlapped segments do not offer improvements on the performance. Similar findings can be found for LSTM settings and the combination of them. More importantly, the performance difference of table 7 is consistently with tables 8 and 11 except for a few outliers. Thus, we conclude that direct concatenation is already enough to exploit the pretrained transformers, and changing pooling strategies do not greatly interfere our analysis.

F Full Experiment Results

In this section, we list the full results of all the experiments in the main paper.

[13] https://github.com/allenai/qasper-led-baseline

[14] https://huggingface.co/allenai/led-base-16384

[15] For the curves in fig. 8, we exclude one exception case at $x = 0.6$. For the curves in fig. 9, we exclude one exception case at $x = 500$.

F.1 Coreference Resolution

The full results of coreference resolution are shown in tables 8 and 11. We use MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and CEAF_{ϕ_4} (Luo, 2005) as the evaluation metrics. Following the convention, we use the "Avg." as the main metric, which is an average among the F1 score of 3 metrics. All the results are reported on the test set.

F.2 Question Answering

The full experiment results on TriviaQA is shown in table 9. We use both F1 and exact match (EM) as the metrics. A few cells are left blank because of the constraints of the transformers.

F.3 Summarization

The full results on summarization is shown in table 10. We used ROUGE (Lin, 2004) as the metric. R1, R2, and R3 stands for ROUGE unigram, bigram, and longest common subsequence. Note that 1536 is the windows size of the LED model, and 1024 is the maximum length supported by the BART model.

	Encoder	MUC			B^3			CEAF_{ϕ_4}			Avg.
		P	R	F1	P	R	F1	P	R	F1	
Model: Coarse2Fine	RoBERTa	81.6	85.0	83.3	72.1	78.2	75.1	72.1	72.2	72.2	76.8
	$\text{RoBERTa}_{\text{overlap}}$	82.9	84.5	83.7	73.1	77.1	75.0	73.5	71.7	72.5	77.1
	$\text{RoBERTa}^{\text{LSTM}}$	84.3	83.9	84.1	75.0	76.2	75.6	74.5	71.0	72.7	77.5
	$\text{RoBERTa}^{\text{LSTM}}_{\text{overlap}}$	84.1	84.8	84.5	75.4	77.4	76.4	74.5	72.4	73.5	78.1
	Longformer	82.4	85.3	83.8	73.0	78.6	75.7	72.6	72.5	72.5	77.4
	$\text{Longformer}_{\text{overlap}}$	82.4	83.8	83.1	72.9	76.0	74.4	72.3	70.5	71.4	76.3
	$\text{Longformer}^{\text{LSTM}}$	84.2	83.8	84.0	75.3	75.8	75.5	73.8	71.7	72.8	77.4
	$\text{Longformer}^{\text{LSTM}}_{\text{overlap}}$	84.3	84.2	84.2	75.4	76.6	76.0	74.4	72.3	73.4	77.9
	BigBird	81.5	86.9	84.1	71.5	80.9	75.9	72.7	73.1	72.9	77.6
	$\text{BigBird}_{\text{overlap}}$	83.0	84.3	83.6	73.9	76.7	75.3	72.4	72.0	72.2	77.0
	$\text{BigBird}^{\text{LSTM}}$	84.2	84.4	84.3	75.0	77.1	76.0	74.0	72.1	73.1	77.8
	$\text{BigBird}^{\text{LSTM}}_{\text{overlap}}$	84.5	84.3	84.4	75.7	76.8	76.2	74.8	72.2	73.5	78.0
	SpanBERT	83.3	82.9	83.1	74.4	74.8	74.6	72.6	71.7	72.1	76.6
	$\text{SpanBERT}_{\text{overlap}}$	83.1	83.0	83.0	74.4	75.0	74.7	72.2	72.2	72.2	76.7
	$\text{SpanBERT}^{\text{LSTM}}$	83.4	82.8	83.1	74.3	74.7	74.5	72.8	70.9	71.8	76.5
	$\text{SpanBERT}^{\text{LSTM}}_{\text{overlap}}$	83.6	82.6	83.1	74.4	74.6	74.5	72.6	70.8	71.7	76.4
Model: Start2End	RoBERTa	85.7	82.6	84.1	78.1	74.3	76.2	75.2	70.9	73.0	77.8
	$\text{RoBERTa}_{\text{overlap}}$	85.0	82.6	83.8	77.7	73.5	75.6	74.4	71.5	73.0	77.4
	$\text{RoBERTa}^{\text{LSTM}}$	83.8	81.4	82.6	75.1	72.3	73.7	72.1	69.4	70.8	75.7
	$\text{RoBERTa}^{\text{LSTM}}_{\text{overlap}}$	83.7	82.9	83.3	75.6	74.6	75.1	73.7	71.4	72.5	77.0
	Longformer	85.5	82.6	84.0	78.0	74.5	76.2	75.2	70.7	72.9	77.7
	$\text{Longformer}_{\text{overlap}}$	83.9	82.2	83.1	75.5	73.3	74.4	73.8	69.9	71.8	76.4
	$\text{Longformer}^{\text{LSTM}}$	84.0	80.8	82.3	75.7	71.2	73.4	72.5	68.3	70.3	75.3
	$\text{Longformer}^{\text{LSTM}}_{\text{overlap}}$	83.7	81.6	82.6	75.3	72.5	73.8	72.5	69.5	71.0	75.8
	BigBird	85.7	82.3	84.0	77.9	73.9	75.9	75.7	69.4	72.4	77.4
	$\text{BigBird}_{\text{overlap}}$	84.1	82.4	83.2	76.3	74.2	75.2	74.7	70.8	72.7	77.1
	$\text{BigBird}^{\text{LSTM}}$	83.2	81.7	82.4	74.2	72.5	73.3	72.1	68.5	70.3	75.3
	$\text{BigBird}^{\text{LSTM}}_{\text{overlap}}$	83.7	81.9	82.8	75.5	73.0	74.2	73.3	69.9	71.6	76.2
	SpanBERT	83.5	81.3	82.4	74.6	71.9	73.2	72.3	68.5	70.3	75.3
	$\text{SpanBERT}_{\text{overlap}}$	82.9	81.2	82.0	74.0	72.3	73.2	72.2	68.5	70.3	75.2
	$\text{SpanBERT}^{\text{LSTM}}$	70.4	58.9	64.1	47.8	44.7	46.2	59.6	26.7	36.9	49.1
	$\text{SpanBERT}^{\text{LSTM}}_{\text{overlap}}$	68.3	61.5	64.7	43.6	47.8	45.6	59.3	26.3	36.5	48.9

Table 7: The full results of all experiments with different pooling strategies. All models use the segment length L=512. Models with superscript "LSTM" indicate it uses LSTM, and subscript "overlap" indicates it uses overlapped concatenation method. Note that both methods can be applied in the meantime.

Encoder	MUC			B^3			CEAF$_{\phi_4}$			Avg.
	P	R	F1	P	R	F1	P	R	F1	
BigBird (L=128)	80.5	85.5	83.0	69.8	78.5	73.9	70.8	71.2	71.0	75.9
BigBird (L=256)	81.1	86.2	83.6	70.5	79.8	74.9	72.1	71.7	71.9	76.8
BigBird (L=512)	81.5	86.9	84.1	71.5	80.9	75.9	72.7	73.1	72.9	77.6
BigBird (L=1024)	82.2	85.5	83.8	72.8	78.4	75.5	72.7	72.5	72.6	77.3
BigBird (L=4096)	81.8	87.0	84.3	71.5	81.0	76.0	72.7	73.2	73.0	77.7
Longformer (L=128)	81.7	83.7	82.7	71.6	75.8	73.7	71.3	70.4	70.9	75.7
Longformer^G (L=128)	81.1	84.3	82.7	70.6	76.5	73.4	71.0	70.9	70.9	75.7
Longformer (L=256)	81.6	85.2	83.4	71.8	78.1	74.8	71.6	72.3	72.0	76.7
Longformer^G (L=256)	81.4	84.8	83.1	71.4	77.6	74.4	71.0	71.7	71.3	76.3
Longformer (L=512)	82.4	85.3	83.8	73.0	78.6	75.7	72.6	72.5	72.5	77.4
Longformer^G (L=512)	82.6	85.0	83.8	73.2	77.9	75.5	72.4	72.4	72.4	77.2
Longformer (L=1024)	82.1	84.9	83.5	72.3	77.7	74.9	72.0	72.0	72.0	76.8
Longformer (L=4096)	82.0	84.2	83.1	72.4	76.2	74.3	71.6	71.6	71.6	76.3
RoBERTa (L=128)	81.4	82.5	81.9	70.7	73.9	72.2	71.4	68.2	69.7	74.6
RoBERTa^p (L=128)	69.4	57.7	63.0	55.9	42.5	48.3	49.9	38.4	43.4	51.6
RoBERTa (L=256)	82.0	84.5	83.2	72.2	77.2	74.6	72.3	70.7	71.5	76.5
RoBERTa^p (L=256)	68.7	57.9	62.9	55.7	43.1	48.6	49.3	39.2	43.7	51.7
RoBERTa (L=512)	81.6	85.0	83.3	72.1	78.2	75.1	72.1	72.2	72.2	76.8
RoBERTa^p (L=512)	68.0	56.1	61.5	55.1	40.8	46.9	48.4	38.4	42.8	50.4
SpanBERT (L=128)	82.0	82.2	82.1	72.0	73.7	72.8	71.5	69.0	70.2	75.0
SpanBERT^p (L=128)	70.6	56.8	63.0	58.1	42.9	49.4	50.8	40.5	45.1	52.5
SpanBERT (L=256)	82.7	82.8	82.7	73.0	74.1	73.5	71.9	70.6	71.3	75.8
SpanBERT^p (L=256)	70.0	56.4	62.5	58.5	41.7	48.7	50.1	40.8	45.0	52.1
SpanBERT (L=512)	83.3	82.9	83.1	74.4	74.8	74.6	72.6	71.7	72.1	76.6
SpanBERT^p (L=512)	67.6	55.7	61.1	56.2	40.8	47.3	47.2	39.8	43.2	50.5
XLNet (L=128, m=0)	81.6	82.7	82.1	71.2	73.4	72.3	70.0	68.6	69.3	74.6
XLNet (L=128, m=128)	81.7	82.6	82.1	71.9	73.3	72.6	69.9	69.0	69.5	74.7
XLNet (L=256, m=0)	79.4	84.2	81.7	68.5	76.0	72.0	68.5	70.9	69.7	74.5
XLNet (L=256, m=256)	84.3	81.8	83.0	75.4	72.2	73.8	72.2	68.9	70.5	75.8
XLNet (L=512, m=0)	79.0	85.2	82.0	67.4	77.4	72.1	69.1	68.8	69.0	74.3
XLNet (L=512, m=512)	82.1	84.1	83.1	72.5	76.1	74.3	72.8	70.3	71.5	76.3

Table 8: Full results on Ontonotes with the coarse2fine model. L is the segment length used to chunk the text. m is the memory length used for the XLNet model. ^{G} denotes that the global tokens are used. ^{p} denotes that th self-attention computation is replaced with Performer kernels.

Encoder	L=128		L=256		L=512		L=1024		L=∞	
	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Longformer	54.26	50.02	58.83	54.48	63.88	59.13	63.91	58.91	63.41	58.89
Longformer^G	-	-	-	-	-	-	-	-	72.96	67.88
RoBERTa	55.81	50.73	60.29	56.11	63.45	58.84	-	-	-	-
RoBERTa^p	23.17	16.80	21.87	15.56	21.11	15.09	-	-	-	-
BigBird	55.28	50.66	59.39	54.34	63.51	58.50	66.50	61.15	71.78	66.86
XLNet	51.46	47.10	56.26	52.08	60.05	55.62	-	-	-	-
XLNet^m	52.71	48.03	57.96	52.93	62.85	58.13	-	-	-	-

Table 9: Full results on TriviaQA. We adopt the same notation as we used in table 8.

	Encoder	L=512			L=1024			L=1536			L=∞		
		R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
SF	BART	26.3	5.1	16.2	27.2	4.9	16.7	-	-	-	-	-	-
	LED	32.8	7.0	18.8	33.1	7.3	18.9	33.2	7.0	18.6	33.6	7.1	18.7
GR	BART	45.6	16.9	21.8	47.9	18.6	22.7	-	-	-	-	-	-
	LED	53.9	24.7	27.1	54.1	25.1	27.9	54.8	25.7	27.8	56.6	26.6	29.1

Table 10: Full results on summarization. "SS" stands for the SummScreen dataset, and "GR" stands for the GovReport dataset. The BART model does not support sequence longer than 1024 tokens.

Encoder	MUC			B^3			CEAF$_{\phi_4}$			Avg.
	P	R	F1	P	R	F1	P	R	F1	
BigBird (L=128)	84.5	78.5	81.4	75.2	68.6	71.7	73.5	63.2	68.0	73.7
BigBird (L=256)	85.1	80.3	82.6	76.7	71.2	73.8	74.7	66.4	70.3	75.6
BigBird (L=512)	85.7	82.3	84.0	77.9	73.9	75.9	75.7	69.4	72.4	77.4
BigBird (L=1024)	85.2	82.5	83.8	77.1	74.5	75.8	76.3	69.6	72.8	77.4
BigBird (L=4096)	85.1	82.8	83.9	77.7	75.1	76.4	75.6	70.1	72.7	77.7
Longformer (L=128)	84.4	80.0	82.1	75.3	70.5	72.8	72.9	66.0	69.3	74.8
Longformer^G (L=128)	84.4	79.1	81.7	75.1	69.2	72.0	72.8	65.2	68.8	74.2
Longformer (L=256)	84.8	81.7	83.2	76.2	72.5	74.3	73.9	68.8	71.2	76.3
Longformer^G (L=256)	84.4	81.8	83.1	75.6	73.1	74.3	74.3	68.3	71.2	76.2
Longformer (L=512)	85.5	82.6	84.0	78.0	74.5	76.2	75.2	70.7	72.9	77.7
Longformer^G (L=512)	84.5	83.4	83.9	76.2	75.2	75.7	74.3	70.9	72.6	77.4
Longformer (L=1024)	86.0	82.1	84.0	78.7	73.4	76.0	75.2	70.2	72.6	77.5
Longformer^G (L=1024)	82.4	79.2	80.8	72.2	69.5	70.8	72.2	65.4	68.6	73.4
Longformer (L=4096)	85.2	82.9	84.1	77.4	74.6	76.0	74.8	70.7	72.7	77.6
RoBERTa (L=128)	81.1	78.0	79.6	70.5	68.0	69.3	71.2	63.3	67.0	72.0
RoBERTa^p (L=128)	61.3	45.0	51.9	45.9	30.3	36.5	41.4	25.8	31.8	40.1
RoBERTa (L=256)	84.7	81.8	83.2	76.0	72.6	74.3	74.2	68.5	71.3	76.3
RoBERTa^p (L=256)	67.7	46.0	54.8	53.0	30.8	39.0	43.8	26.9	33.3	42.4
RoBERTa (L=512)	85.7	82.6	84.1	78.1	74.3	76.2	75.2	70.9	73.0	77.8
RoBERTa^p (L=512)	67.0	45.0	53.9	53.0	29.8	38.1	43.2	26.8	33.1	41.7
SpanBERT (L=128)	78.2	75.5	76.8	66.6	64.3	65.5	68.2	60.5	64.1	68.7
SpanBERT^p (L=128)	56.5	45.7	50.5	39.8	31.5	35.1	38.4	25.1	30.4	38.7
SpanBERT (L=256)	83.2	79.6	81.4	74.4	70.1	72.2	71.7	67.0	69.3	74.3
SpanBERT^p (L=256)	64.1	46.5	53.9	49.8	31.8	38.8	40.7	27.9	33.1	41.9
SpanBERT (L=512)	83.5	81.3	82.4	74.6	71.9	73.2	72.3	68.5	70.3	75.3
SpanBERT^p (L=512)	63.7	47.0	54.1	48.7	32.0	38.6	42.2	27.9	33.6	42.1
XLNet (L=128, m=0)	79.4	39.8	53.0	69.1	30.1	41.9	61.6	32.7	42.7	45.9
XLNet (L=128, m=128)	78.1	49.1	60.3	66.0	39.4	49.3	63.8	38.8	48.2	52.6
XLNet (L=256, m=0)	78.8	59.5	67.8	68.0	48.6	56.7	66.4	47.9	55.7	60.1
XLNet (L=256, m=256)	64.6	67.5	66.0	48.4	55.0	51.5	59.8	45.3	51.6	56.4
XLNet (L=512, m=0)	80.3	71.7	75.7	70.7	61.4	65.7	66.9	60.0	63.3	68.2
XLNet (L=512, m=512)	76.2	73.0	74.6	64.2	63.8	64.0	66.4	58.3	62.1	66.9

Table 11: Full results on Ontonotes with the start2end model. We adopt the same notation as we used in table 8.